Tighten your network security with TCP Wrappers

KEEP IT UNDER WRAPS

Exclusive nightclubs don't let just anyone wander in from the street and the same should be true for your system. David Tansley shows you how to enforce a strict door policy with TCP Wrappers, the meanest bouncer in town

Most nightclubs these days have door staff to restrict access to certain types of clientele. Not only will there be age restrictions, but a dress code may also come into the equation: no white socks or trainers, for example.

Just like a doorman, you can restrict access to your computers based on certain criteria. Firewalls might automatically spring to mind but this month we're going to be looking at a utility called TCPD, more commonly known as TCP Wrappers.

What is TCP Wrappers

TCP Wrappers is installed by default on most Linux boxes and it can also be built on practically all UNIXes as well. What it actually does is wrap itself around all incoming TCP connections, that is TCP daemons that are controlled via xinetd (or inetd, if you haven't yet moved over to xinetd).

When a TCP connection is made on your system, TCP Wrappers (TCPD) is run instead of the required daemon. For instance, if a user connects to your system via FTP, TCPD is invoked rather than the in.ftpd daemon. TCPD will then look at two files: */etc/hosts.allow* and */etc/hosts.deny*, which – as their names suggest – either allow or deny connections based on rules or patterns. Once TCPD has read these files and found a match, the relevant connection will either be granted or denied.

If the connection is allowed, TCPD then writes to syslog – the system messages file – and hands over control to the real daemon that was called, in.ftpd in our example. TCPD's work is now done, and will sleep until the next connection is invoked through xinted. If the connection is denied, i.e. it fails due to the access rules or a pattern match in the hosts.allow or host.deny file, a message is written to syslog, logging this failure attempt. The connection is then broken and TCPD goes back to sleep awaiting the next connection.

Some of the most popular TCP daemons are: telnet, ftp, shell, rdate, tftp, talk. The rule here is if it is TCP and is invoked from xinetd then you can control access to that service from outside connections.

Getting xinetd to recognise TCP Wrappers

Although TCP Wrappers is installed by default on most Linux systems you will need to tell the xinetd daemon that TCPD is there if you wish to use its services. Generally speaking all the TCP/UDP daemons controlled by xinetd house their configuration files in the */etc/xinetd.d* directory. However, this is governed by the *includedir* entry in the */etc/xinetd.conf* file, so check this out first if you don't have an */etc/xinetd.d* directory. You may find that all the configurations are stuck in the actual xinetd.conf file.

You will need to change every service configuration file where you want TCP Wrappers to handle the connections. For Telnet, you would have an entry like in Listing 1. This shows the Telnet services configuration file – your Telnet service file will probably be slightly different.

Notice the use of the flags and server entries in the Telnet service configuration file the use of the entries, these tell xinetd that it is to call TCPD first, the server_args is the actual daemon to run after TCPD has finished. Make the same sort of changes for the rest of the TCP services files you wish to protect.

After making changes use the service command to restart xinetd:

```
$ /sbin/service xinetd restart
```

Or alternately:

```
$ /etc/rc.d/init.d/xinetd restart
```

Those access files

When a connection is initially established TCP Wrappers will look first look in /etc/hosts.allow before checking /etc/hosts.deny, if there is a pattern match

then access will be denied or allowed. Confused? Don't be, the general rule of thumb here is to allow access unless otherwise specified. In other words, keep it simple.

When TCP Wrappers has been enabled via the services configuration files, if neither the hosts.allow or hosts.deny file exist then TCP Wrappers will deny access to everybody, except connections from the localhost, (the actual Linux system where TCP Wrappers is running). All connections are logged via syslog to either /var/log/messages or /var/log/secure, depending on your TCP Wrappers installation.

The general format of the rules or patterns for both files is:

```
daemon_list : client_list : [Shell
Commands][Banners]
```

Where both Shell Commands and Banners are optional. We'll take a look at banners later in the article.

The daemon list is the names of the daemons you wish to allow or deny. The client list is host names, IP Addresses or domain names you wish to allow or deny. To specify multiple daemons or clients use a comma to separate the entries. You can also use wildcards to specify daemons or clients. For instance:

- ALL will match every daemon or every client list
- LOCAL will match the local host only – any host that does not have a '.' in the name
- . (that's a dot) will match anything, a bit like the * in the bash shell. For example, .boo.com, will match any domain that ends in boo.com

When making changes to the hosts.deny or hosts.allow file, the changes are dynamic, by which we mean you don't have to restart any daemon or process for the changes to take effect.

Types of access

As usual most things become clear with examples, so let's do that now.

To allow access to all daemons belonging to the domain mycompany.com and to deny access from everybody else we would enter:

```
/etc/hosts.allow
ALL:.mycompany.com

/etc/hosts.deny
ALL:ALL
```

Notice in the above example the .mycompany.com, the dot is a wildcard and means "match all domains that have mycompany.com as the end part of their domain name". In the hosts.deny file all other daemons and hosts are denied.

