Kerberos V5

Notes on Building and Installing Kerberos
Edition BETA 6, for Kerberos version BETA 6
Mar 20, 1996

by Theodore Ts'o

1 Introduction

This document describes the procedures necessary to compile Kerberos V5 from the source distribution, and then to install it at a particular site. The reader is assumed to be familiar with C/UNIX development tools.

In any complex software, there will be bugs. In addition, you may find portability problems when you try to build Kerberos V5 on a platform on which the developers have not used. Please send bug reports or other problems you may uncover to the e-mail address **krb5-bugs@mit.edu**. Please mention which version of the Kerberos V5 distribution you are using, and whether you have made any private changes. 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 note that there are still a number of aspects of Kerberos V5 which will likely change before the 1.0 release. As these changes occur, we will update the documentation accordingly.

2 How Kerberos Works: A Schematic Description

This section provides a simplified description of a general user's interaction with the Kerberos system. This interaction happens transparently–users don't need to know and probably don't care about what's going on–but Kerberos administrators might find a schematic description of the process useful. This description glosses over a lot of details; for more information, see *Kerberos: An Authentication Service for Open Network Systems*, a paper presented at Winter USENIX 1988, in Dallas, Texas.

2.1 Network Services and Their Client Programs

In an environment that provides network services, you use *client* programs to request service from *server* programs that are somewhere on the network. Suppose you have logged in to a workstation and you want to '`rlogin`' to another machine. You use the local '`rlogin`' client program to contact the remote machine's '`rlogind`' daemon.

2.2 Kerberos Tickets

Under Kerberos, the '`rlogind`' daemon allows you to login to a remote machine if you can provide '`rlogind`' a Kerberos ticket which proves your identity. In addition to the ticket, you must also have possession of the corresponding ticket session key. The combination of a ticket and the ticket's session key is known as a credential.

Typically, a client program automatically obtains credentials identifying the person using the client program. The credentials are obtained from a Kerberos server that resides somewhere on the network. A Kerberos server maintains a database of user, server, and password information.

2.3 The Kerberos Database

Kerberos will give you credentials only if you have an entry in the Kerberos server's *Kerberos database*. Your database entry includes your Kerberos *principal* (which is often just your username), and your Kerberos password. Every Kerberos user must have an entry in this database.

2.4 Kerberos Realms

Each site (or administrative domain within a site) will have their own Kerberos database, which contains information about the users and services for that particular site or administrative domain. A *Kerberos Realm* is used to distinguish the users and services in one particular area of administrative control from another area of administrative control.

Each Kerberos realm will have at least one Kerberos server, where the master Kerberos database for that site or administrative domain is stored. A Kerberos realm may also have one or more *slave servers*, which have read-only copies of the Kerberos database which are periodically propagated from the master server. For more details on how this is done, see Section 4.3.6 [Setting up Slave Kerberos Servers], page 26.

2.5 The Ticket-Granting Ticket

The '`kinit`' command prompts for your password, and if you enter it successfully, you will obtain a *ticket-granting ticket* and a *ticket session key* which gives you the right to use the ticket. This combination of the ticket and its associated key is also known as your *credentials*. As illustrated below, client programs use your ticket-granting ticket credentials in order to obtain client-specific credentials as needed.

Your credentials are stored in a *credentials cache*, which is often simply just a file in '`/tmp`'. The credentials cache is also referred to as the *ticket file*, especially in Kerberos V4 documentation. It should be noted, however, that a credentials cache does not have to be stored in a file.

2.6 Network Services and the Master Database

The master database also contains entries for all network services that require Kerberos authentication. Suppose that your site has a machine, '`laughter.mit.edu`', that requires Kerberos authentication from anyone who wants to '`rlogin`' to it. The host's Kerberos realm is '`ATHENA.MIT.EDU`'.

This service must be registered in the Kerberos database, using the proper service name, which in this case is '`host/laughter.mit.edu@ATHENA.MIT.EDU`'. The `/` character separates the various *components* of the Kerberos principal; the `@` character separates the realm name from the rest of the principal name. The first component, '`host`', denotes the name or type of the service that is being offered: generic host-level access to the machine. The second component, '`laughter.mit.edu`', names the specific machine which is offering this service. There will generally be many different machines, each offering one particular type of service, and the second component serves to give each one of these servers a different Kerberos name.

2.6.1 The Keytab File

For each service, there must also be a *service key* known only by Kerberos and the service. On the Kerberos server, the service key is stored in the Kerberos database.

On the server host, these service keys are stored in the *Key tables*, or *keytab files*. (Keytab files were previously referred to as *srvtab files* in the V4 literature.) The service keys that are used by services which run as root are often stored in the file '`/etc/v5srvtab`'.

WARNING: This service key is the equivalent of the service's password, and must be kept secure. Data which is meant to be read only by the service is encrypted using this key.

2.7 The User–Kerberos Interaction

Suppose that you (in the guise of a general user) walk up to a workstation intending to login to it, and then '`rlogin`' to the machine '`laughter`'. Here's what happens:

1. You login to the workstation and use the '`kinit`' command to get a ticket-granting ticket. This command prompts you for your Kerberos password. (On some systems which have a modified '`/bin/login`' program, this may be done as part of the login process, not requiring the user to run a separate program.)
   A. The '`kinit`' command sends your request to the Kerberos master server machine. The server software looks for your principal name's entry in the Kerberos database.
   B. If this entry exists, the Kerberos server creates and returns a ticket-granting ticket and the key which allows you to use it, encrypted by your password. If '`kinit`' can decrypt the Kerberos reply using the password you provide, it stores this ticket in a credentials cache on your local machine for later use. The name of the credentials cache can be specified in the '`KRB5_CCNAME`' environment variable. If this variable is not set, the name of the file will be '`/tmp/krb5cc_<uid>`', where `<uid>` is your UNIX user-id, represented in decimal format.
2. Now you use the '`rlogin`' client to access the machine '`laughter`'.

