

TCP Wrapper: do I Need One?

by Boris Loza, PhD

The brave new world of the Internet has brought many dangers as well as opportunities. Security is now a bigger concern than ever, with hackers creating viruses daily. It makes sense to protect your computing resources every way you can, and TCP_wrapper is one of these tools that can help you. This article will explore what TCP_wrapper can do for you.

Understanding TCP Wrapper

The TCP_wrapper is a tiny utility program written by Wietse Venema that allows you to control access to Internet services such as ftp, telnet, rlogin, tftp, finger, rsh, and systat. When the service is requested, TCP_wrapper decides first whether to allow or deny access for this request.

TCP_wrapper is based on the approach that TCP/IP networks use a client/server model. For instance, when you decide to log in to a remote system using the rlogin command, it sends a request to the rlogind process on the server. This request goes through the inetd daemon, which, in turn, checks the /etc/services file for the name of a particular service and the port number. Finally, the /etc/inet/inetd.conf is searched for the name of the program or daemons providing the service.

If you type rlogin one.host.com, the rlogin client on your machine sends a connection request to the server one.host.com with the information about the user and terminal. Because rlogin uses the TCP port 513, inetd looks up this port in /etc/services and finds the service name login. Looking up login in /etc/inet/inetd.conf inetd finds that it needs to run the daemon in.rlogind. After in.rlogind responds to the client, the rlogin session is started.

If you want to control network access to your machine, you can use TCP_wrapper. Instead of directly running the in.rlogind daemon, inetd runs the TCP_wrapper program to perform the source and the destination address checks to decide whether this particular host and user are allowed to communicate with your machine. Finally, it writes a note in the system log about this connection.

Finding and installing the program

A copy of TCP_wrapper can be obtained by ftp at ftp://ftp.cert.org/pub/tools/TCP Wrappers/TCP Wrappers_7.6.tar.gz.

There are two ways to install the TCP_wrapper program: easy and advanced. For the easy installation, move your network daemons to some other directory and fill the resulting holes with copies of the TCP_wrapper programs. This approach doesn't require any changes in the /etc/inet/inetd.conf file. For the advanced installation, leave the network daemons alone and modify the inetd.conf file. In this article, we'll use the advanced method.

First, unpack tcp_wrappers_7.6.tar.gz into the desired directory. Go to the tcp_wrappers_7.6 directory and run the make command with the following options:

```
#make REAL_DAEMON_DIR=/usr/sbin sunos5
```

If you have a GNU gcc compiler, you may want to specify:

```
#make CC=gcc REAL_DAEMON_DIR=/usr/sbin sunos5
```

This will tell TCP_wrapper that all Internet daemons (in.telnetd, in.ftpd, in.fingerd, in.rlogind, etc) are located in the /usr/sbin directory, the C compiler is GNU gcc, and the operating system is Solaris. Be sure to check the default compile options in the Makefile first. For instance, the PARANOID compile option tells TCP_wrapper to always attempt to look up and double check the client host name. It will always refuse service in case of a host name/address discrepancy. For more options, read the Makefile.

Configuring TCP_wrapper

Now you need to modify the `inetd.conf` file to tell the `inetd` daemon that `TCP_wrapper` is in use. Find the service you would like to control and advise it to use `tcpd`. In the following example, we're going to make `ftp`, `telnet`, and `login` services use `TCP_wrapper`. We will modify the following lines in the `/etc/inetd.conf` file:

```
ftp stream tcp nowait root /usr/sbin/in.ftpd in.ftpd
telnet stream tcp nowait root /usr/sbin/in.telnetd in.telnetd
login stream tcp nowait root /usr/sbin/in.telnetd in.rlogind
```

Put in the location of the `tcpd` daemon, instead of the location of the service daemon:

```
ftp stream tcp nowait root /usr/etc/tcpd in.ftpd
telnet stream tcp nowait root /usr/etc/tcpd in.telnetd
telnet stream tcp nowait root /usr/etc/tcpd in.rlogind
```

Don't forget to send `kill -HUP` to the `inetd` process to make the changes effective.

Now you have to deal with two `TCP_wrapper` configuration files: `/etc/hosts.allow` and `/etc/hosts.deny`. You need the `/etc/hosts.allow` file to allow connections to your computer and the `/etc/hosts.deny` file to deny access. If you'd like to allow access to all Internet services on your machine to everyone, leave the `/etc/hosts.allow` and the `/etc/hosts.deny` files empty. This configuration is known as `Mostly Open`.

