NAME

rpc_secure - library routines for secure remote procedure calls

SYNOPSIS

#include <rpc/rpc.h>

*AUTH **
authdes_create(*char *name, unsigned window, struct sockaddr *addr, des_block *ckey*);

int
authdes_getucred(*struct authdes_cred *adc, uid_t *uid, gid_t *gid, int *grouplen, gid_t *groups*);

int
getnetname(*char *name*);

int
host2netname(*char *name, const char *host, const char *domain*);

int
key_decryptsession(*const char *remotename, des_block *deskey*);

int
key_encryptsession(*const char *remotename, des_block *deskey*);

int
key_gendes(*des_block *deskey*);

int
key_setsecret(*const char *key*);

int
netname2host(*char *name, char *host, int hostlen*);

int
netname2user(*char *name, uid_t *uidp, gid_t *gidp, int *gidlenp, gid_t *gidlist*);

int
user2netname(*char *name, const uid_t uid, const char *domain*);

DESCRIPTION

These routines are part of the RPC library. They implement DES Authentication. See rpc(3) for further details about RPC.

The **authdes_create**() is the first of two routines which interface to the RPC secure authentication system, known as DES authentication. The second is **authdes_getucred**(), below.

Note: the keyserver daemon keyserv(8) must be running for the DES authentication system to work.

The **authdes_create**() function, used on the client side, returns an authentication handle that will enable the use of the secure authentication system. The first argument *name* is the network name, or *netname*, of the owner of the server process. This field usually represents a *hostname* derived from the utility routine **host2netname**(), but could also represent a user name using **user2netname**(). The second field is window on the validity of the client credential, given in seconds. A small window is more secure than a large one, but choosing too small of a window will increase the frequency of resynchronizations because of clock drift. The third argument *addr* is optional. If it is NULL, then the authentication system will assume that the local clock is always in sync with the server's clock, and will not attempt resynchronizations. If an address is supplied, however, then the system will use the address for consulting the remote time service whenever resynchronization is required. This argument is usually the address of the RPC server itself. The final argument *ckey* is also optional. If it is NULL, then the authentication system will generate a random DES key to be used for the encryption of credentials. If it is supplied, however, then it will be used instead.

The **authdes_getucred**() function, the second of the two DES authentication routines, is used on the server side for converting a DES credential, which is operating system independent, into a UNIX credential. This routine differs from utility routine **netname2user**() in that **authdes_getucred**() pulls its information from a cache, and does not have to do a Yellow Pages lookup every time it is called to get its information.

The **getnetname**() function installs the unique, operating-system independent netname of the caller in the fixed-length array *name*. Returns TRUE if it succeeds and FALSE if it fails.

The **host2netname**() function converts from a domain-specific hostname to an operating-system independent netname. Returns TRUE if it succeeds and FALSE if it fails. Inverse of **netname2host**().

The **key_decryptsession**() function is an interface to the keyserver daemon, which is associated with RPC's secure authentication system (DES authentication). User programs rarely need to call it, or its associated routines **key_encryptsession**(), **key_gendes**() and **key_setsecret**(). System commands such as login(1) and the RPC library are the main clients of these four routines.

The **key_decryptsession**() function takes a server netname and a DES key, and decrypts the key by using

the public key of the server and the secret key associated with the effective uid of the calling process. It is the inverse of **key_encryptsession**().

The **key_encryptsession**() function is a keyserver interface routine. It takes a server netname and a des key, and encrypts it using the public key of the server and the secret key associated with the effective uid of the calling process. It is the inverse of **key_decryptsession**().

The **key_gendes**() function is a keyserver interface routine. It is used to ask the keyserver for a secure conversation key. Choosing one "random" is usually not good enough, because the common ways of choosing random numbers, such as using the current time, are very easy to guess.

The **key_setsecret**() function is a keyserver interface routine. It is used to set the key for the effective *uid* of the calling process.

The **netname2host**() function converts from an operating-system independent netname to a domain-specific hostname. Returns TRUE if it succeeds and FALSE if it fails. Inverse of **host2netname**().

The **netname2user**() function converts from an operating-system independent netname to a domain-specific user ID. Returns TRUE if it succeeds and FALSE if it fails. Inverse of **user2netname**().

The **user2netname**() function converts from a domain-specific username to an operating-system independent netname. Returns TRUE if it succeeds and FALSE if it fails. Inverse of **netname2user**().

SEE ALSO

rpc(3), xdr(3), keyserv(8)

The following manuals:

Remote Procedure Calls: Protocol Specification.

Remote Procedure Call Programming Guide.

Rpcgen Programming Guide.

RPC: Remote Procedure Call Protocol Specification, RFC1050, Sun Microsystems Inc., USC-ISI.