Kerberos Installation Notes
DRAFT

Bill Bryant
Jennifer Steiner
John Kohl

Project Athena, MIT

Initial Release, January 24, 1989
(plus later patches through patchlevel 7)

The release consists of three parts.

The first part consists of the core Kerberos system, which was developed at MIT and does not require additional licenses for us to distribute. Included in this part are the Kerberos authentication server, the Kerberos library, the *ndbm* database interface library, user programs, administration programs, manual pages, some applications which use Kerberos for authentication, and some utilities.

The second part is the Data Encryption Standard (DES) library, which we are distributing only within the United States.

The third part contains Kerberos modifications to Sun's NFS, which we distribute as ''context diffs'' to the Sun NFS source code. Its distribution is controlled to provide an accounting of who has retrieved the patches, so that Project Athena can comply with its agreements with Sun regarding distribution of these changes.

1. Organization of the Source Directory

The Kerberos building and installation process, as described in this document, builds the binaries and executables from the files contained in the Kerberos source tree, and deposits them in a separate object tree. This is intended to easily support several different build trees from a single source tree (this is useful if you support several machine architectures). We suggest that you copy the Kerberos sources into a */mit/kerberos/src* directory, and create as well a */mit/kerberos/obj* directory in which to hold the executables. In the rest of this document, we'll refer to the Kerberos source and object directories as [SOURCE_DIR] and [OBJ_DIR], respectively.

Below is a brief overview of the organization of the complete source directory. More detailed descriptions follow.

admin	utilities for the Kerberos administrator
appl	applications that use Kerberos
appl/bsd	Berkeley's rsh/rlogin suite, using Kerberos
appl/knetd	(old) software for inetd-like multiplexing of a single TCP listening port
appl/sample	sample application servers and clients
appl/tftp	Trivial File Transfer Protocol, using Kerberos
include	include files
kadmin	remote administrative interface to the Kerberos master database
kuser	assorted user programs
lib	libraries for use with/by Kerberos
lib/acl	Access Control List library
lib/des	Data Encryption Standard library (US only)
lib/kadm	administrative interface library
lib/kdb	Kerberos server library interface to *ndbm*
lib/knet	(old) library for use with **knetd**
lib/krb	Kerberos library
man	manual pages
prototypes	sample configuration files
server	the authentication server
slave	Kerberos slave database propagation software
tools	shell scripts for maintaining the source tree
util	utilities
util/imake	Imakefile-to-Makefile ‘‘compilation’’ tool
util/ss	Sub-system library (for command line subsystems)
util/et	Error-table library (for independent, unique error codes)
util/makedepend	Makefile dependency generator tool

1.1 The *admin* Directory

This directory contains source for the Kerberos master database administration tools.

kdb_init This program creates and initializes the Kerberos master database. It prompts for a Kerberos realmname, and the Kerberos master password.

kstash This program ''stashes'' the master password in the file */.k* so that the master server machine can restart the Kerberos server automatically after an unattended reboot. The hidden password is also available to administrative programs that have been set to run automatically.

kdb_edit This program is a low-level tool for editing the master database.

kdb_destroy This program deletes the master database.

kdb_util This program can be used to dump the master database into an ascii file, and can also be used to load the ascii file into the master database.

ext_srvtab This program extracts information from the master database and creates a host-dependent *srvtab* file. This file contains the Kerberos keys for the host's ''Kerberized'' services. These services look up their keys in the *srvtab* file for use in the authentication process.

1.2 The *kuser* Directory

This directory contains the source code for several user-oriented programs.

kinit This program prompts users for their usernames and Kerberos passwords, then furnishes them with Kerberos ticket-granting tickets.

kdestroy This program destroys any active tickets. Users should use *kdestroy* before they log off their workstations.

klist This program lists a user's active tickets.

ksrvtgt This retrieves a ticket-granting ticket with a life time of five minutes, using a server's secret key in lieu of a password. It is primarily for use in shell scripts and other batch facilities.

ksu Substitute user id, using Kerberos to mediate attempts to change to ''root''.

1.3 The *appl* Directory

If your site has the appropriate BSD license, your Kerberos release provides certain Unix utilities The Berkeley programs that have been modified to use Kerberos authentication are found in the *appl/bsd* directory. They include *login*, *rlogin*, *rsh*, and *rcp*, as well as the associated daemon programs *kshd* and *klogind*. The *login* program obtains ticket-granting tickets for users upon login; the other utilities provide authenticated Unix network services.

The *appl* directory also contains samples Kerberos application client and server programs, an authenticated *tftp* program, *knetd*, an authenticated inet daemon.

