Kerberos V5 Installation Guide

Release: 1.2
Document Edition: 1.1
Last updated: 28 February 2001

MIT

Table of Contents

1 Introduction

1.1 What is Kerberos and How Does it Work?

Kerberos V5 is based on the Kerberos authentication system developed at MIT. Under Kerberos, a client (generally either a user or a service) sends a request for a ticket to the Key Distribution Center (KDC). The KDC creates a *ticket-granting ticket* (TGT) for the client, encrypts it using the client's password as the key, and sends the encrypted TGT back to the client. The client then attempts to decrypt the TGT, using its password. If the client successfully decrypts the TGT (*i.e.*, if the client gave the correct password), it keeps the decrypted TGT, which indicates proof of the client's identity.

The TGT, which expires at a specified time, permits the client to obtain additional tickets, which give permission for specific services. The requesting and granting of these additional tickets is user-transparent.

1.2 Why Should I use Kerberos?

Since Kerberos negotiates authenticated, and optionally encrypted, communications between two points anywhere on the Internet, it provides a layer of security that is not dependent on which side of a firewall either client is on. Since studies have shown that half of the computer security breaches in industry happen from *inside* firewalls, Kerberos V5 from MIT will play a vital role in the security of your network.

This document is one piece of the document set for Kerberos V5. The documents, and their intended audiences, are:

- **Kerberos V5 Installation Guide**: a concise guide for installing Kerberos V5. Kerberos administrators (particularly whoever will be making site-wide decisions about the installation) and the system administrators who will be installing the software should read this guide.
- **Kerberos V5 System Administrator's Guide**: a sysadmin's guide to administering a Kerberos installation. The System Administrator's Guide describes the administration software and suggests policies and procedures for administering a Kerberos installation. Anyone who will have administrative access to your Kerberos database should read this guide.
- **Kerberos V5 UNIX User's Guide**: a guide to using the Kerberos UNIX client programs. All users on UNIX systems should read this guide, particularly the "Tutorial" section.

1.3 Please Read the Documentation

As with any software package that uses a centrallized database, the installation procedure is somewhat involved, and requires forethought and planning. MIT has attempted to make this Kerberos V5 Installation Guide as concise as possible, rather than making it an exhaustive description of the details of Kerberos. Consequently, everything in this guide appears because MIT believes that it is important. Please read and follow these instructions carefully.

1.4 Overview of This Guide

The next chapter describes the decisions you need to make before installing Kerberos V5.

Chapter four describes installation procedures for each class of Kerberos machines:

1. Key Distribution Centers (KDCs).
   A. The Master KDC.
   B. Slave KDCs.
2. UNIX client machines
3. UNIX application server machines

Note that a machine can be both a client machine and an application server.

Chapter five describes procedure for updating previous installations of Kerberos V5.

Chapter six describes our problem reporting system.

The appendices give sample configuration files.

2 Realm Configuration Decisions

Before installing Kerberos V5, it is necessary to consider the following issues:

- The name of your Kerberos realm (or the name of each realm, if you need more than one).
- How you will map your hostnames onto Kerberos realms.
- Which ports your KDC and and kadmin (database access) services will use.
- How many slave KDCs you need and where they should be located.
- The hostnames of your master and slave KDCs.
- How frequently you will propagate the database from the master KDC to the slave KDCs.
- Whether you need backward compatibility with Kerberos V4.

2.1 Kerberos Realms

Although your Kerberos realm can be any ASCII string, convention is to make it the same as your domain name, in upper-case letters. For example, hosts in the domain fubar.org would be in the Kerberos realm FUBAR.ORG.

If you need multiple Kerberos realms, MIT recommends that you use descriptive names which end with your domain name, such as BOSTON.FUBAR.ORG and HOUSTON.FUBAR.ORG.

2.2 Mapping Hostnames onto Kerberos Realms

Mapping hostnames onto Kerberos realms is done in one of two ways.

The first mechanism, which has been in use for years in MIT-based Kerberos distributions, works through a set of rules in the `krb5.conf` configuration file. (See Section A.1 [krb5.conf], page 39.) You can specify mappings for an entire domain or subdomain, and/or on a hostname-by-hostname basis. Since greater specificity takes precedence, you would do this by specifying the mappings for a given domain or subdomain and listing the exceptions.

The Kerberos V5 System Administrator's Guide contains a thorough description of the parts of the `krb5.conf` file and what may be specified in each. A sample `krb5.conf` file appears in Section A.1 [krb5.conf], page 39. You should be able to use this file, substituting the relevant information for your Kerberos installation for the samples.

The second mechanism, recently introduced into the MIT code base but not currently used by default, works by looking up the information in special `TXT` records in the Domain Name Service. If this mechanism is enabled on the client, it will try to look up a `TXT` record for the DNS name formed by putting the prefix `_kerberos` in front of the hostname in question. If that record is not found, it will try using `_kerberos` and the host's domain name, then its parent domain, and so forth. So for the hostname BOSTON.ENGINEERING.FOOBAR.COM, the names looked up would be:

```
_kerberos.boston.engineering.foobar.com
_kerberos.engineering.foobar.com
_kerberos.foobar.com
_kerberos.com
```

The value of the first TXT record found is taken as the realm name. (Obviously, this doesn't work all that well if a host and a subdomain have the same name, and different realms. For example, if all the hosts in the ENGINEERING.FOOBAR.COM domain are in the ENGINEERING.FOOBAR.COM realm, but a host named ENGINEERING.FOOBAR.COM is for some reason in another realm. In that case, you would set up TXT records for all hosts, rather than relying on the fallback to the domain name.)

Even if you do not choose to use this mechanism within your site, you may wish to set up anyways, for use when interacting with other sites.

2.3 Ports for the KDC and Admin Services

The default ports used by Kerberos are port 88 for the KDC[1] and port 749 for the admin server. You can, however, choose to run on other ports, as long as they are specified in each host's `/etc/services` and `krb5.conf` files, and the `kdc.conf` file on each KDC. For a more thorough treatment of port numbers used by the Kerberos V5 programs, refer to the "Configuring Your Firewall to Work With Kerberos V5" section of the *Kerberos V5 System Administrator's Guide*.

2.4 Slave KDCs

Slave KDCs provide an additional source of Kerberos ticket-granting services in the event of inaccessibility of the master KDC. The number of slave KDCs you need and the decision of where to place them, both physically and logically, depends on the specifics of your network.

All of the Kerberos authentication on your network requires that each client be able to contact a KDC. Therefore, you need to anticipate any likely reason a KDC might be unavailable and have a slave KDC to take up the slack.

Some considerations include:

- Have at least one slave KDC as a backup, for when the master KDC is down, is being upgraded, or is otherwise unavailable.

[1] Kerberos V4 used port 750. If necessary, you can run on both ports for backward compatibility.

- If your network is split such that a network outage is likely to cause a network partition (some segment or segments of the network to become cut off or isolated from other segments), have a slave KDC accessible to each segment.
- If possible, have at least one slave KDC in a different building from the master, in case of power outages, fires, or other localized disasters.

2.5 Hostnames for the Master and Slave KDCs

MIT recommends that your KDCs have a predefined set of CNAME records (DNS hostname aliases), such as `kerberos` for the master KDC and `kerberos-1`, `kerberos-2`, ... for the slave KDCs. This way, if you need to swap a machine, you only need to change a DNS entry, rather than having to change hostnames.

A new mechanism for locating KDCs of a realm through DNS has been added to the MIT Kerberos V5 distribution. A relatively new record type called `SRV` has been added to DNS. Looked up by a service name and a domain name, these records indicate the hostname and port number to contact for that service, optionally with weighting and prioritization. (See RFC 2782 if you want more information. You can follow the example below for straightforward cases.)

The use with Kerberos is fairly straightforward. The domain name used in the SRV record name is the domain-style Kerberos realm name. (It is possible to have Kerberos realm names that are not DNS-style names, but we don't recommend it for Internet use, and our code does not support it well.) Several different Kerberos-related service names are used:

`_kerberos._udp`

This is for contacting any KDC. This entry will be used the most often. Normally you should list ports 88 and 750 on each of your KDCs.

`_kerberos-master._udp`

This entry should refer to those KDCs, if any, that will immediately see password changes to the Kerberos database. This entry is used only in one case, when the user is logging in and the password appears to be incorrect; the master KDC is then contacted, and the same password used to try to decrypt the response, in case the user's password had recently been changed and the first KDC contacted hadn't been updated. Only if that fails is an "incorrect password" error given.

If you have only one KDC, or for whatever reason there is no accessible KDC that would get database changes faster than the others, you do not need to define this entry.

