NAME

secure_rpc, authdes_getucred, authdes_seccreate, getnetname, host2netname, key_decryptsession, key_encryptsession, key_gendes, key_setsecret, key_secretkey_is_set, netname2host, netname2user, user2netname – library routines for secure remote procedure calls

MT-LEVEL

MT-Safe

DESCRIPTION

RPC library routines allow C programs to make procedure calls on other machines across the network.

RPC supports various authentication flavors. Among them are:

AUTH_NONE	(none) no authentication.
AUTH_SYS	Traditional UNIX-style authentication.
AUTH_DES	DES encryption-based authentication.
AUTH_KERB	Kerberos encryption-based authentication.

The **authdes_getucred( )** and **authdes_seccreate( )** routines implement the **AUTH_DES** authentication flavor. The keyserver daemon **keyserv** (see **keyserv**(1M)) must be running for the **AUTH_DES** authentication system to work, and **keylogin**(1) must have been run. Only the **AUTH_DES** style of authentication is discussed here. For information about the **AUTH_NONE** and **AUTH_SYS** styles of authentication, refer to **rpc_clnt_auth**(3N). For information about the **AUTH_KERB** style of authentication, refer to **kerberos_rpc**(3N).

The routines documented on this page are MT-Safe. See the pages of the other authentication styles for their MT-level.

Routines

See **rpc**(3N) for the definition of the **AUTH** data structure.

#include <rpc/rpc.h>
#include <sys/types.h>

int authdes_getucred(const struct authdes_cred **adc*, **uid_t** **uidp*, **gid_t** **gidp*, **short** **gidlenp*, **gid_t** **gidlist***);**

authdes_getucred( ) is the first of the two routines which interface to the RPC secure authentication system known as **AUTH_DES**. The second is **authdes_seccreate( )**, below. **authdes_getucred( )** is used on the server side for converting an **AUTH_DES** credential, which is operating system independent, into an **AUTH_SYS** credential. This routine returns **1** if it succeeds, **0** if it fails.

uidp* is set to the user's numerical ID associated with *adc*. **gidp* is set to the numerical ID of the user's group. **gidlist* contains the numerical IDs of the other groups to which the user belongs. **gidlenp* is set to the number of valid group ID entries in **gidlist* (see **netname2user( ), below).

Warning: **authdes_getucred( )** will fail if the authdes_cred structure was created with the netname of a host. In such a case, **netname2host( )** should be used on the host netname in the authdes_cred structure to get the host name.

AUTH ∗**authdes_seccreate(const char** ∗*name*, **const unsigned int** *window*,
const char ∗*timehost*, **const des_block** ∗*ckey*);

authdes_seccreate( ), the second of two **AUTH_DES** authentication routines, is used on the client side to return an authentication handle that will enable the use of the secure authentication system. The first parameter *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( )**, described below.

The second field is *window* on the validity of the client credential, given in seconds. If the difference in time between the client's clock and the server's clock exceeds *window*, the server will reject the client's credentials, and the clock will have to be resynchronized. 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 parameter, *timehost*, the host's name, is optional. If it is **NULL**, then the authentication system will assume that the local clock is always in sync with the *timehost* clock, and will not attempt resynchronizations. If a timehost is supplied, however, then the system will consult with the remote time service whenever resynchronization is required. This parameter is usually the name of the host on which the server is running.

The final parameter *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 *ckey* is supplied, then it will be used instead.

If **authdes_seccreate( )** fails, it returns **NULL**.

int getnetname(char *name*[**MAXNETNAMELEN+1]);**

getnetname( ) returns the unique, operating system independent netname of the caller in the fixed-length array *name*. Returns **1** if it succeeds, and **0** if it fails.

int host2netname(char *name*[**MAXNETNAMELEN+1], const char** ∗*host*,
const char ∗*domain*);

Convert from a domain-specific hostname *host* to an operating system independent netname. Returns **1** if it succeeds, and **0** if it fails. Inverse of **netname2host( )**. If *domain* is **NULL**, **host2netname( )** uses the default domain name of the machine. If *host* is **NULL**, it defaults to that machine itself. If *domain* is **NULL** and *host* is a NIS name like "host1.ssi.sun.com," **host2netname( )** uses the domain "ssi.sun.com" rather than the default domain name of the machine.

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

key_decryptsession( ) is an interface to the keyserver daemon, which is associated with RPC's secure authentication system (**AUTH_DES** authentication). User programs rarely need to call it, or its associated routines **key_encryptsession( )**, **key_gendes( )**, and **key_setsecret( )**.

key_decryptsession( ) takes a server netname *remotename* and a DES key *deskey*, and decrypts the key by using the the public key of the the server and the secret key associated with the effective UID of the calling process. It is the inverse of **key_encryptsession( )**.

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

key_encryptsession( ) is a keyserver interface routine. It takes a server netname *remotename* and a DES key *deskey*, and encrypts it using the public key of the the server and the secret key associated with the effective UID of the calling process. It is the inverse of **key_decryptsession( )**. This routine returns **0** if it succeeds, −**1** if it fails.

int key_gendes(des_block **deskey*);

key_gendes( ) is a keyserver interface routine. It is used to ask the keyserver for a secure conversation key. Choosing one at 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. This routine returns **0** if it succeeds, −**1** if it fails.

int key_setsecret(const char **key*);

key_setsecret( ) is a keyserver interface routine. It is used to set the key for the effective UID of the calling process. This routine returns **0** if it succeeds, −**1** if it fails.

int key_secretkey_is_set(void);

key_secretkey_is_set( ) is a keyserver interface routine that may be used to determine whether a key has been set for the effective UID of the calling process. If the keyserver has a key stored for the effective UID of the calling process, this routine returns **1**. Otherwise it returns **0**.

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

Convert from an operating system independent netname *name* to a domain-specific hostname *host*. *hostlen* is the maximum size of *host*. Returns **1** if it succeeds, and **0** if it fails. Inverse of **host2netname( )**.

int netname2user(const char **name*, **uid_t** **uidp*, **gid_t** **gidp*,
int **gidlenp*, **gid_t** *gidlist*[NGROUPS]);

Convert from an operating system independent netname to a domain-specific user ID. Returns **1** if it succeeds, and **0** if it fails. Inverse of **user2netname( )**.

∗*uidp* is set to the user's numerical ID associated with *name*. ∗*gidp* is set to the numerical ID of the user's group. *gidlist* contains the numerical IDs of the other groups to which the user belongs. ∗*gidlenp* is set to the number of valid group ID entries in *gidlist*.

int user2netname(char *name***[MAXNETNAMELEN+1], const uid_t** *uid***,**
const char ∗*domain***);**

Convert from a domain-specific username to an operating system independent netname. Returns **1** if it succeeds, and **0** if it fails. Inverse of **netname2user( )**.

SEE ALSO

chkey(1), **keyserv**(1M), **newkey**(1M), **kerberos_rpc**(3N), **rpc**(3N), **rpc_clnt_auth**(3N)