   `host%` **rlogin laughter**

A. The '`rlogin`' client checks your ticket file to see if you have a ticket for the '`host`' service for '`laughter`'. You don't, so '`rlogin`' uses the credential cache's ticket-granting ticket to make a request to the master server's ticket-granting service.

B. This ticket-granting service receives the '`host/laughter.mit.edu`' request and looks in the master database for an '`host/laughter.mit.edu`' entry. If the entry exists, the ticket-granting service issues you a ticket for that service. That ticket is also cached in your credentials cache.

C. The '`rlogin`' client now sends that ticket to the '`laughter`' '`rlogind`' service program. The service program checks the ticket by using its own service key. If the ticket is valid, it now knows your identity. If the ticket is valid and you are allowed to login to '`laughter`' (because the your name matches one in /etc/passwd, or you are in the appropriate '`.klogin`' or '`.k5login`' file), will let you login.

3 Building Kerberos

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.src.tar.gz', contains the sources for all of Kerberos except for the crypto library, which is found in the file 'krb5.crypto.tar.gz'.

Both files should be unpacked in the same directory, such as '/u1/krb5'. (In the rest of this document, we will assume that you have chosen to unpack the Kerberos V5 source distribution in this directory.)

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 see Section 3.8 [OS Incompatibilities], page 15 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/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 gmake 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/pmax`
2. `cd /u1/krb5/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 cannot use 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/solaris`
2. `cd /u1/krb5/solaris`
3. `/u1/krb5/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.

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 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. 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 other authenticated means.[1]

---

[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`’ as appropriate to allow you access.

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.

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.

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

`--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. 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 3.8.5 [Solaris versions 2.0 through 2.3], page 16) or fails to pass the tests in '`src/tests/resolv`' you will need to use this option.

`--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-athena`
: Was used to setup configuration files in such a way as to be compatible with the Athena environment at MIT. It currently does not appear to do anything useful.

`--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.

For example, in order to configure Kerberos on a Solaris machine using the '`suncc`' 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_LNAME_FILENAME
: The pathname to the database that maps authentication names to local account names. See kdb5_anadd(8).

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 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: NetBSD 1.0A, AIX 3.2.5, AIX 4.1, Solaris 5.3, Alpha OSF/1 >= 2.1, HP-UX >= 9.X.

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

One special note is that if the Kerberos V4 compatibility is compiled in, you **must not** specify an alternate Kerberos V4 library from the one in the tree or you will be missing references.

3.7.1 Theory of How Shared Libraries are Used

An explanation of how shared libraries are implemented on a given platform is too broad a topic for this manual. Instead this will touch on some of the issues that the Kerberos V5 tree uses to support version numbering and alternate install locations.

Normally when one builds a shared library and then links with it, the name of the shared library is stored in the object (i.e. libfoo.so). Most operating systems allows one to change name that is referenced and we have done so, placing the version number into the shared library (i.e. libfoo.so.0.1). At link time, one would reference libfoo.so, but when one executes the program, the shared library loader would then look for the shared library with the alternate name. Hence multiple versions of shared libraries may be supported relatively easily.[2]

All operating systems (that we have seen) provide a means for programs to specify the location of shared libraries. On different operating systems, this is either specified when creating the shared library, and link time, or both.[3] The build process will hardwire a path to the installed destination.

[2] Under AIX for the RISC/6000, multiple versions of shared libraries are supported by combining two or more versions of the shared library into one file. The Kerberos build procedure produces shared libraries with version numbers in the internal module names, so that the shared libraries are compatible with this scheme. Unfortunately, combining two shared libraries requires internal knowledge of the AIX shared library system beyond the scope of this document. Practicallyspeaking, only one version of AIX shared libraries can be supported on a system, unless the multi-version library is constructed by a programmer familiar with the AIX internals.

[3] Both are necessary sometimes as the shared libraries are dependent on other shared libraries

3.7.2 NetBSD Shared Library Support

Shared library support has been tested under NetBSD 1.0A using GCC 2.4.5. Due to the vagaries of the loader in the operating system, the library load path needs to be specified in building libraries and in linking with them. Unless the library is placed in a standard location to search for libraries, this may make it difficult for developers to work with the shared libraries.

3.7.3 AIX Shared Library Support

AIX specifies shared library versions by combining multiple versions into a single file. Because of the complexity of this process, no automatic procedure for building multi-versioned shared libraries is provided.Therefore, supporting multiple versions of the Kerberos shared libraries under AIX will require significant work on the part of a programmer famiiliar with AIX internals.

AIX allows a single library to be used both as a static library and as a shared library. For this reason, the '`--enable-shared`' switch to configure builds only shared libraries. On other operating systems, both shared and static libraries are built when this switch is specified. As with all other operating systems, only non-shared static libraries are built when '`--enable-shared`' is not specified.

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. In addition, the AIX 4.1 linker is able to build a shared '`libkrb5.a`' when GNU C is used.

3.7.4 Solaris 5.3 Shared Library Support

Shared library support only works when using the Sunsoft C compiler. We are currently using version 3.0.1.

The path to the shared library must be specified at link time as well as when creating libraries.