`_kerberos-adm._tcp`

This should list port 749 on your master KDC. Support for it is not complete at this time, but it will eventually be used by the `kadmin` program and related utilities. For now, you will also need the `admin_server` entry in `krb5.conf`.

`_kpasswd._udp`
: This should list port 464 on your master KDC. It is used when a user changes her password.

Be aware, however, that the DNS SRV specification requires that the hostnames listed be the canonical names, not aliases. So, for example, you might include the following records in your (BIND-style) zone file:

```
$ORIGIN foobar.com.
_kerberos               TXT       "FOOBAR.COM"
kerberos                CNAME     daisy
kerberos-1              CNAME     use-the-force-luke
kerberos-2              CNAME     bunny-rabbit
_kerberos._udp          SRV       0 0 88 daisy
                        SRV       0 0 88 use-the-force-luke
                        SRV       0 0 88 bunny-rabbit
_kerberos-master._udp   SRV       0 0 88 daisy
_kerberos-adm._tcp      SRV       0 0 749 daisy
_kpasswd._udp           SRV       0 0 464 daisy
```

As with the DNS-based mechanism for determining the Kerberos realm of a host, we recommend distributing the information this way for use by other sites that may want to interact with yours using Kerberos, even if you don't immediately make use of it within your own site. If you anticipate installing a very large number of machines on which it will be hard to update the Kerberos configuration files, you may wish to do all of your Kerberos service lookups via DNS and not put the information (except for `admin_server` as noted above) in future versions of your `krb5.conf` files at all. Eventually, we hope to phase out the listing of server hostnames in the client-side configuration files; making preparations now will make the transition easier in the future.

2.6 Database Propagation

The Kerberos database resides on the master KDC, and must be propagated regularly (usually by a cron job) to the slave KDCs. In deciding how frequently the propagation should happen, you will need to balance the amount of time the propagation takes against the maximum reasonable amount of time a user should have to wait for a password change to take effect.

If the propagation time is longer than this maximum reasonable time (*e.g.*, you have a particularly large database, you have a lot of slaves, or you experience frequent network delays), you may wish to cut down on your propagation delay by performing the propagation in parallel. To do this, have the master KDC propagate the database to one set of slaves, and then have each of these slaves propagate the database to additional slaves.

3 Building Kerberos V5

Starting with the Beta 4 distribution, we are using a new configuration system, which was built using the Free Software Foundation's '`autoconf`' program. This system will hopefully make Kerberos V5 much simpler to build and reduce the amount of effort required in porting Kerberos V5 to a new platform.

3.1 Build Requirements

In order to build Kerberos V5, you will need approximately 60-70 megabytes of disk space. The exact amount will vary depending on the platform and whether the distribution is compiled with debugging symbol tables or not.

If you wish to keep a separate *build tree*, which contains the compiled '`*.o`' file and executables, separate from your source tree, you will need a '`make`' program which supports '`VPATH`', or you will need to use a tool such as '`lndir`' to produce a symbolic link tree for your build tree.

3.2 Unpacking the Sources

The first step in each of these build procedures is to unpack the source distribution. The Kerberos V5 distribution comes in two compressed tar files. The first file, which is generally named '`krb5-1.2.src.tar.gz`', contains the sources for all of Kerberos except for the crypto library, which is found in the file '`krb5-1.2.crypto.tar.gz`'.

Both files should be unpacked in the same directory, such as '`/u1/krb5-1.2`'. (In the rest of this document, we will assume that you have chosen to unpack the Kerberos V5 source distribution in this directory. Note that the tarfiles will by default all unpack into the '`./krb5-1.2`' directory, so that if your current directory is '`/u1`' when you unpack the tarfiles, you will get '`/u1/krb5-1.2/src`', etc.)

3.3 Doing the Build

You have a number of different options in how to build Kerberos. If you only need to build Kerberos for one platform, using a single directory tree which contains both the source files and the object files is the simplest. However, if you need to maintain Kerberos for a large number of platforms, you will probably want to use separate build trees for each platform. We recommend that you look at Section 3.8 [OS Incompatibilities], page 13, for notes that we have on particular operating systems.

3.3.1 Building Within a Single Tree

If you don't want separate build trees for each architecture, then use the following abbreviated procedure.

1. `cd /u1/krb5-1.2/src`
2. `./configure`
3. `make`

That's it!

3.3.2 Building with Separate Build Directories

If you wish to keep separate build directories for each platform, you can do so using the following procedure. (Note, this requires that your '`make`' program support '`VPATH`'. GNU's make will provide this functionality, for example.) If your '`make`' program does not support this, see the next section.

For example, if you wish to create a build directory for `pmax` binaries you might use the following procedure:

1. `mkdir /u1/krb5-1.2/pmax`
2. `cd /u1/krb5-1.2/pmax`
3. `../src/configure`
4. `make`

3.3.3 Building Using '`lndir`'

If you wish to keep separate build directories for each platform, and you do not have access to a '`make`' program which supports '`VPATH`', all is not lost. You can use the '`lndir`' program to create symbolic link trees in your build directory.

For example, if you wish to create a build directory for solaris binaries you might use the following procedure:

1. `mkdir /u1/krb5-1.2/solaris`
2. `cd /u1/krb5-1.2/solaris`
3. `` /u1/krb5-1.2/src/util/lndir `pwd`/../src ``
4. `./configure`
5. `make`

You must give an absolute pathname to '`lndir`' because it has a bug that makes it fail for relative pathnames. Note that this version differs from the latest version as distributed and installed by the XConsortium with X11R6. Either version should be acceptable.

3.4 Testing the Build

The Kerberos V5 distribution comes with built-in regression tests. To run them, simply type the following command while in the top-level build directory (i.e., the directory where you sent typed '`make`' to start building Kerberos; see Section 3.3 [Doing the Build], page 7.):

```
% make check
```

3.4.1 The DejaGnu Tests

Some of the built-in regression tests are setup to use the DejaGnu framework for running tests. These tests tend to be more comprehensive than the normal built-in tests as they setup test servers and test client/server activities.

DejaGnu may be found wherever GNU software is archived.

Most of the tests are setup to run as a non-privledged user. For some of the krb-root tests to work properly, either (a) the user running the tests must not have a .k5login file in the home directory or (b) the .k5login file must contain an entry for `<username>@KRBTEST.COM`. There are two series of tests ('`rlogind`' and '`telnetd`') which require the ability to '`rlogin`' as root to the local machine. Admittedly, this does require the use of a '`.rhosts`' file or some authenticated means.[1]

If you cannot obtain root access to your machine, all the other tests will still run. Note however, with DejaGnu 1.2, the "untested testcases" will cause the testsuite to exit with a non-zero exit status which '`make`' will consider a failure of the testing process. Do not worry about this, as these tests are the last run when '`make check`' is executed from the top level of the build tree. This problem does not exist with DejaGnu 1.3.

3.4.2 The KADM5 Tests

Regression tests for the KADM5 system, including the GSS-RPC, KADM5 client and server libraries, and kpasswd, are also included in this release. Each set of KADM5 tests is contained in a sub-directory called `unit-test` directly below the system being tested. For example, lib/rpc/unit-test contains the tests for GSS-RPC. The tests are all based on DejaGnu (but they are not actually called part of "The DejaGnu tests," whose naming predates the inclusion of the KADM5 system). In addition, they require the Tool Command Language (TCL) header files and libraries to be available during compilation and some of the tests also require Perl in order to operate. If all of these resources are not available during configuration, the KADM5 tests will not run. The TCL

[1] If you are fortunate enough to have a previous version of Kerberos V5 or V4 installed, and the Kerberos rlogin is first in your path, you can setup '`.k5login`' or '`.klogin`' respectively to allow you access.

installation directory can be specified with the `--with-tcl` configure option. (See See Section 3.5 [Options to Configure], page 10.) The runtest and perl programs must be in the current execution path.

If you install DejaGnu, TCL, or Perl after configuring and building Kerberos and then want to run the KADM5 tests, you will need to re-configure the tree and run `make` at the top level again to make sure all the proper programs are built. To save time, you actually only need to reconfigure and build in the directories src/kadmin/testing, src/lib/rpc, src/lib/kadm5.

3.5 Options to Configure

There are a number of options to '`configure`' which you can use to control how the Kerberos distribution is built. The following table lists the most commonly used options to Kerberos V5's '`configure`' program.

`--help`
: Provides help to configure. This will list the set of commonly used options for building Kerberos.

`--prefix=PREFIX`
: By default, Kerberos will install the package's files rooted at '/usr/local' as in '/usr/local/bin', '/usr/local/sbin', etc. If you desire a different location, use this option.

`--exec-prefix=EXECPREFIX`
: This option allows one to separate the architecture independent programs from the configuration files and manual pages.

