NAME

hosts_access, hosts_ctl, request_init, request_set - access control library

SYNOPSIS

```
#include <tcpd.h>

extern int allow_severity;
extern int deny_severity;

struct request_info *request_init(request, key, value, ..., 0)
struct request_info *request;

struct request_info *request_set(request, key, value, ..., 0)
struct request_info *request;

void fromhost(request)
struct request_info *request;

int hosts_access(request)
struct request_info *request;

int hosts_ctl(daemon, client_name, client_addr, client_user)
char *daemon;
char *client_name;
char *client_addr;
char *client_user;
```

DESCRIPTION

The routines described in this document are part of the *libwrap.a* library. They implement a rule-based access control language with optional shell commands that are executed when a rule fires.

request_init() initializes a structure with information about a client request. request_set() updates an already initialized request structure. Both functions take a variable-length list of key-value pairs and return their first argument. The argument lists are terminated with a zero key value. All string-valued arguments are copied. The expected keys (and corresponding value types) are:

RQ_FILE (int)
: The file descriptor associated with the request.

RQ_CLIENT_NAME (char *)
: The client host name.

RQ_CLIENT_ADDR (char *)
: A printable representation of the client network address.

RQ_CLIENT_SIN (struct sockaddr_in *)
: An internal representation of the client network address and port. The contents of the structure are not copied.

RQ_SERVER_NAME (char *)
: The hostname associated with the server endpoint address.

RQ_SERVER_ADDR (char *)
: A printable representation of the server endpoint address.

RQ_SERVER_SIN (struct sockaddr_in *)
: An internal representation of the server endpoint address and port. The contents of the structure are not copied.

RQ_DAEMON (char *)
: The name of the daemon process running on the server host.

RQ_USER (char *)
: The name of the user on whose behalf the client host makes the request.

hosts_access() consults the access control tables described in the *hosts_access(5)* manual page. When internal endpoint information is available, host names and client user names are looked up on demand, using the request structure as a cache. hosts_access() returns zero if access should be denied. fromhost() must be called before hosts_access().

hosts_ctl() is a wrapper around the request_init() and hosts_access() routines with a perhaps more convenient interface (though it does not pass on enough information to support automated client username lookups). The client host address, client host name and username arguments should contain valid data or STRING_UNKNOWN. hosts_ctl() returns zero if access should be denied.

The *allow_severity* and *deny_severity* variables determine how accepted and rejected requests may be logged. They must be provided by the caller and may be modified by rules in the access control tables.

DIAGNOSTICS

Problems are reported via the syslog daemon.

SEE ALSO

hosts_access(5), format of the access control tables. hosts_options(5), optional extensions to the base language.

FILES

/etc/hosts.allow, /etc/hosts.deny, access control tables.

BUGS

hosts_access() uses the strtok() library function. This may interfere with other code that relies on strtok().

AUTHOR

Wietse Venema (wietse@wzv.win.tue.nl)
Department of Mathematics and Computing Science
Eindhoven University of Technology
Den Dolech 2, P.O. Box 513,
5600 MB Eindhoven, The Netherlands