3.7.5 Alpha OSF/1 Shared Library Support

Shared library support has been tested with V2.1 and higher of the operating system. Shared libraries may be compiled both with GCC and the native compiler.

One of the nice features on this platform is that the paths to the shared libraries is specified in the library itself without requiring that one specify the same at link time.

We are using the '`-rpath`' option to '`ld`' to place the library load path into the executables. The one disadvantage of this is during testing where we want to make sure that we are using the build tree instead of a possibly installed library. The loader uses the contents of '`-rpath`' before LD_LIBRARY_PATH so we must specify a dummy _RLD_ROOT and complete LD_LIBRARY_PATH in our tests.

The one disadvantage with the method we are using

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 to **krb5-bugs@mit.edu**. Thanks!

3.8.1 Ultrix 4.2/3

On the MIPS platform, using the native compiler, '`md4.c`' and '`md5.c`' can not be compiled with the optimizer set at level 1. That is, you must specify either '`--with-ccopts=-O`' and '`--with-ccopts=-g`' to configure. If you don't specify either, the compile will never complete.

The optimizer isn't hung; it just takes an exponentially long time. Compiling 6 out of the 48 algorithmic steps takes 3 seconds; compiling 7 steps takes 9 seconds; compiling 8 steps takes 27 seconds, and so on. Calculations estimate it will finish in approximately 3,469 billion years....

Using GCC version 2.5 instead of the native compiler will also work fine, both with or without optimization.

3.8.2 Alpha OSF/1 V1.3

There is a bug in OSF/1's fgrep which causes the '`configure`' script to fail.

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 instead of the native compiler will also work fine, both with or without optimization.

3.8.3 Alpha OSF/1 V2.0++

There formerly used to be a bug when using the native compiler in compiling 'md4.c' when compiled without either the '-O' or '-g' compiler options. (This could possibly be the same sort of bug as found in Section 3.8.1 [Ultrix 4.2/3], page 15, but that seems rather remarkable.) 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 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.
2. Upgrade to Solaris 2.4

3. Make sure your /etc/nsswitch.conf has the line:

   ```
   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
   ```

3.8.6 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`'.

3.8.7 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.

3.9 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/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/src
% make all
% make install DESTDIR=somewhere-else
```

4 Installation

When you are installing Kerberos for the first time at a site, you must first decide on the realm name you will use for your site, and select a machine to host the *Kerberos server*, which is also known as the *KDC* (*Key Distribution Center*). The KDC must be located on a secure machine, since its database contains the keys for the entire realm. It is extremely important that the database be kept secure, if the realm's Kerberos service is to remain secure.

Once a KDC is installed and configured, you may then set up one or more client machines, and one or more application machines.

4.1 Background Information

Your system's security is only as good as the security of your '`root`' password. You should take other precautions to protect your system security in addition to installing Kerberos. Kerberos cannot protect you from someone who is able to steal '`root`' privileges. Kerberos also does not protect you from break-ins caused by bugs in your daemons (e.g., '`fingerd`' or '`sendmail`'). On almost all UNIX systems, if intruders can break in as an ordinary users, they can obtain superuser privileges by exploiting bugs or imperfect configuration files.

4.2 Installation on any Machine

The following steps must be performed no matter what type of machine (KDC, Client, or Application server) you are installing. All machines functioning within the same administrative domain must use the same Kerberos realm name; and all machines which are using Kerberos must have the Kerberos configuration files properly installed.

If you are installing Kerberos on a machine that will act only as a Kerberos client, this section describes all that you need to do. If you are installing a KDC, or an Kerberos Application server, you will also need to complete the procedures detailed in Section 4.3 [Installing the KDC], page 23, or Section 4.5 [A Sample Application], page 27, after you finish with the procedures found in this section.

4.2.1 Picking a Realm Name

Before you install Kerberos V5 at your site, you have to choose a *realm name*, the name that specifies the system's administrative domain. By convention, we suggest that you use your internet domain name, in capital letters. (Kerberos realm names are case-sensitive, so it's important that your realm name be in all upper case.) For example, if your internet domain is `cygnus.com` (so that you have hostnames such as `ftp.cygnus.com` and `tweedledum.cygnus.com`), then your Kerberos realm should be `CYGNUS.COM`.

4.2.2 Configuration files

4.2.2.1 The '`krb5.conf`' File

The '`krb5.conf`' file contains information needed by the Kerberos V5 library including a system's default Kerberos realm, and the locations of the Kerberos servers.

The '`krb5.conf`' uses an INI-style format. Sections are delimited by square braces; within each section, there are relations where tags can be assigned to have specific values. Tags can also contain a subsection, which contains further relations or subsections. A tag can be assigned to multiple values.

Create a '`/etc/krb5.conf`' file using the following format:

```
[libdefaults]
        default_realm = <realm_name>

[realms]
        <realm_name> = {
                kdc = <master_server_name>
                admin_server = <master_server_name>
                default_domain = <domain_name>
        }

[domain_realm]
        <.domain.name> = <realm_name>
```

Where '`realm_name`' specifies the default realm to be used by that particular system, and '`master_server_name`' specifies the machine name on which you will run the master server. The keywords '`kdc`' and '`admin_server`' lists the location of the realms KDC and administration servers.

For example, if your realm name is 'ATHENA.MIT.EDU' and your master server's name is 'kerberos.mit.edu', the file should have these contents:

```
[libdefaults]
        default_realm = ATHENA.MIT.EDU

[realms]
        ATHENA.MIT.EDU = {
                kdc = KERBEROS.MIT.EDU
                admin_server = KERBEROS.MIT.EDU
                default_domain = MIT.EDU
        }