`--localstatedir=LOCALSTATEDIR`
: This option sets the directory for locally modifiable single-machine data. In Kerberos, this mostly is useful for setting a location for the KDC data files, as they will be installed in `LOCALSTATEDIR/krb5kdc`, which is by default `PREFIX/var/krb5kdc`.

`--with-cc=COMPILER`
: Use `COMPILER` as the C compiler.

`--with-ccopts=FLAGS`
: Use `FLAGS` as the default set of C compiler flags.

  Note that if you use the native Ultrix compiler on a DECstation you are likely to lose if you pass no flags to cc; md4.c takes an estimated 3,469 billion years to compile if you provide neither the '`-g`' flag nor the '`-O`' flag to '`cc`'.

`--with-cppopts=CPPOPTS`
: Use `CPPOPTS` as the default set of C preprocessor flags. The most common use of this option is to select certain `#define`'s for use with the operating system's include files.

`--with-linker=LINKER`
: Use `LINKER` as the default loader if it should be different from C compiler as specified above.

`--with-ldopts=LDOPTS`
: This option allows one to specify optional arguments to be passed to the linker. This might be used to specify optional library paths.

`--with-krb4`
: This option enables Kerberos V4 backwards compatibility using the builtin Kerberos V4 library.

`--with-krb4=KRB4DIR`
: This option enables Kerberos V4 backwards compatibility using a pre-existing Kerberos V4 installation. The directory specified by `KRB4DIR` specifies where the V4 header files should be found (‘`KRB4DIR/include`’) as well as where the V4 Kerberos library should be found (‘`KRB4DIR/lib`’).

`--without-krb4`
: Disables Kerberos V4 backwards compatibility. This prevents Kerberos V4 clients from using the V5 services including the KDC. This would be useful if you know you will never install or need to interact with V4 clients.

`--with-netlib[=libs]`
: Allows for suppression of or replacement of network libraries. By default, Kerberos V5 configuration will look for `-lnsl` and `-lsocket`. If your operating system has a broken resolver library (see Section 3.8.6 [Solaris versions 2.0 through 2.3], page 14) or fails to pass the tests in ‘`src/tests/resolv`’ you will need to use this option.

`--with-vague-errors`
: If enabled, gives vague and unhelpful error messages to the client... er, attacker. (Needed to meet silly government regulations; most other sites will want to keep this undefined.)

`--with-kdc-kdb-update`
: Set this option if you want to allow the KDC to modify the Kerberos database; this allows the last request information to be updated, as well as the failure count information. Note that this doesn't work if you're using slave servers!!! It also causes the database to be modified (and thus needing to be locked) frequently. Please note that the implementors do not regularly test this feature.

`--with-tcl=TCLPATH`
: Some of the unit-tests in the build tree rely upon using a program in Tcl. The directory specified by `TCLPATH` specifies where the Tcl header file (‘`TCLPATH/include/tcl.h`’ as well as where the Tcl library should be found (‘`TCLPATH/lib`’).

`--enable-shared`
: This option will turn on the building and use of shared library objects in the Kerberos build. This option is only supported on certain platforms.

`--enable-dns`

`--enable-dns-for-kdc`

`--enable-dns-for-realm`
: Enable the use of DNS to look up a host's Kerberos realm, or a realm's KDCs, if the information is not provided in krb5.conf. See See Section 2.5 [Hostnames for the Master and Slave KDCs], page 5, and See Section 2.2 [Mapping Hostnames onto Kerberos Realms], page 3. By default, DNS lookups are enabled for the latter but not for the former.

`--enable-kdc-replay-cache`
: Enable a cache in the KDC to detect retransmitted messages, and resend the previous responses to them. This protects against certain types of attempts to extract information from the KDC through some of the hardware preauthentication systems.

For example, in order to configure Kerberos on a Solaris machine using the '`suncc`' compiler with the optimizer turned on, run the configure script with the following options:

```
% ./configure --with-cc=suncc --with-ccopts=-O
```

3.6 '`osconf.h`'

There is one configuration file which you may wish to edit to control various compile-time parameters in the Kerberos distribution: '`include/krb5/stock/osconf.h`'. The list that follows is by no means complete, just some of the more interesting variables.

Please note: The former configuration file '`config.h`' no longer exists as its functionality has been merged into the auto-configuration process. See Section 3.5 [Options to Configure], page 10.

`DEFAULT_PROFILE_PATH`
: The pathname to the file which contains the profiles for the known realms, their KDCs, etc.

  The profile file format is no longer the same format as Kerberos V4's '`krb.conf`' file.

`DEFAULT_KEYTAB_NAME`
: The type and pathname to the default server keytab file (the equivalent of Kerberos V4's '`/etc/srvtab`').

`DEFAULT_KDC_ENCTYPE`
: The default encryption type for the KDC.

`KDCRCACHE`
: The name of the replay cache used by the KDC.

`RCTMPDIR`
: The directory which stores replay caches.

`DEFAULT_KDB_FILE`
: The location of the default database

3.7 Shared Library Support

Shared library support is provided for a few operating systems. There are restrictions as to which compiler to use when using shared libraries. In all cases, executables linked with the shared libraries in this build process will have built in the location of the libraries, therefore obliterating the need for special LD_LIBRARY_PATH, et al environment variables when using the programs. Except where noted, multiple versions of the libraries may be installed on the same system and continue to work.

Currently the supported platforms are Solaris 2.6 (aka SunOS 5.6) and Irix 6.5.

Shared library support has been tested on the following platforms but not exhaustively (they have been built but not necessarily tested in an installed state): Tru64 (aka Alpha OSF/1 or Digital Unix) 4.0, NetBSD 1.4.x (i386), and HP/UX 10.20.

Platforms for which there is shared library support but not significant testing include FreeBSD, OpenBSD, MacOS 10, AIX, Linux, and SunOS 4.x.

To enable shared libraries on the above platforms, run the configure script with the option '`--enable-shared`'.

3.8 Operating System Incompatibilities

This section details operating system incompatibilities with Kerberos V5 which have been reported to the developers at MIT. If you find additional incompatibilities, and/or discover work arounds to such problems, please send a report via the `krb5-send-pr` program. Thanks!

3.8.1 AIX

The AIX 3.2.5 linker dumps core trying to build a shared '`libkrb5.a`' produced with the GNU C compiler. The native AIX compiler works fine. This problem is fixed using the AIX 4.1 linker.

3.8.2 Alpha OSF/1 V1.3

Using the native compiler, compiling with the '`-O`' compiler flag causes the `asn.1` library to be compiled incorrectly.

Using GCC version 2.6.3 or later instead of the native compiler will also work fine, both with or without optimization.

3.8.3 Alpha OSF/1 V2.0++

There used to be a bug when using the native compiler in compiling '`md4.c`' when compiled without either the '`-O`' or '`-g`' compiler options. We have changed the code and there is no problem under V2.1, but we do not have access to V2.0 to test and see if the problem would exist there. (We welcome feedback on this issue). There was never a problem in using GCC version 2.6.3.

In version 3.2 and beyond of the operating system, we have not seen any problems with the native compiler.

3.8.4 BSDI

BSDI versions 1.0 and 1.1 reportedly has a bad '`sed`' which causes it to go into an infinite loop during the build. The work around is to use a '`sed`' from somewhere else, such as GNU. (This may be true for some versions of other systems derived from BSD 4.4, such as NetBSD and FreeBSD.)

3.8.5 HPUX

The native (bundled) compiler for HPUX currently will not work, because it is not a full ANSI C compiler. The optional compiler (c89) should work as long as you give it the '`-D_HPUX_SOURCE`' flag (i.e. '`./configure --with-cc='c89 -D_HPUX_SOURCE'`'). This has only been tested recently for HPUX 10.20.

3.8.6 Solaris versions 2.0 through 2.3

The `gethostbyname()` routine is broken; it does not return a fully qualified domain name, even if you are using the Domain Name Service routines. Since Kerberos V5 uses the fully qualified domain name as the second component of a service principal (i.e, '`host/tsx-11.mit.edu@ATHENA.MIT.EDU`'), this causes problems for servers who try to figure out their own fully qualified domain name.

Workarounds:

1. Supply your own resolver library. (such as bind-4.9.3pl1 available from ftp.vix.com)
2. Upgrade to Solaris 2.4

3. Make sure your /etc/nsswitch.conf has 'files' before 'dns' like:

   ```
   hosts:      files dns
   ```

   and then in /etc/hosts, make sure there is a line with your workstation's IP address and hostname, with the fully qualified domain name first. Example:

   ```
   18.172.1.4      dcl.mit.edu dcl
   ```

   Note that making this change may cause other programs in your environment to break or behave differently.

