## NAME

krb_sendauth, krb_recvauth, krb_net_write, krb_net_read – Kerberos routines for sending authentication via network stream sockets

## SYNOPSIS

**cc** [ *flag* ... ] *file* ... **–lkrb** [ *library* ... ]

**#include <kerberos/krb.h>**
**#include <netinet/in.h>**

**int krb_sendauth(const long** *options***, const int** *fd***, KTEXT** *ktext***, const char** **service***,**
    **const char** **inst***, const char** **realm***, const u_long** *checksum,*
    **MSG_DATA** **msg_data***, CREDENTIALS** **cred***, Key_schedule** *schedule***,**
    **const struct sockaddr_in** **laddr***, const struct sockaddr_in** **faddr***,**
    **const char** **version***);**

**int krb_recvauth(const long** *options***, const int** *fd***, KTEXT** *ktext***, const char** **service***,**
    **char** **inst***, const struct sockaddr_in** **faddr***, const struct sockaddr_in** **laddr,*
    **AUTH_DAT** **auth_data***, const char** **filename***, Key_version** *schedule***,**
    **char** **version***);**

**int krb_net_write(const int** *fd***, const char** **buf***, const int** *len***);**

**int krb_net_read(const int** *fd***, char** **buf***, const int** *len***);**

## MT-LEVEL

Unsafe

## DESCRIPTION

These functions, which are built on top of the core Kerberos library, provide a convenient means for client and server programs to send authentication messages to one another through network connections.

The **krb_sendauth( )** function sends an authenticated ticket from the client program to the server program by writing the ticket to a network socket.

The **krb_recvauth( )** function receives the ticket from the client by reading from a network socket.

### krb_sendauth( )

This function writes the ticket to the network socket specified by the file descriptor *fd*, returning KSUCCESS if the write proceeds successfully, and an error code if it does not.

The *ktext* argument should point to an allocated KTEXT_ST structure. The *service*, *inst*, and *realm* arguments specify the server program's Kerberos principal name, instance, and realm. If you are writing a client that uses the local realm exclusively, you can set the *realm* argument to NULL.

The *version* argument allows the client program to pass an application-specific version string that the server program can then match against its own version string. The *version* string can be up to KSEND_VNO_LEN (see **<kerberos/krb.h>**) characters in length.

The *checksum* argument can be used to pass checksum information to the server program. The client program is responsible for specifying this information. This checksum information is difficult to corrupt because **krb_sendauth( )** passes it over the network in encrypted form. The *checksum* argument is passed as the checksum argument to **krb_mk_req( )** (see **kerberos**(3N)).

You can set **krb_sendauth( )**’s other arguments to NULL unless you want the client and server programs to mutually authenticate themselves. In the case of mutual authentication, the client authenticates itself to the server program, and demands that the server in turn authenticate itself to the client.

**krb_sendauth( ) and Mutual Authentication**

If you want mutual authentication, make sure that you read all pending data from the local socket before calling **krb_sendauth( )**. Set **krb_sendauth( )**’s *options* argument to KOPT_DO_MUTUAL (this macro is defined in **<kerberos/krb.h>**); make sure that the *laddr* argument points to the address of the local socket, and that *faddr* points to the foreign socket’s network address.

**krb_sendauth( )** fills in the other arguments — *msg_data*, *cred*, and *schedule* — before sending the ticket to the server program. You must, however, allocate space for these arguments before calling the function.

**krb_sendauth( )** supports two other options: KOPT_DONT_MK_REQ and KOPT_DONT_CANON. If called with *options* set as KOPT_DONT_MK_REQ, **krb_sendauth( )** will not use the **krb_mk_req( )** (see **kerberos**(3N)) function to retrieve the ticket from the Kerberos server. The *ktext* argument must point to an existing ticket and authenticator (such as would be created by **krb_mk_req( )**), and the *service*, *inst*, and *realm* arguments can be set to NULL.

If called with *options* set as KOPT_DONT_CANON, **krb_sendauth( )** will not convert the service’s instance to canonical form using **krb_get_phost( )** (see **krb_realmofhost**(3N)).

If you want to call **krb_sendauth( )** with a multiple *options* specification, construct *options* as a bitwise-OR of the options you want to specify.

**krb_recvauth( )**

The **krb_recvauth( )** function reads a ticket/authenticator pair from the socket pointed to by the *fd* argument. Set the *options* argument as a bitwise-OR of the options desired. Currently only KOPT_DO_MUTUAL is useful to the receiver.

The *ktext* argument should point to an allocated KTEXT_ST structure. **krb_recvauth( )** fills *ktext* with the ticket/authenticator pair read from *fd*, then passes it to **krb_rd_req( )** (see **kerberos**(3N)).

The *service* and *inst* arguments specify the expected service and instance for which the ticket was generated. They are also passed to **krb_rd_req( )** (see **kerberos**(3N)). The *inst* argument may be set to "*" if the caller wishes **krb_mk_req( )** (see **kerberos**(3N)) to fill in the instance used (note that there must be space in the *inst* argument to hold a full instance name, see **krb_mk_req( )** on **kerberos**(3N)).

The *faddr* argument should point to the address of the peer which is presenting the ticket. It is also passed to **krb_rd_req( )** (see **kerberos**(3N)).

If the client and server plan to mutually authenticate one another, the *laddr* argument should point to the local address of the file descriptor. Otherwise you can set this argument to NULL.

The *auth_data* argument should point to an allocated AUTH_DAT area. It is passed to and filled in by **krb_rd_req( )** (see **kerberos**(3N)). The checksum passed to the corresponding **krb_sendauth( )** is available as part of the filled-in AUTH_DAT area.

The *filename* argument specifies the filename which the service program should use to obtain its service key. **krb_recvauth( )** passes *filename* to the **krb_rd_req( )** function, see **kerberos**(3N), If you set this argument to " ", **krb_rd_req( )** looks for the service key in the file **/etc/srvtab**.

If the client and server are performing mutual authentication, the *schedule* argument should point to an allocated Key_schedule. Otherwise it is ignored and may be NULL.

The *version* argument should point to a character array of at least KSEND_VNO_LEN characters. It is filled in with the version string passed by the client to **krb_sendauth( )**.

**krb_net_write( ) and krb_net_read( )**

The **krb_net_write( )** function emulates the **write**(2) system call, but guarantees that all data specified is written to *fd* before returning, unless an error condition occurs.

The **krb_net_read( )** function emulates the **read**(2) system call, but guarantees that the requested amount of data is read from *fd* before returning, unless an error condition occurs.

## SEE ALSO

**read**(2), **write**(2), **kerberos**(3N), **kerberos_rpc**(3N), **krb_realmofhost**(3N)

## NOTES

These interfaces are unsafe in multithreaded applications. Unsafe interfaces should be called only from the main thread.

## BUGS

**krb_sendauth( )**, **krb_recvauth( )**, **krb_net_write( )**, and **krb_net_read( )** will not work properly on sockets set to non-blocking I/O mode.

## AUTHOR

John T. Kohl, MIT Project Athena

## RESTRICTIONS