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

In many situations, the default realm in which a host operates will be identical to its Internet domain name, with the first component removed and all letters capitalized. For example, `ftp.cygnus.com` is traditionally in the realm `CYGNUS.COM`. If this is not the case, you will need to establish a translation from host name or domain name to realm name. This is accomplished with the '`[domain_realm]`' stanza

Each line of the translation file specifies either a host name or domain name, and its associated realm:

```
[domain_realm]
        <.domain.name> = KERBEROS.REALM1
        <host.name> = KERBEROS.REALM2
```

For example, to map all hosts in the domain LSC.MIT.EDU to LCS.MIT.EDU but the host FILMS.LSC.MIT.EDU to MIT.EDU your file would read:

```
[domain_realm]
        .LSC.MIT.EDU = LSC.MIT.EDU
        FILMS.LSC.MIT.EDU = MIT.EDU
```

If a particular host name matches both a domain name and a host name in '`[domain_realm]`', the entry containing the host name takes precedence.

See the '[SOURCE_DIR]/config-files/krb5.conf' file for an example 'krb5.conf' file. That file has examples of how to provide backup servers for a given realm (additional lines with the same leading realm name) and how to designate servers for remote realms.

4.2.2.2 Conversion of V4 configuration files

Kerberos V4's 'krb.conf' and 'krb.realms' files formats are no longer used by the V5 library. A Perl script has been provided to allow for "easy" generation of an initial 'krb5.conf'. It is located in '[SOURCE_DIR]/config-files/convert-config-files'. The produced file should be checked for errors.

Note that if you are planning on using certain applications with Kerberos V4 compatibility compiled in, the V4 library still needs the files 'krb.conf' and 'krb.realms'.

4.2.2.3 /etc/services

All hosts which will use Kerberos will need to have certain ports defined in the system's '/etc/services' file. The file '[SOURCEDIR]/config-files/services.append' contains the entries which should be appended to the '/etc/services' file.

If you are using the Network Information Services (NIS) or Yellow Pages (YP), you will need to add the services in the 'services.append' file to the services exported in the NIS or YP server.

4.2.3 Required Programs

The following files should be installed on all machines which are running Kerberos, either as a client, a KDC, or an application server:

If you used the default 'make install' without using the '--prefix' argument to 'configure', then [K_USER] refers to '/usr/local/bin'.

- '/etc/krb5.conf' — This files contains information required by Kerberos V5 giving system defaults as well as locations of Kerberos servers.
- '[K_USER]/kinit' — This program allows you to obtain Kerberos credentials.

- '[K_USER]/bin/kdestroy' — This program allows you to destroy Kerberos credentials which are no longer needed.
- '[K_USER]/klist' —- This program allows you to list the credentials found in your credentials cache.

4.3 Installing the KDC

4.3.1 The optional 'kdc.conf' profile

There is an optional configuration file 'kdc.conf' that allows one to specify per-realm configuration data to be used by the Kerberos V5 Authentication Service and Key Distribution Center. This information includes database locations, key and per-realm defaults, port numbers, ticket life times, etc. The location of the configuration file is '/usr/local/lib/krb5kdc/kdc.conf'.

See the man page or '[SOURCEDIR]/config-files/kdc.conf.M' for more details.

This document assumes that you do not have the 'kdc.conf' configuration file installed.

Note also that [logging] subsection to '/etc/krb5.conf' can be used to specify where to syslog messages regarding server activity.

4.3.2 Initializing the Kerberos Database

Login to the Kerberos KDC, and use the 'su' command to become root. If you installed the Kerberos administration tools with the 'make install' command and the default pathnames, they should be in the '/usr/local/admin' directory. If you installed the tools in a different directory, hopefully you know what it is. From now on, we will refer to this directory as [ADMIN_DIR].

The default database will be located in the directory '/usr/local/lib/krb5kdc' unless changed in the configuration process. From now on, we will call this [DB_DIR].

The 'kdb5_create' command creates and initializes the master database. It asks you to the database's master password. Do not forget this password. If you do, the database becomes useless. (Your realm name should be substituted for [REALMNAME] below.)

Because the install process does not create [DB_DIR] you need to do so yourself.

Use 'kdb5_create' as follows:

```
# mkdir [DB_DIR]
# [ADMIN_DIR]/kdb5_create
Initializing database '[DB_DIR]/principal' for realm '[REALMNAME]',
master key name 'K/M@[REALMNAME]'
You will be prompted for the database Master Password.
It is important that you NOT FORGET this password.
Enter KDC database master key:             <- [Enter the master password.]
Re-enter KDC database master key to verify: <- [Re-enter it.]
```

4.3.3 Storing the Master Password

The 'kdb5_stash' command "stashes" the master password in the file '[DB_DIR]/.k5.[REALM.NAME]' so that the Kerberos server can be started automatically during an unattended reboot of the master server. Other administrative programs use this hidden password so that they can access the master database without someone having to manually provide the master password. This command is an optional one; if you'd rather enter the master password each time you start the Kerberos server, don't use 'kdb5_stash'.

On the one hand, if you use 'kdb5_stash', a copy of the master key will reside on a disk, which may not be acceptable; on the other hand, if you don't use 'kdb5_stash', the server cannot be started unless someone is around to type the password manually.