3.8.7 Solaris 2.X

You **must** compile Kerberos V5 without the UCB compatibility libraries. This means that '`/usr/ucblib`' must not be in the LD_LIBRARY_PATH environment variable when you compile it. Alternatively you can use the `-i` option to '`cc`', by using the specifying `--with-ccopts=-i` option to '`configure`'.

3.8.8 SGI Irix 5.X

If you are building in a tree separate from the source tree, the vendors version of make does not work properly with regards to '`VPATH`'. It also has problems with standard inference rules in 5.2 (not tested yet in 5.3) so one needs to use GNU's make.

Under 5.2, there is a bug in the optional System V `-lsocket` library in which the routine `gethostbyname()` is broken. The system supplied version in `-lc` appears to work though so one may simply specify `--with-netlib` option to '`configure`'.

In 5.3, `gethostbyname()` is no longer present in `-lsocket` and is no longer an issue.

3.8.9 Ultrix 4.2/3

The DEC MIPS platform currently will not support the native compiler, since the Ultrix compiler is not a full ANSI C compiler. You should use GCC instead.

3.9 Using 'Autoconf'

(If you are not a developer, you can skip this section.)

In most of the Kerberos V5 source directories, there is a 'configure' script which automatically determines the compilation environment and creates the proper Makefiles for a particular platform. These 'configure' files are generated using 'autoconf' version 2.4, which can be found in the 'src/util/autoconf' directory in the distribution.

Normal users will not need to worry about running 'autoconf'; the distribution comes with the 'configure' files already prebuilt. Developers who wish to modify the 'configure.in' files should see section "Overview" in *The Autoconf Manual*.

Note that in order to run 'autoconf', you must have GNU 'm4' in your path. Before you use the 'autoconf' in the Kerberos V5 source tree, you may also need to run 'configure', and then run 'make' in the 'src/util/autoconf' directory in order to properly set up 'autoconf'.

One tool which is provided for the convenience of developers can be found in 'src/util/reconf'. This program should be run while the current directory is the top source directory. It will automatically rebuild any 'configure' files which need rebuilding. If you know that you have made a change that will require that all the 'configure' files need to be rebuilt from scratch, specify the --force option:

```
% cd /u1/krb5-1.2/src
% ./util/reconf --force
```

The developmental sources are a raw source tree (before it's been packaged for public release), without the pre-built 'configure' files. In order to build from such a source tree, you must do:

```
% cd krb5/util/autoconf
% ./configure
% make
% cd ../..
% util/reconf
```

Then follow the instructions for building packaged source trees (above). To install the binaries into a binary tree, do:

```
% cd /u1/krb5-1.2/src
% make all
% make install DESTDIR=somewhere-else
```

4 Installing Kerberos V5

The sections of this chapter describe procedures for installing Kerberos V5 on:

1. The KDCs
2. UNIX client machines
3. UNIX Application Servers

4.1 Installing KDCs

The Key Distribution Centers (KDCs) issue Kerberos tickets. Each KDC contains a copy of the Kerberos database. The master KDC contains the master copy of the database, which it propagates to the slave KDCs at regular intervals. All database changes (such as password changes) are made on the master KDC.

Slave KDCs provide Kerberos ticket-granting services, but not database administration. This allows clients to continue to obtain tickets when the master KDC is unavailable.

MIT recommends that you install all of your KDCs to be able to function as either the master or one of the slaves. This will enable you to easily switch your master KDC with one of the slaves if necessary. (See Section 4.1.7 [Switching Master and Slave KDCs], page 26.) This installation procedure is based on that recommendation.

4.1.1 Install the Master KDC

This installation procedure will require you to go back and forth a couple of times between the master KDC and each of the slave KDCs. The first few steps must be done on the master KDC.

4.1.1.1 Edit the Configuration Files

Modify the configuration files, `/etc/krb5.conf` (see Section A.1 [krb5.conf], page 39) and `/usr/local/var/krb5kdc/kdc.conf` (see Section A.2 [kdc.conf], page 40) to reflect the correct information (such as the hostnames and realm name) for your realm. MIT recommends that you keep `krb5.conf` in `/etc`.

Among the settings in your `/etc/krb5.conf` file, be sure to create a `[logging]` stanza so that the KDC and kadmind will generate logging output. For example:

```
[logging]
    kdc = FILE:/var/log/krb5kdc.log
    admin_server = FILE:/var/log/kadmin.log
    default = FILE:/var/log/krb5lib.log
```

4.1.1.2 Create the Database

You will use the `kdb5_util` command *on the Master KDC* to create the Kerberos database and the optional stash file. The *stash file* is a local copy of the master key that resides in encrypted form on the KDC's local disk. The stash file is used to authenticate the KDC to itself automatically before starting the `kadmind` and `krb5kdc` daemons (*e.g.*, as part of the machine's boot sequence). The stash file, like the keytab file (see See Section 4.3.3 [The Keytab File], page 30, for more information) is a potential point-of-entry for a break-in, and if compromised, would allow unrestricted access to the Kerberos database. If you choose to install a stash file, it should be readable only by root, and should exist only on the KDC's local disk. The file should not be part of any backup of the machine, unless access to the backup data is secured as tightly as access to the master password itself.

Note that `kdb5_util` will prompt you for the master key for the Kerberos database. This key can be any string. A good key is one you can remember, but that no one else can guess. Examples of bad keys are words that can be found in a dictionary, any common or popular name, especially a famous person (or cartoon character), your username in any form (*e.g.*, forward, backward, repeated twice, *etc.*), and any of the sample keys that appear in this manual. One example of a key which might be good if it did not appear in this manual is “MITiys4K5!”, which represents the sentence “MIT is your source for Kerberos 5!” (It's the first letter of each word, substituting the numeral “4” for the word “for”, and includes the punctuation mark at the end.)

The following is an example of how to create a Kerberos database and stash file on the master KDC, using the `kdb5_util` command. (The line that begins with ⇒ is a continuation of the previous line.) Replace *ATHENA.MIT.EDU* with the name of your Kerberos realm.

```
shell% /usr/local/sbin/kdb5_util create -r ATHENA.MIT.EDU -s
Initializing database '/usr/local/var/krb5kdc/principal' for
⇒ realm 'ATHENA.MIT.EDU',
master key name 'K/M@ATHENA.MIT.EDU'
You will be prompted for the database Master Password.
It is important that you NOT FORGET this password.

Enter KDC database master key:   ⇐ Type the master password.
Re-enter KDC database master key to verify:   ⇐ Type it again.

shell%
```

This will create five files in the directory specified in your `kdc.conf` file: two Kerberos database files, `principal.db`, and `principal.ok`; the Kerberos administrative database file, `principal.kadm5`; the administrative database lock file, `principal.kadm5.lock`; and the stash file, `.k5stash`. (The default directory is `/usr/local/var/krb5kdc`.) If you do not want a stash file, run the above command without the `-s` option.

4.1.1.3 Add Administrators to the Acl File

Next, you need create an Access Control List (acl) file, and put the Kerberos principal of at least one of the administrators into it. The filename should match the value you have set for "acl_file" in your `kdc.conf` file. The default file name is '`kadm5.acl`'. The format of the file is:

```
Kerberos principal      permissions     optional target principal
```

The Kerberos principal (and optional target principal) can include the "`*`" wildcard, so if you want any principal with the instance "admin" to have full permissions on the database, you could use the principal "`*/admin@REALM`" where "REALM" is your Kerberos realm.

Note: a common use of an *admin* instance is so you can grant separate permissions (such as administrator access to the Kerberos database) to a separate Kerberos principal. For example, the user `joeadmin` might have a principal for his administrative use, called `joeadmin/admin`. This way, `joeadmin` would obtain `joeadmin/admin` tickets only when he actually needs to use those permissions. Refer to the Kerberos V5 Administrator's Guide or the Kerberos V5 User's Guide for more detailed explanations of *principals* and *instances*.

The permissions (acls) recognized in the acl file are the following:

a	allows the addition of principals or policies in the database.
A	prohibits the addition of principals or policies in the database.
d	allows the deletion of principals or policies in the database.
D	prohibits the deletion of principals or policies in the database.
m	allows the modification of principals or policies in the database.
M	prohibits the modification of principals or policies in the database.
c	allows the changing of passwords for principals in the database.
C	prohibits the changing of passwords for principals in the database.
i	allows inquiries to the database.
I	prohibits inquiries to the database.
l	allows the listing of principals or policies in the database.
L	prohibits the listing of principals or policies in the database.
*****	Short for all privileges (admcil).
x	Short for all privileges (admcil); identical to "*".

