

# **TCP WRAPPERS THE COMPLETE STORY**

By Mark E. Donaldson  
bandwidthco.com

## **Contents of Paper:**

- ⇒ **Introduction**
- ⇒ **The History of TCP Wrappers: From The Eyes of the Maker**
- ⇒ **TCP Wrappers Installation**
- ⇒ **TCP Wrappers Configuration**
- ⇒ **Logging Information**
- ⇒ **Checking and Testing the Configuration**
- ⇒ **TCP Wrappers Limitations**
- ⇒ **Conclusion**

# TCP WRAPPERS

## THE COMPLETE STORY

By Mark E. Donaldson  
bandwidthco.com

### Introduction

TCP Wrappers, an applications layer program, was written Wietse Venema of Eindhoven University of Technology, The Netherlands. TCP Wrapper is a very flexible public domain tool used to monitor and control remote network TCP/IP services on UNIX systems. Wrappers also protect each network service that can be protected, limit access to as much as possible, and log all connections to simplify the detection and resolution of problems.

Almost every application of the TCP/IP protocol suite is based on a client-server model. Four of the client-server processes are shown in the Table I below.

TABLE I CLIENT-SERVER PROCESSES		
Client	Server	Application
telnet	telnetd	remote login
ftp	ftpd	file transfer
systat	systatd	show users
finger	fingerd	show users

For example, when a user invokes the **telnet** command to connect to a computer system, a telnet server process is executed on the target host. The telnet server process connects the user to the login process. When someone connects to a host, a single daemon (**inetd**) waits for the connection to be established. It runs the appropriate server program, and then returns to waiting for other connection calls.

The TCP Wrapper programs are configured so that instead of running the server program, **inetd** is essentially tricked into running a small wrapper program. The wrapper logs the client host name or address, performs some additional checks, and executes the appropriate server program. One of the additional checks includes the definition of access control lists which are used to allow or deny access to the host.

Applied usage of access control lists demonstrates how flexible the wrappers program really is. For example, access restrictions can be modified for each individual network service. **Finger**, for instance, can be restricted to a local subnet while **ftp** access is provided to the entire world. Each individual remote access is logged with a time-date stamp, service, connection accepted-rejected status, and remote host name. The access control language also allows for an arbitrary shell command to be executed whenever an associated access rule fires.

# TCP WRAPPERS

## THE COMPLETE STORY

By Mark E. Donaldson  
bandwidthco.com

The following attributes of TCP Wrappers are of prime importance:

- Wrappers can be installed without any changes to existing software, or to existing configuration files.
- The wrapper programs have no interaction with the client user.
- The wrappers have no interaction with the server application.
- Once the wrappers have established interaction between client and server, the wrapper disappears. Consequently, there is no overhead on either end.

The TCP Wrapper package can monitor and filter incoming requests for the **SYSTAT**, **FINGER**, **FTP**, **TELNET**, **RLOGIN**, **RSH**, **EXEC**, **TFTP**, **TALK**, and several other network services. It supports both 4.3 BSD style sockets and SRV4 style TLI. The program requires that network daemons be spawned by a super server, such as *inetd*, and an available syslog library and *syslog* daemon.

### The History of TCP Wrappers From The Eyes of The Maker

The story begins in 1990 at Eindhoven University of Technology where Wietse Venema was administrator of the computer system network. The university system was coming under increasingly heavy attacks from a Dutch computer cracker who was consistently able to gain root privilege. The cracker was skilled at typing the following command sequence:

```
rm -rf /
```

This command, when executed with root privilege, can be as destructive as the MS-DOS *format* command. Typically, all data on the system was erased or destroyed. The destructive behavior of the cracker made it nearly impossible to determine exactly what was being done to the system, or how it was done, since *rm -rf* effectively removed all traces of the break-in and the related actions.

One night, Wietse noticed the cracker was watching him over the network, making contact with the *finger* network service. Since *finger* does not require a password, he was finally able to explain why much of the cracker's activities had gone unnoticed. Wietse's first reaction was to shutdown the *finger* network service. Instead, he decided it would prove more beneficial to maintain the service and determine where the *finger* requests were coming from. In order to do

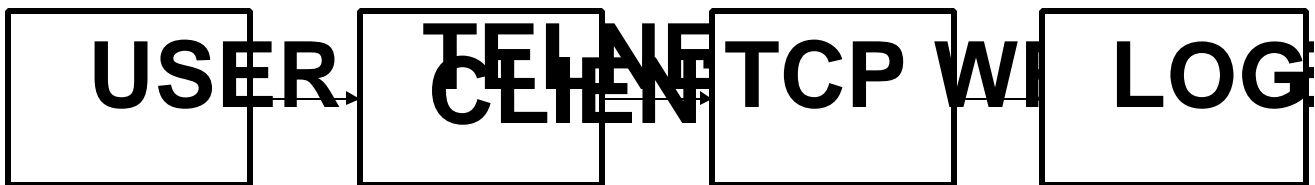
# TCP WRAPPERS

## THE COMPLETE STORY

By Mark E. Donaldson  
bandwidthco.com

this, he decided he must get the name of the host that the cracker was spying from. However, he wanted to be able to achieve this without changing his software configuration because he did not own a source code license.