The command merely prompts for the master password:

```
# [ADMIN_DIR]/kdb5_stash
Enter KDC database master key:    <- [Enter the master password.]
```

WARNING: If your Kerberos database master key is compromised and the database is obtained, the security of your entire authentication system is compromised. (If this happens you all of your user's passwords must be set to new values manually — i.e., not using Kerberos.) The master key must be a carefully kept secret. If you keep backups, you must guard all the master keys you use, in case someone has stolen an old backup and wants to attack users' whose passwords haven't changed since the backup was stolen. This is why we provide the option not to store it on disk.

4.3.4 Adding Users to the Database

The 'kdb5_edit' program is used to add new users and services to the master database, and to modify existing database information.

For example, to add yourself to the newly created database: (replace '[USERNAME]' with your username with.

```
# [ADMIN_DIR]/kdb5_edit
kdb5_edit:  ank [USERNAME]
Enter password:                           <- [Enter your password.]
Re-enter password for verification:       <- [Re-enter it.]
kdb5_edit: quit
```

The 'ank' command is short for "add_new_key"; this command adds a new user to the database. To see what other commands which 'kdb5_edit' supports, type ? RET at the 'kdb5_edit' prompt.

4.3.5 Starting the Kerberos Server

Assuming that you didn't use the '--prefix' argument to 'configure', then [K_SERVER] refers to '/usr/local/sbin'.

In order to start the Kerberos server, simply run it. It will fork and disassociate from the terminal unless the '-n' argument is used.

```
# [K_SERVER]/krb5kdc
```

If you have used the 'kdb5_stash' command to store the master database password, the server will start automatically. If you did not use 'kdb5_stash', use the following command:

```
# [K_SERVER]/krb5kdc -m
```

The server will prompt you to enter the master password before actually starting itself.

4.3.6 Setting up Slave Kerberos Servers

Slave Kerberos servers allow clients to be able to get Kerberos tickets even when the Master server is not available. Users will not be able to change their passwords — changes can only be made to database on the Master server; however, users will be able to authenticate to application servers, which is critically important in a distributed client-server environment.

4.3.6.1 Kerberos Slave Database Propagation

In order to propagate the Kerberos database from the Master server to the slaves, the '`kprop`' and '`kpropd`' client/server programs are used. Periodically, the Master server will dump the Kerberos database out into an ASCII format, using the '`kdb5_edit`' program. The master server will then run '`kprop`' to propagate the dumped database file to each slave server.

On the slave server, the '`kpropd`' program is invoked out of '`/etc/inetd`' server. After '`kprop`' and '`kpropd`' have mutually authenticated with one another, and '`kpropd`' is satisfied with the identity of the Master server, then the dumped ASCII database is transferred to the slave server in an encrypted fashion. After the database is transfered, '`kpropd`' will then run '`kdb5_edit`' with the appropriate arguments in order to undump the database into a usable form by the KDC on the slave server.

4.3.6.2 Installing a Slave Server

To be written.

4.3.7 Inter-realm Kerberos Operation

To be written.

4.3.8 The Administration Server

There is currently an administration server in the Kerberos source tree. It is, however, very rough, and it will likely be significantly changed or replaced before the 1.0 release. Hence, this manual does not document the current administration server. Changes to the database can be made by logging in to the KDC directly, and using the '`kdb5_edit`' program; see Section 4.3.4 [Adding Users to the Database], page 25.

4.3.9 Testing the Kerberos Server

Use the '`kinit`' command to obtain Kerberos credentials. This command creates your credentials cache and stores your credentials in it.

If you used the default '`make install`' command and directories to install the Kerberos user utilities, '`kinit`' will be in the '`/usr/local/bin`' directory. From now on, we'll refer to the Kerberos user commands directory as [K_USER].

Use '`kinit`' as follows:

```
# [K_USER]/kinit [USERNAME]
Password for [USERNAME]@[REALMNAME]:         <- enter your password
```

Use the '`klist`' program to list the contents of your ticket file.

```
# [K_USER]/klist
Ticket cache: /tmp/krb5cc_15806
Default principal: [USERNAME]@[REALMNAME]

  Valid starting       Expires           Service principal
31-Jan-95 21:58:32   1-Feb-95 05:57:39  krbtgt/[REALMMNAME]@[REALMNAME]
```

4.4 Migrating from V4 to V5 Kerberos

To be written.

4.5 A Sample Application

This release of Kerberos comes with a sample application server and a corresponding client program. You will find this software '`[K_SERVER]/sserver`' and '`[K_USER]/sclient`', which is the server's executable, and the client program's executable, respectively.

The programs are rudimentary. When they have been installed (the installation procedure is described in detail later), they work as follows:

1. The user starts 'sclient' and provides as arguments to the command the name of the server machine and an optional port on which to contact the server.
2. 'sclient' contacts the server machine and authenticates the user to 'sserver'.
3. 'sserver' authenticates itself to 'sclient' (thus performing mutual authentication), and then returns a message to the client program. This message contains the name of the Kerberos principal that was used to authenticate to 'sserver'.
4. 'sclient' displays the server's message on the user's terminal screen.

4.5.1 Installing the Sample Application

In general, you use the following procedure to install a Kerberos-authenticated server-client system.

1. Add the appropriate entry to the Kerberos database using 'kdb_edit'
2. Create a '/etc/v5srvtab' file for the server machine.
3. Install the service program and the '/etc/v5srvtab' file on the server machine.
4. Install the client program on the client machine.
5. Update the '/etc/services' file on the client and server machines.

We will use the sample application as an example, although programs which do not take requests via the 'inetd' program may have slightly different installation procedures. 'Inetd' starts 'sserver' each time a client process contacts the server machine. 'sserver' processes the request, terminates, then is restarted when 'inetd' receives another 'sserver' request. When you install the program on the server, you must add a 'sample' entry to the server machine's '/etc/inetd.conf' file.