To give the principal `*/admin@ATHENA.MIT.EDU` permission to change all of the database permissions on any principal permissions, you would place the following line in the file:

```
*/admin@ATHENA.MIT.EDU  *
```

To give the principal `joeadmin@ATHENA.MIT.EDU` permission to add, list, and inquire about any principal that has the instance "root", you would add the following line to the acl file:

```
joeadmin@ATHENA.MIT.EDU  ali  */root@ATHENA.MIT.EDU
```

4.1.1.4 Add Administrators to the Kerberos Database

Next you need to add administrative principals to the Kerberos database. (You must add at least one now.) To do this, use `kadmin.local` *on the master KDC*. The administrative principals you create should be the ones you added to the ACL file. (See See Section 4.1.1.3 [Add Administrators to the Acl File], page 19.) In the following example, the administration principal `admin/admin` is created:

```
shell% /usr/local/sbin/kadmin.local
kadmin.local: addprinc admin/admin@ATHENA.MIT.EDU
WARNING: no policy specified for "admin/admin@ATHENA.MIT.EDU";
defaulting to no policy.

Enter password for principal admin/admin@ATHENA.MIT.EDU:  ⇐ Enter a password.
Re-enter password for principal admin/admin@ATHENA.MIT.EDU:  ⇐ Type it again.

Principal "admin/admin@ATHENA.MIT.EDU" created.
kadmin.local:
```

4.1.1.5 Create a kadmind Keytab

The kadmind keytab is the key that kadmind will use to decrypt administrators' Kerberos tickets to determine whether or not it should give them access to the database. You need to create the kadmin keytab with entries for the principals `kadmin/admin` and `kadmin/changepw`. (These principals are placed in the Kerberos database automatically when you create it.) To create the kadmin keytab, run `kadmin.local` and use the `ktadd` command, as in the following example. (The line beginning with ⇒ is a continuation of the previous line.):

```
shell% /usr/local/sbin/kadmin.local
kadmin.local: ktadd -k /usr/local/var/krb5kdc/kadm5.keytab
⇒ kadmin/admin kadmin/changepw
Entry for principal kadmin/admin@ATHENA.MIT.EDU with
    kvno 3, encryption type DES-CBC-CRC added to keytab
    WRFILE:/usr/local/var/krb5kdc/kadm5.keytab.
Entry for principal kadmin/changepw@ATHENA.MIT.EDU with
    kvno 3, encryption type DES-CBC-CRC added to keytab
    WRFILE:/usr/local/var/krb5kdc/kadm5.keytab.
kadmin.local: quit
shell%
```

As specified in the '`-k`' argument, `ktadd` will save the extracted keytab as `/usr/local/var/krb5kdc/kadm5.keytab`. The filename you use must be the one specified in your `kdc.conf` file.

4.1.1.6 Start the Kerberos Daemons on the Master KDC

At this point, you are ready to start the Kerberos daemons on the Master KDC. To do so, type:

```
shell% /usr/local/sbin/krb5kdc
shell% /usr/local/sbin/kadmind
```

Each daemon will fork and run in the background. Assuming you want these daemons to start up automatically at boot time, you can add them to the KDC's `/etc/rc` or `/etc/inittab` file. You need to have a stash file in order to do this.

You can verify that they started properly by checking for their startup messages in the logging locations you defined in `/etc/krb5.conf`. (See See Section 4.1.1.1 [Edit the Configuration Files], page 17.) For example:

```
shell% tail /var/log/krb5kdc.log
Dec 02 12:35:47 beeblebrox krb5kdc[3187](info): commencing operation
shell% tail /var/log/kadmin.log
Dec 02 12:35:52 beeblebrox kadmind[3189](info): starting
```

Any errors the daemons encounter while starting will also be listed in the logging output.

4.1.2 Install the Slave KDCs

You are now ready to start configuring the slave KDCs. Assuming you are setting the KDCs up so that you can easily switch the master KDC with one of the slaves, you should perform each of these steps on the master KDC as well as the slave KDCs, unless these instructions specify otherwise.

4.1.2.1 Create Host Keys for the Slave KDCs

Each KDC needs a host principal in the Kerberos database. You can enter these from any host, once the `kadmind` daemon is running. For example, if your master KDC were called kerberos.mit.edu, and you had two KDC slaves named kerberos-1.mit.edu and kerberos-2.mit.edu, you would type the following:

```
shell% /usr/local/sbin/kadmin
kadmin: addprinc -randkey host/kerberos.mit.edu
WARNING: no policy specified for "host/kerberos.mit.edu@ATHENA.MIT.EDU";
defaulting to no policy.
Principal "host/kerberos.mit.edu@ATHENA.MIT.EDU" created.
kadmin: addprinc -randkey host/kerberos-1.mit.edu
WARNING: no policy specified for "host/kerberos-1.mit.edu@ATHENA.MIT.EDU";
defaulting to no policy.
Principal "host/kerberos-1.mit.edu@ATHENA.MIT.EDU" created.
kadmin: addprinc -randkey host/kerberos-2.mit.edu
WARNING: no policy specified for "host/kerberos-2.mit.edu@ATHENA.MIT.EDU";
defaulting to no policy.
Principal "host/kerberos-2.mit.edu@ATHENA.MIT.EDU" created.
kadmin:
```

It is not actually necessary to have the master KDC server in the Kerberos database, but it can be handy if:

- anyone will be logging into the machine as something other than root
- you want to be able to swap the master KDC with one of the slaves if necessary.

4.1.2.2 Extract Host Keytabs for the KDCs

Each KDC (including the master) needs a keytab to decrypt tickets. Ideally, you should extract each keytab locally on its own KDC. If this is not feasible, you should use an encrypted session to send them across the network. To extract a keytab on a KDC called kerberos.mit.edu, you would execute the following command:

```
kadmin: ktadd host/kerberos.mit.edu
kadmin: Entry for principal host/kerberos.mit.edu@ATHENA.MIT.EDU with
    kvno 1, encryption type DES-CBC-CRC added to keytab
    WRFILE:/etc/krb5.keytab.
kadmin:
```

Note that the principal must exist in the Kerberos database in order to extract the keytab.

4.1.2.3 Set Up the Slave KDCs for Database Propagation

The database is propagated from the master KDC to the slave KDCs via the `kpropd` daemon. To set up propagation, create a file on each KDC, named `/usr/local/var/krb5kdc/kpropd.acl`, containing the principals for each of the KDCs.

For example, if the master KDC were `kerberos.mit.edu`, the slave KDCs were `kerberos-1.mit.edu` and `kerberos-2.mit.edu`, and the realm were `ATHENA.MIT.EDU`, then the file's contents would be:

```
host/kerberos.mit.edu@ATHENA.MIT.EDU
host/kerberos-1.mit.edu@ATHENA.MIT.EDU
host/kerberos-2.mit.edu@ATHENA.MIT.EDU
```

Then, add the following lines to `/etc/inetd.conf` file on each KDC (the line beginnng with ⇒ is a continuation of the previous line):

```
krb5_prop stream tcp nowait root /usr/local/sbin/kpropd kpropd
eklogin   stream tcp nowait root /usr/local/sbin/klogind
⇒ klogind -k -c -e
```

The first line sets up the `kpropd` database propagation daemon. The second line sets up the `eklogin` daemon, allowing Kerberos-authenticated, encrypted rlogin to the KDC.

You also need to add the following lines to `/etc/services` on each KDC:

```
kerberos        88/udp      kdc        # Kerberos authentication (udp)
kerberos        88/tcp      kdc        # Kerberos authentication (tcp)
krb5_prop       754/tcp                # Kerberos slave propagation
kerberos-adm    749/tcp                # Kerberos 5 admin/changepw (tcp)
kerberos-adm    749/udp                # Kerberos 5 admin/changepw (udp)
eklogin         2105/tcp               # Kerberos encrypted rlogin
```

4.1.3 Back on the Master KDC

Now that the slave KDCs are able to accept database propagation, you'll need to propagate the database to each of them.

4.1.3.1 Propagate the Database to Each Slave KDC

First, create a dump of the database on the master KDC, as follows:

```
shell% /usr/local/sbin/kdb5_util dump /usr/local/var/krb5kdc/slave_datatrans
shell%
```

Next, you need to manually propagate the database to each slave KDC, as in the following example. (The lines beginning with ⇒ are continuations of the previous line.):

```
/usr/local/sbin/kprop -f /usr/local/var/krb5kdc/slave_datatrans
⇒ kerberos-1.mit.edu
/usr/local/sbin/kprop -f /usr/local/var/krb5kdc/slave_datatrans
⇒ kerberos-2.mit.edu
```

You will need a script to dump and propagate the database. The following is an example of a bourne shell script that will do this. (Note that the line that begins with ⇒ is a continuation of the previous line. Remember that you need to replace /usr/local with the name of the directory in which you installed Kerberos V5.)

```
#!/bin/sh

kdclist = "kerberos-1.mit.edu kerberos-2.mit.edu"

/usr/local/sbin/kdb5_util -R "dump
⇒ /usr/local/var/krb5kdc/slave_datatrans"

for kdc in $kdclist
do
/usr/local/sbin/kprop -f /usr/local/var/krb5kdc/slave_datatrans $kdc
done
```

You will need to set up a cron job to run this script at the intervals you decided on earlier (See Section 2.6 [Database Propagation], page 6.)