Assume you want to allow `one.trust.host` access through `telnet` only. In this scenario, you have to modify the `/etc/hosts.allow` file as follows:

```
ALL : LOCAL
in.telnetd : one.trust.host
```

and the `/etc/hosts.deny` file:

```
ALL : ALL
```

The first line in the `/etc/hosts.allow` tells the `tcpd` daemon that an access is allowed for all machines in our local domain. Second, only `telnet` service is allowed from the `one.trust.host`. The `/etc/hosts.deny` file tells the `tcpd` daemon to close all services for any other machine. In `hosts.allow`, we list only those specific services we want others to use. The pound sign (`#`) could be used as a comment.

One more example: suppose you need to deny access to anyone not in your `/etc/hosts` file to `telnet` and `rlogin` services. The `hosts.deny` file would be:

```
in.telnetd in.rlogind: UNKNOWN
```

We use the `TCP_wrapper`'s wildcard `UNKNOWN` that matches any IP address that doesn't have a corresponding hostname. Other wildcards that can replace specific host names are `KNOWN` and `PARANOID`. `KNOWN` matches any IP address that has a corresponding hostname, and `PARANOID` matches any host whose name doesn't match its Internet address. For the complete list of wildcards, see man pages for `hosts_access(5)`.

After creating rules, you can check the configuration files with the `tcpdchk` and `tcpdmatch` commands. The first one checks the configuration files for any problems. It will tell if you've used

wildcards incorrectly, if there are non-corresponded host names in the access file, and much more. For example:

```
#tcpdchk -v
Using network configuration file: /etc/inet/inetd.conf
>>> Rule /etc/hosts.allow line 1:
daemons: ALL
clients: LOCA
warning: /etc/hosts.allow, line 1: LOCA: host not found
access: granted
>>> Rule /etc/hosts.deny line 1:
daemons: in.telnetd in.rlogind
clients: UNKNOWN
access: denied
```

In this example the wildcard LOCAL in the /etc/hosts.allow file is misspelled. The tcpdchk utility expects it to be a host name, but can't find this name in the /etc/hosts file.

The tcpdmatch utility tests the configuration files against a virtual request for an Internet connection. You have to provide the name of the daemon and a host name, and it tells you whether that connection would be allowed or denied:

```
#tcpdmatch in.rlogind one.untrust.host

client: address one.untrusted.host
server: process in.rlogind
matched: /etc/hosts.deny line 1
access: denied
```

For more details, see the man pages for TCP_wrapper.

What else can I do with it?

Another useful feature of TCP_wrapper is that it logs every request for a connection, whether it's granted access or not. By default, the wrapper logs go to the same place as the transaction logs of the sendmail daemon. So, if your mail messages are written into the /var/log/syslog file, you can grep this file for telnetd or rlogind messages to check who has been trying to connect.

In addition, you can run shell commands using "booby traps" when certain services are requested. Assume that you want to know exactly who did a telnet into your machine. In this scenario, add the following line into the /etc/hosts.deny file:

```
in.telnetd: ALL: (/usr/sbin/safe_finger -l \@%h | /usr/ucb/mail -s %d-%h root) &
```

The safe_finger command comes with TCP_wrapper, and as the TCP_wrapper README says, it "gives better protection against nasty stuff that remote hosts may do in response to your finger probes." The strings %h and %d are called expansions, and tcpd replaces them with the corresponding text for the host name and daemon file and sends to root. There are other expansions that can be used in booby traps (see man pages for hosts_access(5)).

Instead of specifying a shell command that should be executed, TCP_wrapper allows you to specify a set of options. To use this feature, you have to compile the TCP_wrapper with the option DPROCESS_OPTIONS. After doing this, you can have in the configuration files a new format of the access control rules:

```
daemon : host : option : option ...
```

Options include allow, deny, and many others.

Suppose you wish to deny all connections to your server, except from the host my.host.com. You just need to put two lines into the hosts.allow file:

```
telnetd,rlogind : my.host.com : allowall : all : deny
```

In this case, we can get rid of the hosts.deny file completely. You can find more information about these features in the hosts_options(5) man page.

Should I use TCP_wrapper?

TCP_wrapper is a very powerful program that could arm your system with a good access control solution. If you use the secure shell (ssh) it comes with Wietse Venema's TCP_wrapper package. If ssh is compiled with the TCP_wrapper, you can control the basic ssh connections, X11 forwarding, and TCP port forwarding.

As useful as TCP_wrapper sounds, keep in mind that it has some limitations. First of all, TCP_wrapper can't protect services started at boot time and run until system shutdown, like sendmail and httpd. Additionally, it's vulnerable to IP spoofing because it uses the IP address for authentication. (The solution for this is to use ssh or to keep your /etc/hosts file up to date and don't rely on outside DNS). But in spite of these limitations, we have found the TCP_wrapper to be extremely useful for securing our networked environment.