NAME

authuser − remote authentication library using the Authentication Server

SYNTAX

#include <authuser.h>

unsigned short **auth_tcpport**;

char ***auth_xline**(*user,fd,&inremote*);

int **auth_fd**(*fd,&inremote,&local,&remote*);
int **auth_fd2**(*fd,&inlocal,&inremote,&local,&remote*);

int **auth_tcpsock**(*fd,inlocal,inremote*);

char ***auth_tcpuser**(*inremote,local,remote*);
char ***auth_tcpuser2**(*inlocal,inremote,local,remote*);
char ***auth_tcpuser3**(*inlocal,inremote,local,remote,timeout*);

char ***auth_sockuser**(*s,local,remote*);

char **user*;
int *fd*;
int *s*;
unsigned long *inlocal*;
unsigned long *inremote*;
unsigned short *local*;
unsigned short *remote*;
int *timeout*;

DESCRIPTION

Actually should talk about 'identification', there is no definite authentication in here..

The **authuser** library provides a simple interface for finding out the remote identity of a connection through the Authentication Server as specified by RFC 931. Use the -lauthuser loader option to compile a program with this library.

auth_xline(*user,fd,&inremote*) returns a line of the form X-Auth-User: user or X-Forgery-By: username, depending upon what the host on the other side of *fd* thinks of the user. This is particularly appropriate for mail and news headers.

If the remote host reports that *user* owns the connection on that side, **auth_xline** will return X-Auth-User: user. If the remote host reports that a different username owns the connection, **auth_xline** will return X-Forgery-By: username. If user is NULL, it returns X-Auth-User: username with the username reported by the remote host. If *fd* is not a TCP connection or authentication is impossible, **auth_xline** returns NULL, setting errno appropriately.

The line is not cr-lf terminated. It is stored in a static area which is overwritten on each call to **auth_xline.** **auth_xline** places the Internet address of the other host into *inremote.*

auth_fd2(*fd,&inlocal,&inremote,&local,&remote*) retrieves address information from the connection in socket *fd.* It places the Internet addresses of the connection into *inlocal* and *inremote* and the local and remote TCP ports into *local* and *remote.* **auth_fd2** returns -1 upon error, setting errno appropriately.

auth_tcpuser2(*inlocal,inremote,local,remote*) returns the name of the user on the other end of the TCP connection between *remote@inremote* and *local@inlocal.* If authentication is impossible,

auth_tcpuser2 returns NULL, setting errno appropriately. The user name is stored in a static area which is overwritten on each call to **auth_tcpuser2, auth_tcpuser, auth_sockuser,** and **auth_xline.**

s = **auth_tcpsock(***fd,inlocal,inremote***)** sets *s* to a non-blocking socket which is connecting to the Authentication Server at *inremote.* It returns -1 on error, setting errno appropriately. **auth_sockuser(***s,local,remote***)** makes sure that the socket has connected and then does the same job as **auth_tcpuser2,** returning the name of the user on the other end of the TCP connection between *remote@inremote* and *local@inlocal,* or NULL (with errno set) if authentication is not possible. *s* is closed by **auth_sockuser.** The advantage of using **auth_tcpsock** and **auth_sockuser** instead of **auth_tcpuser2** is that you can perform other actions while waiting for the authentication request to complete. You can select *s* for writing to see if it is ready for **auth_sockuser** yet.

auth_tcpuser3(*inlocal,inremote,local,remote,timeout***)** is like **auth_tcpuser2** but returns NULL with errno set to ETIMEDOUT if the authentication request has not been accepted or refused after *timeout* seconds.

auth_fd(*fd,&inremote,&local,&remote***)** is the same as **auth_fd2** but throws away the *inlocal* information. **auth_tcpuser(** *inremote,local,remote***)** is the same as **auth_tcpuser2** but may not bind to the proper local address on hosts with multiple IP addresses. These functions do not perform properly on multihomed hosts and should not be used. They are provided only for backwards compatibility.

The authentication routines check with the remote Authentication Server on port **auth_tcpport,** which defaults to 113 as specified by RFC 931. You can set **auth_tcpport** to other values for nonstandard implementations.

RESTRICTIONS

authuser does no backslash interpretation upon the remote user name. This is conformance with the proposed revision to RFC 931.

authuser does not use the operating system type information provided by the Authentication Server.

VERSION

authuser version 4.0, February 9, 1992.

AUTHOR

Placed into the public domain by Daniel J. Bernstein.

REFERENCES

The authentication server is more secure than passwords in some ways, but less secure than passwords in many ways. (It's certainly better than no password at all---e.g., for mail or news.) It is not the final solution. For an excellent discussion of security problems within the TCP/IP protocol suite, see Steve Bellovin's article "Security Problems in the TCP/IP Protocol Suite."

SEE ALSO

tcpclient(1), *tcpserver*(1), *getpeername*(3), *getsockname*(3), *tcp*(4), *authd*(8)