4.1.4 Finish Installing the Slave KDCs

Now that the slave KDCs have copies of the Kerberos database, you can create stash files for them and start the `krb5kdc` daemon.

4.1.4.1 Create Stash Files on the Slave KDCs

Create stash files, by issuing the following commands on each slave KDC:

```
shell% kdb5_util stash
kdb5_util: Cannot find/read stored master key while reading master key
kdb5_util: Warning: proceeding without master key

Enter KDC database master key:    ⇐ Enter the database master key.

shell%
```

As mentioned above, the stash file is necessary for your KDCs to be able authenticate to themselves, such as when they reboot. You could run your KDCs without stash files, but you would then need to type in the Kerberos database master key by hand every time you start a KDC daemon.

4.1.4.2 Start the krb5kdc Daemon on Each KDC

The final step in configuing your slave KDCs is to run the KDC daemon:

```
shell% /usr/local/sbin/krb5kdc
```

As with the master KDC, you will probably want to add this command to the KDCs' `/etc/rc` or `/etc/inittab` files, so they will start the krb5kdc daemon automatically at boot time.

4.1.5 Add Kerberos Principals to the Database

Once your KDCs are set up and running, you are ready to use `kadmin` to load principals for your users, hosts, and other services into the Kerberos database. This procedure is described fully in the "Adding or Modifying Principals" section of the Kerberos V5 System Administrator's Guide. (See Section 4.1.2.1 [Create Host Keys for the Slave KDCs], page 21, for a brief description.) The keytab is generated by running `kadmin` and issuing the `ktadd` command.

4.1.6 Limit Access to the KDCs

To limit the possibility that your Kerberos database could be compromised, MIT recommends that each KDC be a dedicated host, with limited access. If your KDC is also a file server, FTP server, Web server, or even just a client machine, someone who obtained root access through a security hole in any of those areas could gain access to the Kerberos database.

MIT recommends that your KDCs use the following `/etc/inetd.conf` file. (Note: each line beginning with ⇒ is a continuation of the previous line.):

```
#
# Configuration file for inetd(1M).  See inetd.conf(4).
#
# To re-configure the running inetd process, edit this file, then
# send the inetd process a SIGHUP.
#
# Syntax for socket-based Internet services:
#  <service_name> <socket_type> <proto> <flags> <user>
⇒ <server_pathname> <args>
#
# Syntax for TLI-based Internet services:
#
#  <service_name> tli <proto> <flags> <user> <server_pathname> <args>
#
# Ftp and telnet are standard Internet services.
#
# This machine is a secure Kerberos Key Distribution Center (KDC).
# Services are limited.
#
#
# Time service is used for clock synchronization.
#
time    stream  tcp     nowait  root    internal
time    dgram   udp     wait    root    internal
#
# Limited Kerberos services
#
krb5_prop stream tcp nowait root /usr/local/sbin/kpropd  kpropd
eklogin   stream tcp nowait root /usr/local/sbin/klogind
⇒ klogind -5 -c -e
```

4.1.7 Switching Master and Slave KDCs

You may occasionally want to use one of your slave KDCs as the master. This might happen if you are upgrading the master KDC, or if your master KDC has a disk crash.

Assuming you have configured all of your KDCs to be able to function as either the master KDC or a slave KDC (as this document recommends), all you need to do to make the changeover is:

If the master KDC is still running, do the following on the *old* master KDC:

1. Kill the `kadmind` process.
2. Disable the cron job that propagates the database.
3. Run your database propagation script manually, to ensure that the slaves all have the latest copy of the database. (See Section 4.1.3.1 [Propagate the Database to Each Slave KDC], page 23.) As of the 1.2.2 release, it is no longer necessary to use "kdb5_util dump -ov" in order to preserve per-principal policy information, as the default dump format now supports it. Note you should update your slaves prior to your master, so that they will understand the new dump format. (This is a good policy anyway.)

On the *new* master KDC:

1. Create a database keytab. (See Section 4.1.1.5 [Create a kadmind Keytab], page 20.)
2. Start the `kadmind` daemon. (See Section 4.1.1.6 [Start the Kerberos Daemons], page 21.)
3. Set up the cron job to propagate the database. (See Section 4.1.3.1 [Propagate the Database to Each Slave KDC], page 23.)
4. Switch the CNAMEs of the old and new master KDCs. (If you don't do this, you'll need to change the `krb5.conf` file on every client machine in your Kerberos realm.)

4.2 Installing and Configuring UNIX Client Machines

Client machine installation is much more straightforward than installation of the KDCs.

4.2.1 Client Programs

The Kerberized client programs are `login.krb5`, `rlogin`, `telnet`, `ftp`, `rcp`, `rsh`, `kinit`, `klist`, `kdestroy`, `kpasswd`, `ksu`, and `krb524init`. All of these programs are in the directory `/usr/local/bin`, except for `login.krb5` which is in `/usr/local/sbin`.

You will probably want to have your users put `/usr/local/bin` ahead of `/bin` and `/usr/bin` in their paths, so they will by default get the Kerberos V5 versions of `rlogin`, `telnet`, `ftp`, `rcp`, and `rsh`.

MIT recommends that you use `login.krb5` in place of `/bin/login` to give your users a single-sign-on system. You will need to make sure your users know to use their Kerberos passwords when they log in.

You will also need to educate your users to use the ticket management programs `kinit`, `klist`, `kdestroy`, and to use the Kerberos programs `ksu`, and `kpasswd` in place of their non-Kerberos counterparts `su`, `passwd`, and `rdist`.

4.2.2 Client Machine Configuration Files

Each machine running Kerberos must have a `/etc/krb5.conf` file. (See Section A.1 [krb5.conf], page 39)

Also, for most UNIX systems, you must add the appropriate Kerberos services to each client machine's `/etc/services` file. If you are using the default configuration for Kerberos V5, you should be able to just insert the following code:

```
#
# Note --- if you are using Kerberos V4 and you either:
#
#    (a) haven't converted all your master or slave KDCs to V5, or
#
#    (b) are worried about inter-realm interoperability with other KDC's
#        that are still using V4
#
# you will need to switch the "kerberos" service to port 750 and create a
# "kerberos-sec" service on port 88.
#
kerberos      88/udp    kdc    # Kerberos V5 KDC
kerberos      88/tcp    kdc    # Kerberos V5 KDC
klogin        543/tcp          # Kerberos authenticated rlogin
kshell        544/tcp   cmd    # and remote shell
kerberos-adm  749/tcp          # Kerberos 5 admin/changepw
kerberos-adm  749/udp          # Kerberos 5 admin/changepw
krb5_prop     754/tcp          # Kerberos slave propagation
eklogin       2105/tcp         # Kerberos auth. & encrypted rlogin
krb524        4444/tcp         # Kerberos 5 to 4 ticket translator
```

As described in the comments in the above code, if your master KDC or any of your slave KDCs is running Kerberos V4, (or if you will be authenticating to any Kerberos V4 KDCs in another realm) you will need to switch the port number for `kerberos` to 750 and create a `kerberos-sec` service (tcp and udp) on port 88, so the Kerberos V4 KDC(s) will continue to work properly.

4.2.2.1 Mac OS X Configuration

To install Kerberos V5 on Mac OS X and Mac OS X Server, follow the directions for generic Unix-based OS's, except for the `/etc/services` updates described above.