1.4 The *server* Directory

The *server* directory contains the Kerberos KDC server, called *kerberos*. This program manages read-only requests made to the master database, distributing tickets and encryption keys to clients requesting authentication service.

1.5 The *kadmin* Directory

The *kadmin* directory contains the Kerberos administration server and associated client programs. The server accepts network requests from the user program *kpasswd* (used to change a user's password), the Kerberos administration program *kadmin*, and the srvtab utility program *ksrvutil*. The administration server can make modifications to the master database.

1.6 The *include* Directory

This directory contains the *include* files needed to build the Kerberos system.

1.7 The *lib* Directory

The *lib* directory has six subdirectories: *acl*, *des*, *kadm*, *kdb*, *knet*, and *krb*. The *des* directory contains source for the DES encryption library. The *kadm* directory contains source for the Kerberos administration server utility library. The *kdb* directory contains source for the Kerberos database routine library. The *knet* directory contains source for a library used by clients of the *knetd* server. The *krb* directory contains source for the *libkrb.a* library. This library contains routines that are used by the Kerberos server program, and by applications programs that require authentication service.

1.8 The *man* Directory

This directory contains manual pages for Kerberos programs and library routines.

1.9 The *prototypes* Directory

This directory contains prototype */etc/services* and */etc/krb.conf* files. New entries must be added to the */etc/services* file for the Kerberos server, and possibly for Kerberized applications (*services.append* contains the entries used by the Athena-provided servers & applications, and is suitable for appending to your existing */etc/services* file.). The */etc/krb.conf* file defines the local Kerberos realm for its host and lists Kerberos servers for given realms. The */etc/krb.realms* file defines exceptions for mapping machine names to Kerberos realms.

1.10 The *tools* Directory

This directory contains a makefile to set up a directory tree for building the software in, and a shell script to format code in the style we use.

1.11 The *util* Directory

This directory contains several utility programs and libraries. Included are Larry Wall's *patch* program, a *make* pre-processor program called *imake*, and a program for generating Makefile dependencies, *makedepend*, as well as the Sub-system library and utilities (*ss*), and the Error table library and utilities (*et*).

2. Preparing for Installation

This document assumes that you will build the system on the machine on which you plan to install the Kerberos master server and its database. You'll need about 10 megabytes for source and executables.

By default, there must be a */kerberos* directory on the master server machine in which to store the Kerberos database files. If the master server machine does not have room on its root partition for these files, create a */kerberos* symbolic link to another file system.

3. Preparing for the Build

Before you build the system, you have to choose a **realm name**, the name that specifies the system's administrative domain. Project Athena uses the internet domain name ATHENA.MIT.EDU to specify its Kerberos realm name. We recommend using a name of this form. **NOTE:** the realm-name is case sensitive; by convention, we suggest that you use your internet domain name, in capital letters.

Edit the [SOURCE_DIR]/*include/krb.h* file and look for the following lines of code:

```
/*
 * Kerberos specific definitions
 *
 * KRBLOG is the log file for the kerberos master server.
 * KRB_CONF is the configuration file where different host
 * machines running master and slave servers can be found.
 * KRB_MASTER is the name of the machine with the master
 * database.  The admin_server runs on this machine, and all
 * changes to the db (as opposed to read-only requests, which
 * can go to slaves) must go to it.
 * KRB_HOST is the default machine when looking for a kerberos
 * slave server.  Other possibilities are in the KRB_CONF file.
 * KRB_REALM is the name of the realm.
 */

#ifdef notdef
this is server-only, does not belong here;
#define       KRBLOG          "/kerberos/kerberos.log"
are these used anyplace '?';
#define               VX_KRB_HSTFILE  "/etc/krbhst"
#define               PC_KRB_HSTFILE  "\\kerberos\\krbhst"
#endif

#define               KRB_CONF        "/etc/krb.conf"
#define               KRB_RLM_TRANS   "/etc/krb.realms"
#define               KRB_MASTER      "kerberos"
#define               KRB_HOST         KRB_MASTER
#define               KRB_REALM       "ATHENA.MIT.EDU"
```

Edit the last line as follows:

1. Change the KRB_REALM definition so that it specifies the realm name you have chosen for your Kerberos system. This is a default which is usually overridden by a configuration file on each machine; however, if that config file is absent, many programs will use this "built-in" realm name.

