NAME

kerberos_rpc, authkerb_getucred, authkerb_seccreate, svc_kerb_reg – library routines for remote procedure calls using Kerberos authentication

MT-LEVEL

Unsafe

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 **authkerb_getucred( )**, **authkerb_seccreate( )**, and **svc_kerb_reg( )** routines implement the **AUTH_KERB** authentication flavor. The **kerbd** daemon (see **kerbd**(1M)) must be running for the **AUTH_KERB** authentication system to work for kernel based services such as NFS, and **kinit**(1) must have been run by the user in all cases. Only the **AUTH_KERB** 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_DES** style of authentication, refer to **secure_rpc**(3N).

Routines

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

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

#include <rpc/rpc.h>

#include <sys/types.h>

int authkerb_getucred(const struct svc_rqst **rqst*, **uid_t** **uidp*, **gid_t** **gidp*,
short **gidlenp*, **int** *gidlist*[**NGROUPS]);**

authkerb_getucred( ) is used on the server side for converting an **AUTH_KERB** credential received in an RPC request, 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 numerical ID of the user associated with the RPC request referenced by *rqst*. **gidp* is set to the numerical ID of the user's group. The numerical IDs of the other groups to which the user belongs are stored in *gidlist*[ ]. **gidlenp* is set to the number of valid group ID entries returned in *gidlist*[ ]. All information returned by this routine is based on the Kerberos principal name contained in *rqst*. This principal name is taken to be the login name of the user, and the IDs returned are the same as if that user had physically logged in to the system.

AUTH ***authkerb_seccreate(const char** **service*, **const char** **srv_inst*,
const char **realm*, **const unsigned int** *window*, **const char** **timehost*,
int **status*);

authkerb_seccreate( ) is used on the client side to return an authentication handle that will enable the use of the Kerberos authentication system. The first parameter *service* is the Kerberos principal name of the service to be used. This name is generally a constant with respect to the service being used. *srv_instance* is the instance of the service to be called, and may be NULL to indicate any instance. *realm* is the Kerberos realm name of the desired service. If it is NULL, then the local default realm will be used.

The fourth parameter is the *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 fifth parameter, *timehost*, 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 *status* is also optional. If *status* is supplied, then it will be used to return a Kerberos error status codes if an error occurs. If *status* is NULL, then no detailed error codes will be returned.

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

int svc_kerb_reg(const SVCXPRT **xprt*, **const char** **name*, **const char** **inst*,
const char **realm*);

svc_kerb_reg( ) performs registration tasks in the server which are required before **AUTH_KERB** requests can be processed. *xprt* is the RPC tranport to which this information is to be associated. If *xprt* is NULL then this registration will be effective for any requests arriving on transports that have not been specifically registered.

The other parameters describe the Kerberos principal identity that this server will take on. This must be the same identity that the clients will use when requesting Kerberos tickets for authentication. *name* is the principal name of the service and must be provided. *inst* is the instance. This parameter may be NULL to specify the NULL instance of the service. Most common would be for *inst* to be **"*"** which allows the Kerberos library to determine the correct instance to use, such as the hostname that the service is running on. *realm* is the Kerberos realm name to use in validating tickets. If it is NULL, then the local default realm will be used.

svc_kerb_reg( ) should generally be called immediately before **svc_run( )**. It returns 0 if it succeeds, and −1 if it fails.

SEE ALSO

kerberos(1), **kinit**(1), **kerbd**(1M), **rpc**(3N), **rpc_clnt_auth**(3N), **secure_rpc**(3N)

NOTES

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