The following description assumes that you are installing 'sserver' on the machine 'jimi.mit.edu'. Here's the process, step by step:

1. Login as root or 'su' to root on the Kerberos server machine. Use the 'kdb5_edit' program to create an entry for sample in the Kerberos database:

   ```
   # [ADMIN_DIR]/kdb5_edit
   kdb5_edit: add_random_key sample/jimi.mit.edu
   kdb5_edit: quit
   ```

   (Note: 'add_random_key' may be abbreviated as 'ark'.)

2. Use '`kdb5_edit`' to create a '`srvtab`' file for '`sserver`''s host machine:

   ```
   # [ADMIN_DIR]/kdb5_edit
   kdb5_edit: extract_srvtab jimi.mit.edu sample
   'sample/jimi.mit.edu@ATHENA.MIT.EDU' added to keytab 'WRFILE:jimi.mit.edu-new-srvtab'
   kdb5_edit: quit
   ```

   (Note: '`extract_srvtab`' may be abbreviated as '`xst`'.)

   Transfer the '`jimi.mit.edu-new-srvtab`' file to '`jimi.mit.edu`' and install it as '`/etc/v5srvtab`'.

   WARNING: Note that this file is equivalent to the service's password and should be treated with care. For example, it could be transferred by removable media, but should not be sent over an open network in the clear. This file should installed such that only the root user can read it.

3. Add the following line to the '`/etc/services`' file on '`jimi.mit.edu`', and on all machines that will run the '`sample_client`' program:

   ```
   sample     906/tcp       # Kerberos sample app server
   ```

4. Add a line similar to the following line to the '`/etc/inetd.conf`' file on '`sample_server`''s machine:

   ```
   sample   stream   tcp   nowait   switched   root
       [K_SERVER]/sserver sample_server
   ```

   where [K_SERVER] should be substituted with the path to the '`sserver`' program. (This '`inetd.conf`' information should be placed on one line.) You should examine existing lines in '`/etc/inetd.conf`' and use the same format used by other entries (e.g. for telnet). Most systems do not have a column for the 'switched' keyword, and some do not have a column for the username (usually 'root', as above).

5. Restart '`inetd`' by sending the current '`inetd`' process a hangup signal:

   # kill -HUP *process_id_number*

6. The '`sserver`' is now ready to take '`sclient`' requests.

4.5.2 Testing the Sample Server

Assume that you have installed '`sserver`' on '`jimi.mit.edu`'.

Login to your workstation and use the '`kinit`' command to obtain a Kerberos ticket-granting ticket:

```
% [K_USER]/kinit [USERNAME]
Password for [USERNAME]@[REALMNAME]:          <— Enter your password
```

Now use the '`sclient`' program as follows:

% [K_USER]/sclient jimi

The command should display something like the following:

```
sendauth succeeded, reply is:
reply len 29, contents:
You are [USERNAME]@[REALMNAME]
```

4.6 Installing Kerberos Applications

In addition to the sample application, Kerberos comes with several servers that can be installed on workstations to provide secure network services such as FTP, Telnet, and Rlogin. This section describes these services and how to install them. First, a general procedure is outlined for configuring a workstation to run Kerberos application servers. Then, details of the particular configuration options required by each server are presented.

4.6.1 Levels of Security

Before installing Kerberized servers, it is a good idea to decide on a security policy for your workstation. While developing a comprehensive security policy is beyond the scope of these instructions, there are several interactions between Kerberos servers and non-Kerberos servers, as well as different modes in which Kerberos servers can run that should be considered. In general, there are three levels of connections:

1. Encrypted Kerberos connections are the most secure services provided by the Kerberos environment. Only encrypted services provide confidentiality—encryption is required to make sure a passive eavesdropper does not collect sensitive data, such as passwords, typed over connections. Besides providing confidentiality, encryption provides protection against active attacks. Without encryption or some other form of integrity, an attacker may be able to insert commands or change data in an authenticated session.l
2. The next level of security is Kerberos-authenticated services without encryption. Without encryption, there is no confidentiality, and some active attacks are possible. However, unencrypted services are faster and are available commercially outside the United States. In addition, the window of exposure to attack is generally limited to the lifetime of a session—it is difficult to start a new session if a Kerberos authentication exchange must take place at the beginning of every session.

3. Passworded services provide the next lower level of security. Unfortunately, it is all-to-easy for a passive attacker to use software such as a packet sniffer to find passwords traveling over a network. Additionally, once an attacker knows a password, they can start additional sessions at future times until the password is changed from a secure location. Despite the disadvantages of password-based network services, there are some common reasons for enabling them. First, most clients only support password-based services; not everyone has Kerberos or other cryptographically-secure network services. Second, in some environments, where the network is secure, passwords may be a reasonable solution. Also, particularly on public-access systems, it is common to enable password-based services for normal users, while using more-secure services such as Kerberos for root logins by administrators.
4. The least secure common category of services are trust or host-based services.These services use an IP address or client-supplied assertion to provide authentication. Examples of host-based services include '`rlogin`', '`rsh`', and the '`on`' or '`rexd`' service. It is generally sufficient to know that a user exists on a target system, and to know that a host-based or trust-based service is enabled in order to exploit the service. Whenever possible, these services should be disabled; there are few situations in which an a security policy is both compatible with Kerberos and host-based services.