Mac OS X and Mac OS X Server use a database called NetInfo to store the contents of files normally found in `/etc`. Instead of modifying `/etc/services`, you should run the following commands to add the Kerberos service entries to NetInfo:

```
$ niutil -create . /services/kerberos
$ niutil -createprop . /services/kerberos name kerberos kdc
$ niutil -createprop . /services/kerberos port 750
$ niutil -createprop . /services/kerberos protocol tcp udp
$ niutil -create . /services/krbupdate
$ niutil -createprop . /services/krbupdate name krbupdate kreg
$ niutil -createprop . /services/krbupdate port 760
$ niutil -createprop . /services/krbupdate protocol tcp
$ niutil -create . /services/kpasswd
$ niutil -createprop . /services/kpasswd name kpasswd kpwd
$ niutil -createprop . /services/kpasswd port 761
$ niutil -createprop . /services/kpasswd protocol tcp
$ niutil -create . /services/klogin
$ niutil -createprop . /services/klogin port 543
$ niutil -createprop . /services/klogin protocol tcp
$ niutil -create . /services/eklogin
$ niutil -createprop . /services/eklogin port 2105
$ niutil -createprop . /services/eklogin protocol tcp
$ niutil -create . /services/kshell
$ niutil -createprop . /services/kshell name kshell krcmd
$ niutil -createprop . /services/kshell port 544
$ niutil -createprop . /services/kshell protocol tcp
```

In addition to adding services to NetInfo, you must also modify the resolver configuration in NetInfo so that the machine resolves its own hostname as a FQDN (fully qualified domain name). By default, Mac OS X and Mac OS X Server machines query NetInfo to resolve hostnames before falling back to DNS. Because NetInfo has an unqualified name for all the machines in the NetInfo database, the machine's own hostname will resolve to an unqualified name. Kerberos needs a FQDN to look up keys in the machine's keytab file.

Fortunately, you can change the `lookupd` caching order to query DNS first. Run the following NetInfo commands and reboot the machine:

```
$ niutil -create . /locations/lookupd/hosts
$ niutil -createprop . /locations/lookupd/hosts LookupOrder CacheAgent DNSAgent
 NIAgent NILAgent
```

Once you have rebooted, you can verify that the resolver now behaves correctly. Compile the Kerberos 5 distribution and run:

```
$ cd .../src/tests/resolve
$ ./resolve
```

This will tell you whether or not your machine returns FQDNs on name lookups. If the test still fails, you can also try turning off DNS caching. Run the following commands and reboot:

```
$ niutil -create . /locations/lookupd/hosts
$ niutil -createprop . /locations/lookupd/hosts LookupOrder DNSAgent
 CacheAgent NIAgent NILAgent
```

The remainder of the setup of a Mac OS X client machine or application server should be the same as for other UNIX-based systems.

4.3 UNIX Application Servers

An application server is a host that provides one or more services over the network. Application servers can be "secure" or "insecure." A "secure" host is set up to require authentication from every client connecting to it. An "insecure" host will still provide Kerberos authentication, but will also allow unauthenticated clients to connect.

If you have Kerberos V5 installed on all of your client machines, MIT recommends that you make your hosts secure, to take advantage of the security that Kerberos authentication affords. However, if you have some clients that do not have Kerberos V5 installed, you can run an insecure server, and still take advantage of Kerberos V5's single sign-on on capability.

4.3.1 Server Programs

Just as Kerberos V5 provided its own Kerberos-enhanced versions of client UNIX network programs, Kerberos V5 also provides Kerberos-enhanced versions of server UNIX network daemons. These are `ftpd`, `klogind`, `kshd`, and `telnetd`. These programs are installed in the directory `/usr/local/sbin`. You may want to add this directory to root's path.

4.3.2 Server Configuration Files

For a *secure* server, make the following changes to `/etc/inetd.conf`:

Find and comment out any lines for the services `ftp`, `telnet`, `shell`, `login`, and `exec`.

Add the following lines. (Note: each line beginning with ⇒ is a continuation of the previous line.)

```
klogin  stream  tcp  nowait  root  /usr/local/sbin/klogind
⇒ klogind -k -c
eklogin stream  tcp  nowait  root  /usr/local/sbin/klogind
⇒ klogind -k -c -e
kshell  stream  tcp  nowait  root  /usr/local/sbin/kshd
⇒ kshd -k -c -A
ftp     stream  tcp  nowait  root  /usr/local/sbin/ftpd
⇒ ftpd -a
telnet  stream  tcp  nowait  root  /usr/local/sbin/telnetd
⇒ telnetd -a valid
```

For an *insecure* server, make the following changes instead to `/etc/inetd.conf`:

Find and comment out any lines for the services `ftp` and `telnet`.

Add the following lines. (Note: each line beginning with ⇒ is a continuation of the previous line.)

```
klogin  stream  tcp  nowait  root  /usr/local/sbin/klogind
⇒ klogind -k -c
eklogin stream  tcp  nowait  root  /usr/local/sbin/klogind
⇒ klogind -k -c -e
kshell  stream  tcp  nowait  root  /usr/local/sbin/kshd
⇒ kshd -k -c -A
ftp     stream  tcp  nowait  root  /usr/local/sbin/ftpd
⇒ ftpd
telnet  stream  tcp  nowait  root  /usr/local/sbin/telnetd
⇒ telnetd -a none
```

4.3.3 The Keytab File

All Kerberos server machines need a *keytab* file, called `/etc/krb5.keytab`, to authenticate to the KDC. The keytab file is an encrypted, local, on-disk copy of the host's key. The keytab file, like the stash file (Section 4.1.1.2 [Create the Database], page 18) is a potential point-of-entry for a break-in, and if compromised, would allow unrestricted access to its host. The keytab file should be readable only by root, and should exist only on the machine's local disk. The file should not be part of any backup of the machine, unless access to the backup data is secured as tightly as access to the machine's root password itself.

In order to generate a keytab for a host, the host must have a principal in the Kerberos database. The procedure for adding hosts to the database is described fully in the "Adding or Modifying Principals" section of the *Kerberos V5 System Administrator's Guide*. See Section 4.1.2.1 [Create Host Keys for the Slave KDCs], page 21 for a brief description.) The keytab is generated by running `kadmin` and issuing the `ktadd` command.

For example, to generate a keytab file to allow the host trillium.mit.edu to authenticate for the services `host`, `ftp`, and `pop`, the administrator `joeadmin` would issue the command (on trillium.mit.edu):

```
trillium% /usr/local/sbin/kadmin
kadmin5: ktadd host/trillium.mit.edu ftp/trillium.mit.edu
⇒ pop/trillium.mit.edu
kadmin: Entry for principal host/trillium.mit.edu@ATHENA.MIT.EDU with
kvno 3, encryption type DES-CBC-CRC added to keytab
WRFILE:/etc/krb5.keytab.
kadmin: Entry for principal ftp/trillium.mit.edu@ATHENA.MIT.EDU with
kvno 3, encryption type DES-CBC-CRC added to keytab
WRFILE:/etc/krb5.keytab.
kadmin: Entry for principal pop/trillium.mit.edu@ATHENA.MIT.EDU with
kvno 3, encryption type DES-CBC-CRC added to keytab
WRFILE:/etc/krb5.keytab.
kadmin5: quit
trillium%
```

If you generate the keytab file on another host, you need to get a copy of the keytab file onto the destination host (`trillium`, in the above example) without sending it unencrypted over the network. If you have installed the Kerberos V5 client programs, you can use encrypted `rcp`.

4.3.4 Some Advice about Secure Hosts

Kerberos V5 can protect your host from certain types of break-ins, but it is possible to install Kerberos V5 and still leave your host vulnerable to attack. Obviously an installation guide is not the place to try to include an exhaustive list of countermeasures for every possible attack, but it is worth noting some of the larger holes and how to close them.

As stated earlier in this section, MIT recommends that on a secure host, you disable the standard `ftp`, `login`, `telnet`, `shell`, and `exec` services in `/etc/inetd.conf`. We also recommend that secure hosts have an empty `/etc/hosts.equiv` file and that there not be a `.rhosts` file in `root`'s home directory. You can grant Kerberos-authenticated root access to specific Kerberos principals by placing those principals in the file `.k5login` in root's home directory.

We recommend that backups of secure machines exclude the keytab file (`/etc/krb5.keytab`). If this is not possible, the backups should at least be done locally, rather than over a network, and the backup tapes should be physically secured.

Finally, the keytab file and any programs run by root, including the Kerberos V5 binaries, should be kept on local disk. The keytab file should be readable only by root.

5 Upgrading Existing Kerberos V5 Installations

If you already have an existing Kerberos database that you created with a prior release of Kerberos 5, you can upgrade it to work with the current release with the `kdb5_util` command. It is only necessary to perform this dump/undump procedure if you were running a krb5-1.0.x KDC and are migrating to a krb5-1.1.x or newer KDC. The process for upgrading a Master KDC involves the following steps:

1. Stop your current KDC and administration server processes, if any.
2. Dump your existing Kerberos database to an ASCII file with `kdb5_util`'s "dump" command:
   ```
   shell% cd /usr/local/var/krb5kdc
   shell% kdb5_util dump old-kdb-dump
   shell% kdb5_util dump -ov old-kdb-dump.ov
   shell%
   ```
3. Create a new Master KDC installation (See Section 4.1.1 [Install the Master KDC], page 17). If you have a stash file for your current database, choose any new master password but then copy your existing stash file to the location specified by your kdc.conf; if you do not have a stash file for your current database, you must choose the same master password.
4. Load your old Kerberos database into the new system with `kdb5_util`'s "load" command:
   ```
   shell% cd /usr/local/var/krb5kdc
   shell% kdb5_util load old-kdb-dump
   shell% kdb5_util load -update old-kdb-dump.ov
   shell%
   ```

The "dump -ov" and "load -update" commands are necessary in order to preserve per-principal policy information, since the dump format in releases prior to 1.2.2 filters out that information. If you omit those steps, the loaded database database will lose the policy information for each principal that has a policy.