3.1 The */etc/krb.conf* File

Create a */etc/krb.conf* file using the following format:

realm_name
realm_name ***master_server_name*** `admin server`

where *realm_name* specifies the system's realm name, and *master_server_name* specifies the machine name on which you will run the master server. The words 'admin server' must appear next to the name of the server on which you intend to run the administration server (which must be a machine with access to the database).

For example, if your realm name is *tim.edu* and your master server's name is *kerberos.tim.edu*, the file should have these contents:

```
tim.edu
tim.edu  kerberos.tim.edu admin server
```

See the [SOURCE_DIR]/*prototypes/etc.krb.conf* file for an example */etc/krb.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.

3.2 The */etc/krb.realms* File

In many situations, the default realm in which a host operates will be identical to the domain portion its Internet domain name.

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 */etc/krb.realms* file.

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

```
.domain.name kerberos.realm1
host.name kerberos.realm2
```

For example, to map all hosts in the domain LSC.TIM.EDU to KRB.REALM1 but the host FILMS.LSC.TIM.EDU to KRB.REALM2 your file would read:

```
.LSC.TIM.EDU KRB.REALM1
FILMS.LSC.TIM.EDU KRB.REALM2
```

If a particular host matches both a domain and a host entry, the host entry takes precedence.

4. Building the Software

Before you build the software read the **README** file in [SOURCE_DIR]. What follows is a more detailed description of the instructions listed in README.

1. Create an [OBJ_DIR] directory to hold the tree of Kerberos object files you are about to build, for example, */mit/kerberos/obj*.
2. Change directory to [OBJ_DIR]. The following command creates directories under [OBJ_DIR] and installs Makefiles for the final build.

   `host%` **make -f [SOURCE_DIR]/tools/makeconfig SRCDIR=[SOURCE_DIR]**
3. Change directory to util/imake.includes. Read through config.Imakefile, turning on appropriate flags for your installation. Change SRCTOP so that it is set to the top level of your source directory.
4. Check that your machine type has a definition in include/osconf.h & related files in the source tree (if it doesn't, then you may need to create your own; if you get successful results, please post to kerberos@athena.mit.edu)
5. Change directory to [OBJ_DIR]. The next command generates new Makefiles based on the configuration you selected in config.Imakefile, then adds dependency information to the Makefiles, and finally builds the system:

   `host%` **make world**

   This command takes a while to complete; you may wish to redirect the output onto a file and put the job in the background:

   `host%` **make world >&WORLDLOG_891201 &**

   If you need to rebuild the Kerberos programs and libraries after making a change, you can usually just type:

```
host% make all
```

However, if you changed the configuration in config.Imakefile or modified the Imakefiles or Makefiles, you should run *make world* to re-build all the Makefiles and dependency lists.

4.1 Testing the DES Library

Use the *verify* command to test the DES library implementation:

```
host% [OBJ_DIR]/lib/des/verify
```

The command should display the following:

```
Examples per FIPS publication 81, keys ivs and cipher
in hex.  These are the correct answers, see below for
the actual answers.

Examples per Davies and Price.

EXAMPLE ECB     key = 08192a3b4c5d6e7f
        clear = 0
        cipher = 25 dd ac 3e 96 17 64 67
ACTUAL ECB
        clear ""
        cipher  = (low to high bytes)
                25 dd ac 3e 96 17 64 67

EXAMPLE ECB     key = 0123456789abcdef
        clear = "Now is the time for all "
        cipher = 3f a4 0e 8a 98 4d 48 15 ...
ACTUAL ECB
        clear "Now is the time for all "
        cipher  = (low to high bytes)
                3f a4 0e 8a 98 4d 48 15

EXAMPLE CBC     key = 0123456789abcdef  iv = 1234567890abcdef
        clear = "Now is the time for all "
        cipher =        e5 c7 cd de 87 2b f2 7c
                        43 e9 34 00 8c 38 9c 0f
                        68 37 88 49 9a 7c 05 f6
ACTUAL CBC
        clear "Now is the time for all "
        ciphertext = (low to high bytes)
                e5 c7 cd de 87 2b f2 7c
                43 e9 34 00 8c 38 9c 0f
                68 37 88 49 9a 7c 05 f6
                00 00 00 00 00 00 00 00
                00 00 00 00 00 00 00 00
                00 00 00 00 00 00 00 00
                00 00 00 00 00 00 00 00
                00 00 00 00 00 00 00 00
        decrypted clear_text = "Now is the time for all "
EXAMPLE CBC checksum    key =  0123456789abcdef iv =  1234567890abcdef
        clear =         "7654321 Now is the time for "
        checksum        58 d2 e7 7e 86 06 27 33  or some part thereof
ACTUAL CBC checksum
                encrypted cksum = (low to high bytes)
                58 d2 e7 7e 86 06 27 33
```

If the *verify* command fails to display this information as specified above, the implementation of DES for your hardware needs to be adjusted. Your Kerberos system cannot work properly if your DES library fails this test.

