The [**TCP**](javascript:void(0);) Wrappers package is installed by default on Fedora Linux and pro- vides host-based security separate from that provided by a firewall running on [**the server**](javascript:void(0);) itself or elsewhere.

The TCP wrappers package (tcp\_wrappers) is installed by default and provides host-based access control to network services. The most important component within the package is the /usr/lib/libwrap.a library. In general terms, a TCP wrapped service is one that has been compiled against the libwrap.a library.

When a connection attempt is made to a TCP wrapped service, the service first references the *hosts access* files (/etc/hosts.allow and /etc/hosts.deny) to determine whether or not the [**client**](javascript:void(0);) host is allowed to connect. In most cases, it then uses the syslog daemon (syslogd) to write the name of the requesting host and the requested service to /var/log/secure or /var/log/messages.

If a client host is allowed to connect, TCP wrappers release control of the connection to the requested service and do not interfere further with communication between the client host and the server.

In addition to access control and logging, TCP wrappers can activate commands to interact with the client before denying or releasing control of the connection to the requested network service.

Because TCP wrappers are a valuable addition to any server administrator’s arsenal of security tools, most network services within Red Hat Enterprise Linux are linked against the libwrap.a library. Some such applications include /usr/sbin/sshd,/usr/sbin/sendmail, and /usr/sbin/xinetd.

**Advantages of TCP Wrappers**

TCP wrappers provide the following advantages over other network service control techniques:

* *Transparency to both the client host and the wrapped network service* — Both the connecting client and the wrapped network service are unaware that TCP wrappers are in use. Legitimate users are logged and connected to the requested service while connections from banned clients fail.
* *Centralized management of multiple protocols* — TCP wrappers operate separately from the network services they protect, allowing many [**server applications**](javascript:void(0);) to share a common set of configuration files for simpler management.

The application relies on two main files:

* /etc/hosts.allow: Defines the hosts and networks allowed to connect to the server. The TCP Wrappers enabled application searches this file for a matching entry, and if it finds one, then the connection is allowed.
* /etc/hosts.deny: Defines the hosts and networks prohibited from con- necting to the server. If a match is found in this file, the connection is denied. No match means the connection proceeds normally.The /etc/hosts.allow file is always read first and both files are always read from top to bottom, therefore the ordering of the entries is important.The TCP Wrappers File Format

The format of the file is:

<TCP-daemon-name> <client-list> : <option>

This example allows all traffic from the 192.168.1.0/24 and the 192.168.2.0/255.255.255.0 networks and [**SSH**](javascript:void(0);) from only two hosts, 172.16.1.1 and 216.10.119.244. All HTTP Web traffic is allowed. All other TCP traffic to [**the host**](javascript:void(0);) is denied. Notice how the subnet masks can use the slash nomencla- ture or the dotted decimal 255.255.255.0 format.

#

# File: hosts.allow

#

ALL: 192.168.1.0/24 192.168.2.0/255.255.255.0

sshd: 172.16.1.1 216.10.119.244

httpd: ALL

#

# File: hosts.deny

#

ALL: ALL

Determining the TCP Daemon’s Name

The easiest way of determining the name of a daemon is to use the ps com- mand and then use grep to filter for the name of the service. Here, the example quickly determines the daemon name (/usr/sbin/sshd) for the [**SSH server**](javascript:void(0);) process. Because TCP Wrappers only requires the program name and not the path, sshd therefore becomes the entry to place in the TCP-daemon-name column of the configuration file.

[root@server tmp]# ps -ef | grep -i ssh

root 10053 1 0 Nov06 ?

root 14145 10053 0 Nov13 ?

root 18100 14148 0 21:56 pts/1

[root@bigboy tmp]#

00:00:00 /usr/sbin/sshd

00:00:02 sshd: root@pts/1

00:00:00 grep ssh

For a full explanation of all the options available, refer to section 5 of the man pages for hosts\_access:

[root@server tmp]# man 5 hosts\_access

TCP Wrappers is simple to implement, but you have to set them on every host. Management is usually easier on a firewall that protects the entire network.

Wildcards allow TCP wrappers to more easily match groups of daemons or hosts. They are used most frequently in the client list field of access rules.

The following wildcards may be used:

* ALL — Matches everything. It can be used for both the daemon list and the client list.
* LOCAL — Matches any host that does not contain a period (.), such as localhost.
* KNOWN — Matches any host where the hostname and host address are known or where the user is known.
* UNKNOWN — Matches any host where the hostname or host address are unknown or where the user is unknown.
* PARANOID — Matches any host where the hostname does not match the host address.

Expansions, when used in conjunction with the spawn and twist directives, provide information about the client, server, and processes involved.

Below is a list of supported expansions:

* %a — Supplies the client’s [**IP address**](javascript:void(0);).
* %A — Supplies the server’s IP address.
* %c — Supplies a variety of client information, such as the username and hostname, or the username and IP address.
* %d — Supplies the daemon process name.
* %h — Supplies the client’s hostname (or IP address, if the hostname is unavailable).
* %H — Supplies the server’s hostname (or IP address, if the hostname is unavailable).
* %n — Supplies the client’s hostname. If unavailable, unknown is printed. If the client’s hostname and host address do not match, paranoid is printed.
* %N — Supplies the server’s hostname. If unavailable, unknown is printed. If the server’s hostname and host address do not match, paranoid is printed.
* %p — Supplies the daemon process ID.
* %s —Supplies various types of server information, such as the daemon process and the host or IP address of the server.
* %u — Supplies the client’s username. If unavailable, unknown is printed.