NAME

krb5_verify_user, **krb5_verify_user_lrealm**, **krb5_verify_user_opt**, **krb5_verify_opt_init**, **krb5_verify_opt_alloc**, **krb5_verify_opt_free**, **krb5_verify_opt_set_ccache**, **krb5_verify_opt_set_flags**, **krb5_verify_opt_set_service**, **krb5_verify_opt_set_secure**, **krb5_verify_opt_set_keytab** - Heimdal password verifying functions

LIBRARY

Kerberos 5 Library (libkrb5, -lkrb5)

SYNOPSIS

#include <krb5.h>

krb5_error_code
krb5_verify_user(*krb5_context context, krb5_principal principal, krb5_ccache ccache, const char *password, krb5_boolean secure, const char *service*);

krb5_error_code
krb5_verify_user_lrealm(*krb5_context context, krb5_principal principal, krb5_ccache ccache, const char *password, krb5_boolean secure, const char *service*);

void
krb5_verify_opt_init(*krb5_verify_opt *opt*);

void
krb5_verify_opt_alloc(*krb5_verify_opt **opt*);

void
krb5_verify_opt_free(*krb5_verify_opt *opt*);

void
krb5_verify_opt_set_ccache(*krb5_verify_opt *opt, krb5_ccache ccache*);

void
krb5_verify_opt_set_keytab(*krb5_verify_opt *opt, krb5_keytab keytab*);

void
krb5_verify_opt_set_secure(*krb5_verify_opt *opt, krb5_boolean secure*);

void
krb5_verify_opt_set_service(*krb5_verify_opt *opt, const char *service*);

void
krb5_verify_opt_set_flags(*krb5_verify_opt *opt, unsigned int flags*);

krb5_error_code
krb5_verify_user_opt(*krb5_context context, krb5_principal principal, const char *password, krb5_verify_opt *opt*);

DESCRIPTION

The **krb5_verify_user** function verifies the password supplied by a user. The principal whose password will be verified is specified in *principal*. New tickets will be obtained as a side-effect and stored in *ccache* (if NULL, the default ccache is used). **krb5_verify_user**() will call **krb5_cc_initialize**() on the given *ccache*, so *ccache* must only initialized with **krb5_cc_resolve**() or **krb5_cc_gen_new**(). If the password is not supplied in *password* (and is given as NULL) the user will be prompted for it. If *secure* the ticket will be verified against the locally stored service key *service* (by default 'host' if given as NULL).

The **krb5_verify_user_lrealm**() function does the same, except that it ignores the realm in *principal* and tries all the local realms (see krb5.conf(5)). After a successful return, the principal is set to the authenticated realm. If the call fails, the principal will not be meaningful, and should only be freed with krb5_free_principal(3).

krb5_verify_opt_alloc() and **krb5_verify_opt_free**() allocates and frees a krb5_verify_opt. You should use the the alloc and free function instead of allocation the structure yourself, this is because in a future release the structure wont be exported.

krb5_verify_opt_init() resets all opt to default values.

None of the krb5_verify_opt_set function makes a copy of the data structure that they are called with. It's up the caller to free them after the **krb5_verify_user_opt**() is called.

krb5_verify_opt_set_ccache() sets the *ccache* that user of *opt* will use. If not set, the default credential cache will be used.

krb5_verify_opt_set_keytab() sets the *keytab* that user of *opt* will use. If not set, the default keytab will be used.

krb5_verify_opt_set_secure() if *secure* if true, the password verification will require that the ticket will be verified against the locally stored service key. If not set, default value is true.

krb5_verify_opt_set_service() sets the *service* principal that user of *opt* will use. If not set, the 'host'

service will be used.

krb5_verify_opt_set_flags() sets *flags* that user of *opt* will use. If the flag KRB5_VERIFY_LREALMS is used, the *principal* will be modified like **krb5_verify_user_lrealm**() modifies it.

krb5_verify_user_opt() function verifies the *password* supplied by a user. The principal whose password will be verified is specified in *principal*. Options the to the verification process is pass in in *opt*.

EXAMPLES

Here is a example program that verifies a password. it uses the ‘host/‘hostname‘’ service principal in *krb5.keytab*.

```
#include <krb5.h>

int
main(int argc, char **argv)
{
   char *user;
   krb5_error_code error;
   krb5_principal princ;
   krb5_context context;

   if (argc != 2)
           errx(1, "usage: verify_passwd <principal-name>");

   user = argv[1];

   if (krb5_init_context(&context) < 0)
           errx(1, "krb5_init_context");

   if ((error = krb5_parse_name(context, user, &princ)) != 0)
           krb5_err(context, 1, error, "krb5_parse_name");

   error = krb5_verify_user(context, princ, NULL, NULL, TRUE, NULL);
   if (error)
      krb5_err(context, 1, error, "krb5_verify_user");

   return 0;
}
```

SEE ALSO

krb5_cc_gen_new(3), krb5_cc_initialize(3), krb5_cc_resolve(3), krb5_err(3), krb5_free_principal(3), krb5_init_context(3), krb5_kt_default(3), krb5.conf(5)