Listing 1: Listing of /etc/xinetd.d/telnet.

```
service telnet
{
        flags           = REUSE NAMEINARGS
        protocol        = tcp
        socket_type      = stream
        wait                    = no
        user              = root
        server           = /usr/sbin/tcpd
        server_args     = /usr/sbin/in.telnetd
        log_on_failure  = USERID
        disable          = no
}
```

When initially learning the rules and patterns, it is best to keep the hosts.deny file to ALL:ALL and only allow access to hosts/daemons specified in the hosts.allow file. Remember – keep it simple, it works!

To allow (only) Telnet and FTP from everybody.

```
/etc/hosts.allow
in.telnetd,in.ftpd:ALL

/etc/hosts.deny
ALL:ALL
```

Notice the use of the comma to separate the two daemons, in the client list.

To allow access to Telnet only from hosts that have the network address part 192.168.1:

```
/etc/hosts.allow
in.telentd: 192.168.1.

/etc/hosts.deny
ALL:ALL
```

Notice the use of the dot at the end of 192.168.1. This will match all IP (network) addresses that start with the IP number 192.168.1.

To allow access to all hosts belonging to the domain mycompany.com but to deny hosts belonging to the bighacker.com domain:

```
/etc/hosts.allow
ALL: .mycompany.com EXCEPT bighacker.com

/etc/hosts.deny
ALL:ALL
```

In the above example using the EXCEPT does what it says: it allows the client lists on the left of the word EXCEPT, but disallows access to those on its right. You can use EXCEPT to allow all of the 192.168.2

When making changes to the hosts.deny or hosts.allow file, the changes are dynamic

network in, but not the hosts with say, the following IP addresses:

```
192.168.2.12 , 192.168.2.12, 192.168.2.22

/etc/hosts.allow
ALL: 192.168.2. EXCEPT
192.168.2.12,192.168.2.12,192.168.2.22

/etc/hosts.deny
ALL:ALL
```

However, when using TCP Wrappers internally don't use EXCEPT with IP numbers on an exposed side of your network, as you are open to potential spoofing.

When a host tries to connect to your Linux machine using a denied daemon the connecting host will simply get a blank screen. It is considered good form to display a refusal message, as that way the connecting user will immediately know that they are not allowed to access this particular host. These refusal dialogs are called banner messages.

You have a banner message for each daemon that you wish to protect or guard. In most cases you'll want to display the same message, so it makes sense to copy the same message across to the different banner daemon files you are creating. We will create a denial message for Telnet and FTP connections, which are denied access. From the /etc directory create a new directory structure to hold the banner file(s).

```
$ pwd
/etc
$ mkdir banners
$ cd banners
$ mkdir deny
$ cd deny
```

First create the banner file for the Telnet daemon. Insert the following text into the file called in.telnetd in the */etc/banners/deny* directory:

```
You are not authorised to enter this machine!
Your attempt has been logged.
Access denied to %c
```

Notice the %c at the end of the text: this will display the calling host's IP address.

Next, we handle the FTP connections. There's no need to re-type the text, simply copy the file. Staying in the same directory:

```
$ cp in.telnetd in.ftpd
```

The next task is to tell TCP Wrappers about the banners. Edit */etc/hosts.deny* and add the following:

```
:banners /etc/banners/deny/
```

to the end of the line entry. The hosts.deny file should now look like this:

```
ALL:ALL :banners /etc/banners/deny/
```

If a host is denied from connecting via Telnet or FTP, based on your rules in hosts.deny or hosts.allow, they will now get a denial message before the connection is closed. The connecting host has an IP address of 192.168.1.12. My hosts.allow file contains the following:

```
ALL:192.168.1. EXCEPT 192.168.1.12
```

Notice the above example accepts all IP addresses that start with 192.168.1, except a host that has an IP address of 192.168.1.12.

Using the rules in the last example the message below is printed to the */var/log/messages* file courtesy of syslog:

```
Feb 14 20:43:54 bumper xinetd[1057]: refused
connect from 192.168.1.12
```

You know the IP address of the rogue host trying to connect, though in reality this will probably be the NAT address or the gateway address the user connected to via the Web. If you're running TCP Wrappers on an internal network, then you've got your culprit pinned down to rights.

Similarly the following messages are printed to the /var/log/secure file from the previous example:

```
Feb 14 20:43:53 bumper xinetd[658]: START: ftp
pid=1057 from=192.168.1.12
Feb 14 20:43:54 bumper xinetd[1057]: FAIL: ftp
libwrap from=192.168.1.12
Feb 14 20:43:54 bumper xinetd[658]: EXIT: ftp
pid=1057 duration=1(sec)
```

Informing you that access was denied and what service the calling host tried to connect with.

Listen in please

When putting your rules to the test it's always a good idea to start off by allowing access to all users and all daemons. From this point gradually start cutting down the hosts you want in, once that is accomplished then start on the daemons. This will save you from struggling up a steep learning curve. Hopefully the basic examples we've provided in this article are enough to get you going and some will probably do the job for you.

Conclusion

This utility allows you to quickly and easily close the doors of your computer to potential trouble. Be sure to check out the manpages of TCPD and hosts_options for a full description of this utility.

Info

ftp://ftp.porcupine.org/pub/security/index.html
ftp://ftp.pld.org.pl/software/tcpd/binary/