To update a Slave KDC, you must stop the old server processes on the Slave KDC, install the new server binaries, reload the most recent slave dump file, and re-start the server processes.

5.1 Upgrading to Triple-DES Encryption Keys

Beginning with the 1.2 release from MIT, Kerberos includes a stronger encryption algorithm called "triple DES" – essentially, three applications of the basic DES encryption algorithm, greatly increasing the resistance to a brute-force search for the key by an attacker. This algorithm is more secure, but encryption is much slower. We expect to add other, faster encryption algorithms at some point in the future.

Release 1.1 had some support for triple-DES service keys, but with release 1.2 we have added support for user keys and session keys as well. Release 1.0 had very little support for multiple cryptosystems, and some of that software may not function properly in an environment using triple-DES as well as plain DES.

Because of the way the MIT Kerberos database is structured, the KDC will assume that a service supports only those encryption types for which keys are found in the database. Thus, if a service has only a single-DES key in the database, the KDC will not issue tickets for that service that use triple-DES session keys; it will instead issue only single-DES session keys, even if other services are already capable of using triple-DES. So if you make sure your application server software is updated before adding a triple-DES key for the service, clients should be able to talk to services at all times during the updating process.

Normally, the listed `supported_enctypes` in `kdc.conf` are all used when a new key is generated. You can control this with command-line flags to `kadmin` and `kadmin.local`. You may want to exclude triple-DES by default until you have updated a lot of your application servers, and then change the default to include triple-DES. We recommend that you always include `des-cbc-crc` in the default list.

6 Bug Reports for Kerberos V5

In any complex software, there will be bugs. If you have successfully built and installed Kerberos V5, please use the `krb5-send-pr` program to fill out a Problem Report.

Bug reports that include proposed fixes are especially welcome. If you do include fixes, please send them using either context diffs or unified diffs (using '`diff -c`' or '`diff -u`', respectively). Please be careful when using "cut and paste" or other such means to copy a patch into a bug report; depending on the system being used, that can result in converting TAB characters into spaces, which makes applying the patches more difficult.

The `krb5-send-pr` program is installed in the directory `/usr/local/sbin`.

The `krb5-send-pr` program enters the problem report into our Problem Report Management System (PRMS), which automatically assigns it to the engineer best able to help you with problems in the assigned category.

The `krb5-send-pr` program will try to intelligently fill in as many fields as it can. You need to choose the *category*, *class*, *severity*, and *priority* of the problem, as well as giving us as much information as you can about its exact nature.

The PR **category** will be one of:

```
krb5-admin   krb5-appl    krb5-build   krb5-clients
krb5-doc     krb5-kdc     krb5-libs    krb5-misc
pty          telnet       test
```

Choose the category that best describes the area under which your problem falls.

The **class** can be *sw-bug*, *doc-bug*, *change-request*, or *support*. The first two are exactly as their names imply. Use *change-request* when the software is behaving according to specifications, but you want to request changes in some feature or behavior. The *support* class is intended for more general questions about building or using Kerberos V5.

The **severity** of the problem indicates the problem's impact on the usability of Kerberos V5. If a problem is *critical*, that means the product, component or concept is completely non-operational, or some essential functionality is missing, and no workaround is known. A *serious* problem is one in which the product, component or concept is not working properly or significant functionality is missing. Problems that would otherwise be considered *critical* are rated *serious* when a workaround is known. A *non-critical* problem is one that is indeed a problem, but one that is having a minimal effect on your ability to use Kerberos V5. *E.g.*, The product, component or concept is working in general, but lacks features, has irritating behavior, does something wrong, or doesn't match its documentation. The default severity is *serious*.

The **priority** indicates how urgent this particular problem is in relation to your work. Note that low priority does not imply low importance. A priority of *high* means a solution is needed as soon as possible. A priority of *medium* means the problem should be solved no later than the next release. A priority of *low* means the problem should be solved in a future release, but it is not important to your work how soon this happens. The default priority is *medium*.

Note that a given severity does not necessarily imply a given priority. For example, a non-critical problem might still have a high priority if you are faced with a hard deadline. Conversely, a serious problem might have a low priority if the feature it is disabling is one that you do not need.

It is important that you fill in the *release* field and tell us what changes you have made, if any.

Bug reports that include proposed fixes are especially welcome. If you include proposed fixes, please send them using either context diffs (‘`diff -c`’) or unified diffs (‘`diff -u`’).

A sample filled-out form from a company named “Toasters, Inc.” might look like this:

```
To: krb5-bugs@mit.edu
Subject: misspelled "Kerberos" in title of installation guide
From: jcb
Reply-To: jcb
Cc:
X-send-pr-version: 3.99


>Submitter-Id:  mit
>Originator:  Jeffrey C. Gilman Bigler
>Organization:
mit
>Confidential:  no
>Synopsis:  Misspelled "Kerberos" in title of installation guide
>Severity:  non-critical
>Priority:  low
>Category:  krb5-doc
>Class:  doc-bug
>Release:  1.0-development
>Environment:
<machine, os, target, libraries (multiple lines)>
System: ULTRIX imbrium 4.2 0 RISC
Machine: mips
>Description:
        Misspelled "Kerberos" in title of "Kerboros V5 Installation Guide"
>How-To-Repeat:
        N/A
>Fix:
        Correct the spelling.
```

If the `krb5-send-pr` program does not work for you, or if you did not get far enough in the process to have an installed and working `krb5-send-pr`, you can generate your own form, using the above as an example.

Appendix A Files

A.1 krb5.conf

Here is an example `krb5.conf` file:

```
[libdefaults]
    ticket_lifetime = 600
    default_realm = ATHENA.MIT.EDU
    default_tkt_enctypes = des3-hmac-sha1 des-cbc-crc
    default_tgs_enctypes = des3-hmac-sha1 des-cbc-crc

[realms]
    ATHENA.MIT.EDU = {
        kdc = kerberos.mit.edu:88
        kdc = kerberos-1.mit.edu:88
        kdc = kerberos-2.mit.edu:88
        admin_server = kerberos.mit.edu:749
        default_domain = mit.edu
    }

[domain_realm]
    .mit.edu = ATHENA.MIT.EDU
    mit.edu = ATHENA.MIT.EDU
```

For the KDCs, add a section onto the end of the `krb5.conf` file telling how logging information should be stored, as in the following example:

```
[logging]
    kdc = FILE:/var/log/krb5kdc.log
    admin_server = FILE:/var/log/kadmin.log
    default = FILE:/var/log/krb5lib.log
```

A.2 kdc.conf

Here's an example of a kdc.conf file:

```
[kdcdefaults]
    kdc_ports = 88,750

[realms]
    ATHENA.MIT.EDU = {
        database_name = /usr/local/var/krb5kdc/principal
        admin_keytab = /usr/local/var/krb5kdc/kadm5.keytab
        acl_file = /usr/local/var/krb5kdc/kadm5.acl
        dict_file = /usr/local/var/krb5kdc/kadm5.dict
        key_stash_file = /usr/local/var/krb5kdc/.k5.ATHENA.MIT.EDU
        kadmind_port = 749
        max_life = 10h 0m 0s
        max_renewable_life = 7d 0h 0m 0s
        master_key_type = des3-hmac-sha1
        supported_enctypes = des3-hmac-sha1:normal des-cbc-crc:normal
    }
```

To add Kerberos V4 support, add `des-cbc-crc:v4` to the `supported_enctypes` line.

A.2.1 Encryption Types and Salt Types

Currently, Kerberos V5 supports only DES and triple-DES encryption. The encoding types include `des-cbc-crc` and `des3-cbc-sha1`. The *salt* is additional information encoded within the key that tells what kind of key it is. The only salts that you will be likely to encounter are:

- *normal*, which MIT recommends using for all of your Kerberos V5 keys
- *v4*, which is necessary only for compatibility with a v4 KDC or a v4 version of `kinit`, and then only with `des-cbc-crc` encryption
- *afs*, which you will never need to generate, and which you will encounter only if you dump an AFS database into a Kerberos database

Support for additional encryption types is planned in the future.