

**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 be 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 *alloc* and *free* function instead of allocating the structure yourself, this is because in a future release the structure won't be exported.

**krb5\_verify\_opt\_init()** resets all *opt* to default values.

None of the *krb5\_verify\_opt\_set* functions make a copy of the data structure that they are called with. It's up to 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* is 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 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)