NAME

RPCSEC_GSS — GSS-API based authentication for RPC

SYNOPSIS

```
#include <rpc/rpcsec_gss.h>
```

DESCRIPTION

RPCSEC_GSS is a security mechanism for the RPC protocol. It uses the Generic Security Service API (GSS-API) to establish a security context between a client and a server and to ensure that all subsequent communication between client and server are properly authenticated. Optionally, extra protection can be applied to the connection. The integrity service uses checksums to ensure that all data sent by a peer is received without modification. The privacy service uses encryption to ensure that no third party can access the data for a connection.

To use this system, an application must first use **rpc_gss_seccreate**() to establish a security context.

DATA STRUCTURES

Data structures used by **RPCSEC_GSS** appear below.

rpc_gss_service_t

This type defines the types of security service required for **rpc_gss_seccreate**().

```
typedef enum {
        rpc_gss_svc_default   = 0,
        rpc_gss_svc_none      = 1,
        rpc_gss_svc_integrity = 2,
        rpc_gss_svc_privacy   = 3
} rpc_gss_service_t;
```

rpc_gss_options_ret_t

This structure contains various optional values which are used while creating a security context.

```
typedef struct {
        int             req_flags;     /* GSS request bits */
        int             time_req;      /* requested lifetime */
        gss_cred_id_t   my_cred;       /* GSS credential */
        gss_channel_bindings_t input_channel_bindings;
} rpc_gss_options_req_t;
```

rpc_gss_options_ret_t

Various details of the created security context are returned using this structure.

```
typedef struct {
        int             major_status;
        int             minor_status;
        u_int           rpcsec_version;
        int             ret_flags;
        int             time_req;
        gss_ctx_id_t    gss_context;
        char            actual_mechanism[MAX_GSS_MECH];
} rpc_gss_options_ret_t;
```

rpc_gss_principal_t

This type is used to refer to an client principal which is represented in GSS-API exported name form (see gss_export_name(3) for more details). Names in this format may be stored in access control lists or compared with other names in exported name form. This structure is returned by

rpc_gss_get_principal_name() and is also referenced by the *rpc_gss_rawcred_t* structure.

```
typedef struct {
        int             len;
        char            name[1];
} *rpc_gss_principal_t;
```

rpc_gss_rawcred_t

This structure is used to access the raw credentials associated with a security context.

```
typedef struct {
        u_int           version;        /* RPC version number */
        const char      *mechanism;     /* security mechanism */
        const char      *qop;           /* quality of protection */
        rpc_gss_principal_t client_principal; /* client name */
        const char      *svc_principal;/* server name */
        rpc_gss_service_t service;      /* service type */
} rpc_gss_rawcred_t;
```

rpc_gss_ucred_t

Unix credentials which are derived form the raw credentials, accessed via **rpc_gss_getcred**().

```
typedef struct {
        uid_t           uid;            /* user ID */
        gid_t           gid;            /* group ID */
        short           gidlen;
        gid_t           *gidlist;       /* list of groups */
} rpc_gss_ucred_t;
```

rpc_gss_lock_t

Structure used to enforce a particular QOP and service.

```
typedef struct {
        bool_t          locked;
        rpc_gss_rawcred_t *raw_cred;
} rpc_gss_lock_t;
```

rpc_gss_callback_t

Callback structure used by **rpc_gss_set_callback**().

```
typedef struct {
        u_int           program;        /* RPC program number */
        u_int           version;        /* RPC version number */
                                        /* user defined callback */
        bool_t          (*callback)(struct svc_req *req,
                                    gss_cred_id_t deleg,
                                    gss_ctx_id_t gss_context,
                                    rpc_gss_lock_t *lock,
                                    void **cookie);
} rpc_gss_callback_t;
```

rpc_gss_error_t

Structure used to return error information by **rpc_gss_get_error**().

```
typedef struct {
        int             rpc_gss_error;
```

```
        int             system_error;  /* same as errno */
} rpc_gss_error_t;

/*
 * Values for rpc_gss_error
 */
#define RPC_GSS_ER_SUCCESS    0       /* no error */
#define RPC_GSS_ER_SYSTEMERROR        1       /* system error */
```

INDEX

`rpc_gss_seccreate`(3)
Create a new security context

`rpc_gss_set_defaults`(3)
Set service and quality of protection for a context

`rpc_gss_max_data_length`(3)
Calculate maximum client message sizes.

`rpc_gss_get_error`(3)
Get details of the last error

`rpc_gss_mech_to_oid`(3)
Convert a mechanism name to the corresponding GSS-API oid.

`rpc_gss_oid_to_mech`(3)
Convert a GSS-API oid to a mechanism name

`rpc_gss_qop_to_num`(3)
Convert a quality of protection name to the corresponding number

`rpc_gss_get_mechanisms`(3)
Get a list of security mechanisms.

`rpc_gss_get_mech_info`(3)
Return extra information about a security mechanism

`rpc_gss_get_versions`(3)
Return the maximum and minimum supported versions of the **RPCSEC_GSS** protocol

`rpc_gss_is_installed`(3)
Query for the presence of a particular security mechanism

`rpc_gss_set_svc_name`(3)
Set the name of a service principal which matches a given RPC program plus version pair

`rpc_gss_getcred`(3)
Get credential details for the security context of an RPC request

`rpc_gss_set_callback`(3)
Install a callback routine which is called on the server when new security contexts are created

`rpc_gss_get_principal_name`(3)
Create a client principal name from various strings

`rpc_gss_svc_max_data_length`(3)
Calculate maximum server message sizes.

AVAILABILITY

These functions are part of libtirpc.

SEE ALSO

`rpc`(3), `gssapi`(3)

AUTHORS

This manual page was written by Doug Rabson <dfr@FreeBSD.org>.