Next, decide whether Kerberos4 compatibility is necessary. Unencrypted Kerberos4 services suffer from all the problems of unencrypted Kerberos V5 services: lack of confidentiality and susceptibility to active attack. In addition, the lack of a replay cache in Kerberos4makes these active attacks much easier. Also, design bugs in the Kerberos4 BSD utilities such as '`rlogin`', '`rsh`' and '`rcp`' cause the availability of an unencrypted service to significantly decrease the security of a system, even if only the encrypted service is ever used. For example, a system that runs both encrypted and unencrypted Kerberos4 '`rlogin`' servers is less secure than a system only running the encrypted service, even if users only ever connect to the encrypted service.

Therefore, if Kerberos4 interoperability is desired or required, try to avoid running unencrypted Kerberos4 services wherever possible. In particular, only enable encrypted '`rlogin`' if at all possible. Naturally, some environments will require additional Kerberos4 functionality, like '`rsh`'.The Kerberos V5 versions of these services, with Kerberos4 interoperability enabled are not any weaker than their Kerberos4 counterparts. So, if the existing Kerberos4 security policy allows these services, then enabling them under the Kerberos V5 security policy should not be a problem.

Finally, the issue of compatibility with older Kerberos V5 clients must be considered. For the most part, this compatibility is automatic, and has few security implications. The major exception to this is the BSD utilities. Until Kerberos V5 Beta6, these utilities inherited a few design defects from the Kerberos V4 BSD utilities. In particular, the presence of an unencrypted service that was never used adversely effected the security of an encrypted service. The solution that was adopted preserves compatibility with Kerberos V5 Beta5 clients. Unfortunately, older clients are incompatible with this scheme. If interoperability with older clients is necessary, then the new scheme for checksums can be disabled on the server. This results in slightly less secure operation, but allows for compatibility with the older clients. Alternatively, sites wishing to enable all security features may wish to disable compatibility with Kerberos V5 Beta5 BSD utilities. See Section 4.6.3.1 [Checksums], page 34, for full details.

In conclusion, as you prepare to install Kerberos clients, you should have answers to the following questions:

1. What levels of services are appropriate for your security policy: encrypted services, authenticated but not encrypted services, password-based services, or host-based services?
2. Do you wish to enable Kerberos V4 compatibility? If so, do you need to enable unencrypted services?
3. Do you need compatibility with Kerberos V5 clients before Beta5 (released in May, 1995)? Do you wish to disable compatibility with Beta5 clients for slightly enhanced security?

4.6.2 Preparing a Workstation for Kerberos Application Servers

The following procedure is very similar to the installation procedure for the sample Kerberos server. See Section 4.5.1 [Installing the Sample Application], page 28. However, instead of using the sample service, this configuration is designed to set up host-based services.

The following description assumes that you are installing '`sserver`' on the machine '`jimi.mit.edu`'. Repeat these steps for each workstation that will be running host-based servers. Note that it is not necessary to run these procedures on client machines–machines used only to connect to secure network services, but that do not run any servers of their own. Here's the process, step by step:

1. Login as root or '`su`' to root on the Kerberos server machine. Use the '`kdb5_edit`' program to create a service entry for `host` in the Kerberos database:

   ```
   # [ADMIN_DIR]/kdb5_edit
   kdb5_edit: add_random_key host/jimi.mit.edu
   kdb5_edit: quit
   ```

   (Note: '`add_random_key`' may be abbreviated as '`ark`'.)

2. Use '`kdb5_edit`' to create a '`srvtab`' file for '`sserver`''s host machine:

   ```
   # [ADMIN_DIR]/kdb5_edit
   kdb5_edit: extract_srvtab jimi.mit.edu host
   'host/jimi.mit.edu@ATHENA.MIT.EDU' added to keytab 'WRFILE:jimi.mit.edu-new-srvtab'
   kdb5_edit: quit
   ```

(Note: '`extract_srvtab`' may be abbreviated as '`xst`'. Also, additional services can be listed after '`host`' on the '`extract_srvtab`' line; for example if the host jimi also runs the sample server, the '`sample`' service might have been listed after host.)

Transfer the '`jimi.mit.edu-new-srvtab`' file to '`jimi.mit.edu`' and install it as '`/etc/v5srvtab`'.

WARNING: Note that this file is equivalent to the service's password and should be treated with care. For example, it could be transferred by removable media, but should not be sent over an open network in the clear. This file should be installed such that only the root user can read it.

3. Make sure that the '`/etc/services`' file has been updated to include Kerberos services. In particular, look for '`klogin`', '`eklogin`', and '`kshell`'.
4. For each server you plan to run, add a line to '`/etc/inetd.conf`'. The following sections will give details on what this line should look like for each Kerberos application service.
5. Consider removing non-Kerberized services like '`rlogin`', '`rsh`', '`rexd`', and others, in accordance with the security policy you decided on in the last section.
6. Restart '`inetd`'. On most systems, this is done by sending the current '`inetd`' process a hangup signal:

   # kill -HUP *process_id_number*
7. Try using the Kerberos applications to connect to the host.

4.6.3 BSD Utilities

This section describes installing servers for the BSD utilities: '`kshd`' and '`klogind`'.The '`klogind`' server implementsthe protocol used by the Kerberized '`rlogin`' client. The '`kshd`' server implements the protocol used by the '`rsh`' and '`rcp`' clients.

These daemons take a common set of options to enable support for different versions of Kerberos. The '`-5`' option enables Kerberos V5 support, and the '`-4`' option enables Kerberos V4 support. At least one of these options must be supplied or the server will refuse all connections. Both options can be supplied if compatibility with both versions of Kerberos is desired.