When you have finished building the software, you will find the executables in the object tree as follows:

[OBJ_DIR]/admin *ext_srvtab*, *kdb_destroy*, *kdb_edit*, *kdb_init*, *kdb_util*, and *kstash*.

[OBJ_DIR]/kuser *kdestroy*, *kinit*, *klist*, *ksrvtgt*, and *ksu*.

[OBJ_DIR]/server *kerberos*.

[OBJ_DIR]/appl/bsd *klogind*, *kshd*, *login.krb*, *rcp*, *rlogin*, and *rsh*.

[OBJ_DIR]/appl/knetd *knetd*.

[OBJ_DIR]/appl/sample
sample_server, *sample_client*, *simple_server*, and *simple_client*.

[OBJ_DIR]/appl/tftp *tcom*, *tftpd*, and *tftp*.

[OBJ_DIR]/slave *kprop* and *kpropd*.

5. Installing the Software

To install the software, issue the *make install* command from the [OBJ_DIR] (you need to be a privileged user in order to properly install the programs). Programs can either be installed in default directories, or under a given root directory, as described below.

5.1 The ‘‘Standard’’ Places

If you use the *make* command as follows:

`host#` **make install**

the installation process will try to install the various parts of the system in ‘‘standard’’ directories. This process creates the ‘‘standard’’ directories as needed.

The standard installation process copies things as follows:

- The *include* files *krb.h*, *des.h*, *mit-copyright.h*, *kadm.h* and *kadm_err.h* get copied to the */usr/include* directory.
- The Kerberos libraries *libdes.a*, *libkrb.a*, *libkdb.a*, *libkadm.a*, *libknet.a*, and *libacl.a* get copied to the */usr/athena/lib* (or wherever you pointed LIBDIR in config.Imakefile) directory.
- The Kerberos master database utilities *kdb_init*, *kdb_destroy*, *kdb_edit*, *kdb_util*, *kstash*, and *ext_srvtab* get copied to the */usr/etc* (DAEMDIR) directory.
- The Kerberos user utilities *kinit*, *kdestroy*, *klist*, *ksrvtgt* and *ksu* get copied to the */usr/athena* (PROGDIR) directory.
- The modified Berkeley utilities *rsh*, *rlogin* get copied to the */usr/ucb* (UCBDIR) directory; *rcp* gets copied to the */bin* (SLASHBINDIR) directory; and *rlogind*, *rshd*, and *login.krb* get copied to the */usr/etc* (DAEMDIR) directory. The old copies of the user programs are renamed *rsh.ucb*, *rlogin.ucb* and *rcp.ucb*, respectively. The Kerberos versions of these programs are designed to fall back and execute the original versions if something prevents the Kerberos versions from succeeding.
- The Kerberos version of *tftp* and *tcom* get copied to the */usr/athena* (PROGDIR) directory; *tftpd* gets copied to the */etc* (ETCDIR) directory. *tftp* and *tftpd* are installed set-uid to an unprivileged user (user id of DEF_UID).
- The *knetd* daemon gets copied to the */usr/etc* (DAEMDIR) directory.

- The Kerberos server *kerberos*, the slave propagation software *kprop* and *kpropd*, and the administration server *kadmind* get copied to the */usr/etc* (SVRDIR, SVRDIR, and DAEMDIR) directory.
- The remote administration tools *kpasswd*, *ksrvutil* and *kadmin* get copied to the */usr/athena* (PROGDIR) directory.
- The Kerberos manual pages get installed in the appropriate */usr/man* directories. Don't forget to run *makewhatis* after installing the manual pages.

5.2 ''Non-Standard'' Installation

If you'd rather install the software in a different location, you can use the *make* command as follows, where [DEST_DIR] specifies an alternate destination directory which will be used as the root for the installed programs, i.e. programs that would normally be installed in /usr/athena would be installed in [DEST_DIR]/usr/athena.

`host#` **make install DESTDIR=[DEST_DIR]**

6. Conclusion

Now that you have built and installed your Kerberos system, use the accompanying Kerberos Operation Notes to create a Kerberos Master database, install authenticated services, and start the Kerberos server.

7. Acknowledgements

We'd like to thank Henry Mensch and Jon Rochlis for helping us debug this document.

Table of Contents