NAME

getuserent, putuserent, setuserent, enduserent, getuserentbyname, getuserentbyuid, free_userent – Get Trusted Solaris user security attributes

SYNOPSIS

cc [*flag* ...] *file* ... **–ltsoldb –ltsol –lnsl –lcmd** [*library* ...]

#include <tsol/user.h>

userent_t *getuserentbyname(char *user, int src);

userent_t *getuserentbyuid(uid_t uid, int src);

userent_t *getuserent(int src);

void setuserent(int *stayopen*, **int** *src***);**

void enduserent(int *src***);**

void free_userent(userent_t ***userent*);**

AVAILABILITY

SUNWtsolu

MT-LEVEL

MT-Safe

DESCRIPTION

These functions are used to obtain entries describing Trusted Solaris user attributes from the **tsoluser** NIS+ database.

getuserbyname() searches for information for a user with the specified user name *user*. **getuserbyuid**() searches for information for a user with the specified user ID *uid*.

The functions **setuserent**(), **getuserent**(), and **enduserent**() are used to list user entries from the database. **setuserent** sets (or resets) the enumeration to the beginning of the set of Trusted Solaris user entries. This function should be called before the first call to **getuserent**. A call to **getuserbyname**() or **getuserbyuid**() leaves the list position in an indeterminate state. If the *stayopen* flag is not zero, the system may keep allocated resources such as open file descriptors until a subsequent call to **enduserent**.

Successive calls to **getuserent** return either successive entries or **NULL**, which indicates the end of the list.

enduserent may be called when the caller expects to do no further user-entry-retrieval operations; the system may then deallocate resources it was using. It is still allowed, but possibly less efficient, for the process to call more user-entry-retrieval functions after calling **enduserent**.

The functions **getuserentbyname**, **getuserentbyuid**, and **getuserent** are reentrant interfaces that allocate memory to store returned results, and are safe for use in both single-threaded and multithreaded applications. The function **free_userent**() is used to release memory allocated by these two functions. The parameter *user*, a pointer to the user name, must be a null-terminated character string. The parameter *uid* must be a **uid_t**.

The parameter *src* may be set to **TSOL_DB_SRC_FILES**, **TSOL_DB_SRC_NISPLUS**, or **TSOL_DB_SRC_SWITCH**, which are defined in <**tsol/tsol.h**>. Currently the *src* parameter is always treated as **TSOL_DB_SRC_NISPLUS**, forcing all information to come out of the NIS+ table, regardless of which *src* is actually requested. Use of *files* or the *nsswitch* may be supported in future releases. Developers may wish use the **TSOL_DB_SRC_NISPLUS** parameter until the other parameters are supported.

For listing in multithreaded applications, the position within the list is a processwide property shared by all threads. **setuserent** may be used in a multithreaded application but resets the list position for all threads. If multiple threads interleave calls to **getuserent**, the threads will list disjoint subsets of the **tsoluser** database.

RETURN VALUES

User entries are represented by the **struct userent_t** structure defined in <**tsol/user.h**> **:**

```
typedef struct userent_s {
        char *  name;           /* user associated with this entry */
        char *  lock;           /* is account locked? */
        char *  badlogins;      /* how many failed login attempts so far */
        char *  generation;     /* method of password generation */
        char *  profiles;       /* user profiles used */
        char *  roles;          /* roles assumable */
        char *  idletime;       /* minutes a workstation may remain idle*/
        char *  idlecmd;        /* what to do at when idletime reached */
        char *  labelview;      /* can user see ADMIN_HI and ADMIN_LOW labels*/
        char *  labeltrans;     /* process security attributes for label translation*/
        char *  labelmin;       /* allowed low login labels */
        char *  labelmax;       /* allowed high login labels */
        char *  usertype;       /* normal, admin-role, or non-admin-role */
        char *  res1;           /* reserved for future use*/
        char *  res2;           /* reserved for future use*/
        char *  res3;           /* reserved for future use*/
userent_t };
```

If it successfully locates the requested entry, **getuserentbyname** returns a pointer to a **userent_t**. If unsuccessful, **getuserentbyname** returns **NULL.**

If it successfully lists an entry, **getuserent** returns a pointer to a **struct userent_t**. If unsuccessful, **getuserent** returns **NULL**, indicating the end of the list.

Upon success, **putuserent**, **setuserent**, and **enduserent** return **0**.

ERRORS

The functions **getuserentbyname**, **getuserentbyuid**, and **getuserent** return **NULL** on failure.

NOTES

Programs that use the interfaces described in this manual page cannot be linked statically because the implementations of these functions employ dynamic loading and linking of shared objects at run time.

When compiling multithreaded applications, see *Notes on Multithread Applications* in **Intro(3)** for information about the use of the **_REENTRANT** flag.

These interfaces are uncommitted; although not expected to change between minor releases of Trusted Solaris systems, these interfaces may change.