Both the '`klogind`' and '`kshd`' take an '`-e`' option to control encryption; because of the design of the servers, it works slightly differently. The Kerberos login service listens on two different ports, one for encrypted connections, and one for unencrypted connections. Because '`klogind`' is started by '`inetd`', it needs to know whether it is servicing an encrypted or unencrypted connection. Thus, the '`-e`' option tells '`klogind`' that it is listening to the encrypted port. Typically, systems that allow both encrypted and unencrypted logins have two lines in '`inetd.conf`', one for the '`klogin`' service and one for the '`eklogin`' service. The line for the '`eklogin`' service uses the '`-e`' option, while the line for the 'klogin' service does not. Systems only supporting encrypted logins simply have the 'eklogin' line.

On the other hand, 'kshd' listens to encrypted and unencrypted requests on the same port; information from the client identifies the connection's encryption status. So, 'kshd' interprets the '-e' option to mean that encryption is required. If this option is specified, unencrypted connections are dropped. Thus, specifying '-e' to 'kshd' is like only having a line for 'eklogin'.

4.6.3.1 Checksums

Under previous versions of Kerberos, it was possible for an active attacker to change certain information in the initial authentication exchange without this change being detected. Starting with Kerberos V5, Beta6, this information is protected by a checksum passed in the Kerberos authenticator. Ideally, the server could detect the presence of this checksum and use it if it were present. This way, administrators could be sure to use new clients that produced the checksum, while users who were not as concerned about security could use a wider range of clients.

This is how the checksum feature works by default. Unfortunately, clients previous to Kerberos V5, Beta5 used the checksum field to incorporate semi-random data into the Kerberos authentication exchange. It is impossible to distinguish these bogus checksums from checksums that have been produced by modern clients, but modified by an active attacker. Thus, the checksum feature is incompatible with previous releases of Kerberos V5. Three modes of operation are provided to meet the configuration needs of different sites:

1. The '-c' option to both 'kshd' and 'klogind' requires that checksums be present and valid. This only works with clients more recent than Kerberos V5, Beta6. An error will be given when an older client tries to connect. This option is incompatible with Kerberos V4, Kerberos V4 clients do not include a checksum. If this option is used in conjunction with the '-4' option for Kerberos V4 compatibility, a warning about the configuration error will be logged on each connection.
2. If no checksum-related option is specified, then checksums will be validated if they are present, but not required.This is compatible with Kerberos V4 and Kerberos V5 clients as early as Kerberos V5, Beta5.
3. If the '-i' option is provided, then checksums are ignored, even if present. This option is required to maintain compatibility with clients older than Kerberos V5 Beta5. Site should upgrade to newer clients so this option is not required.

4.6.3.2 BSD Utility Configuration Example

This section includes sample entries for '/etc/inetd.conf'. Sample configuration are presented for three systems. Note that while the examples stretch over multiple lines, an entry for a single service should be placed on only one line in 'inetd.conf'. Also, not all systems have all of the columns shown in the example. See Section 4.5.1 [Installing the Sample Application], page 28, for details on adding services to 'inetd.conf'.

The first system enables all security features. Only Kerberos V5 encrypted connections are enabled, and checksums are required.

```
eklogin         stream   tcp   nowait     root
    [K_SERVER]/klogind klogind -5 -e -c
kshell          stream  tcp     nowait  root
    [K_SERVER]/kshd kshd -5 -e -c
```

The second system enables encrypted services for both Kerberos V5 and Kerberos V4. Checksums are not required, but since no V5 clients earlier than Beta5 are used, they are not ignored.

```
eklogin         stream   tcp   nowait     root
    [K_SERVER]/klogind klogind -5 -4 -e
kshell          stream  tcp     nowait  root
    [K_SERVER]/kshd kshd -5 -4 -e
```

The final example has both encrypted and unencrypted services enabled for both Kerberos V5 and Kerberos V4. Checksums are disabled to preserve compatability with older V5 clients.

```
eklogin         stream   tcp   nowait     root
    [K_SERVER]/klogind klogind -5 -4  -i -e
klogin         stream   tcp   nowait     root
    [K_SERVER]/klogind klogind -5 -4  -i
kshell          stream  tcp     nowait  root
    [K_SERVER]/kshd kshd -5 -4 -i
```

4.6.4 Telnet and FTP

The following are example entries for 'inetd.conf' for the 'telnetd' and 'ftpd' servers. See Section 4.5.1 [Installing the Sample Application], page 28, for details on adding services to 'inetd.conf'. Note that unlike other 'inetd.conf' examples in this document, these line should replace existing lines for password-based services of the same name.

The first example only allows encrypted or authenticated connections.

```
telnet          stream  tcp     nowait  root
    [K_SERVER]/telnetd telnetd -a user
ftp             stream  tcp     nowait  root
    [K_server]/ftpd ftpd -a
```

The second example also allows password-based connections to be made.

```
telnet          stream  tcp     nowait  root
    [K_SERVER]/telnetd telnetd
ftp             stream  tcp     nowait  root
    [K_server]/ftpd ftpd
```

4.7 Common Kerberos Service Names

The following service names are used by Kerberos client/server applications.

`host` Used by '`telnet`', '`rlogin`', '`rsh`', '`rcp`', '`ftp`'and other services which generally give login access to the host.

`pop` Used by the Post Office Protocol.

`sample` Used by the Kerberos sample applications.

These service names are used as the first component of the server's principal name, with the second component being the server's fully qualified domain name, in lower case.

5 Troubleshooting

To be written.

Table of Contents