There was a solution. The trick he found was to make a swap by moving the vendor provided network server programs to another location, and install a *trivial* (TCP Wrapper) program in their place. Whenever a connection was made, the trivial program would just record the name of the remote host (Figure 1), and then run the original network server program.



### Figure 1: The First V

Here, the original telnet server program has been moved to some other place, and the TCP Wrapper has taken its place. The wrapper logs the name of the remote host to a file.

The first TCP Wrapper version was just a few lines of code that Wietze carefully copied from an old network daemon source. Because it did not exchange any information with the client or server processes, the same TCP Wrapper version could be used for many types of network service. This is exactly what he did. Figure 2, below, is an example of one of the original logs obtained from the installation.

The cracker literally bombarded Wietze's system with *finger* and *systat* requests. These allowed him to see who was on the system. Every now and then he would make a telnet connection, presumably to make a single login attempt, and disconnect immediately so no *repeated login failure* would be reported to the system console. Thus, while the cracker was spying on the network, Wietze could now see where he was. This was a major improvement over the past, when the only way he knew something had happened was, unfortunately, only after the *rm -rf* act had been performed.

Initially, Wietze feared that he would be swamped by logfile information, and he thought there might be too much noise to find the desired signal. Fortunately, the cracker was easy to recognize because he normally only worked at night, and often used the *systat* service. Also, he would often make a series of connections to a number of the university systems. By spreading

# TCP WRAPPERS

## THE COMPLETE STORY

By Mark E. Donaldson  
bandwidthco.com

his probes, the cracker hoped to hide his activities. However, by merging logs, Wietse was easily able to finger the cracker.

may 21 14:06:53	tuegate:	systatd:	connect	from	monk.rutgers.edu
may 21 16:08:45	tuegate:	systatd:	connect	from	monk.rutgers.edu
may 21 16:13:58	trf.urc:	systatd:	connect	from	monk.rutgers.edu
may 21 18:38:17	tuegate:	systatd:	connect	from	apl.eeb.ele.tue.nl
may 21 23:41:12	tuegate:	systatd:	connect	from	mcl2.utcs.utoronto.ca
may 21 23:48:14	tuegate:	systatd:	connect	from	monk.rutgers.edu
may 22 01:08:28	tuegate:	systatd:	connect	from	HAWAII-EMHI.PACOM.MIL
may 22 01:14:46	tuewsd:	fingerd:	connect	from	HAWAII-EMHI.PACOM.MIL
may 22 01:15:32	tuewso:	fingerd:	connect	from	HAWAII-EMH1.PACOM.MIL
may 22 01:55:46	tuegate:	systatd:	connect	from	monk.rutgers.edu
may 22 01:58:33	tuegate:	systatd:	connect	from	monk.rutgers.edu
may 22 02:00:14	tuewsd:	fingerd:	connect	from	monk.rutgers.edu
may 22 02:14:51	tuegate:	systatd:	connect	from	RICHARKF-TCACCIS.ARMY.MIL
may 22 02:19:45	tuewsd:	fingerd:	connect	from	RICHARKF-TCACCIS.ARMY.MIL
may 22 02:20:24	tuewso:	fingerd:	connect	from	RICHARKF-TCACCIS.ARMY.MIL
may 22 14:43:29	tuegate:	systatd:	connect	from	monk.rutgers.edu
may 22 15:08:30	tuegate:	systatd:	connect	from	monk.rutgers.edu
may 22 15:09:19	tuewse:	fingerd:	connect	from	monk.rutgers.edu
may 22 15:14:27	tuegate:	telnetd:	connect	from	cumbic.bmb.columbia.edu
may 22 15:23:06	tuegate:	systatd:	connect	from	cumbic.bmb.columbia.edu
may 22 15:23:56	tuewse:	fingerd:	connect	from	cumbic.bmb.columbia.edu

**Figure 2: Some of the first cracker connections observed with the tcp wrapper program. Each connection is recorded with a time stamp, the name of the local host, the name of the requested service (actually, the network server process name), and the name of the remote host. The examples show that the cracker not only used dial-up terminal servers (such as monk.rutgers.edu), but also that he had broken into military and university computer systems.**

Having come this far, Wietse realized he now needed some help from the telephone companies of both Holland and United States, and assistance from the administrators of the terminal servers recorded on the logs. He also knew the best thing would be to trace from open terminal servers so the cracker could only reach him after breaking into a regular user account. His hope was that the cracker would leave some useful traces. Based on this thinking, Wietze built a simple access control mechanism into the TCP Wrapper. Whenever a connection from a terminal server appeared in the logs, all traffic from that system would be blocked on the system side. The responsible system administrators from the crackers side would do the same.

# TCP WRAPPERS

## THE COMPLETE STORY

By Mark E. Donaldson  
bandwidthco.com

The cracker was now being substantially limited substantially by what he could do. Foremost, he could no longer attack from publicly accessible terminal servers. All he could do break into regular user accounts and proceed from there. The next step was to find out what user accounts were involved. What Wietze did was stage a program that would consult a table of bad sites and send a *finger* and *systat* probe whenever a connection was made to his system. He was now able to watch the cracker just like the cracker had once watched him. During the next several months he was able to identify several compromised accounts. Each time he would send a notice to the system administrator, and a copy of it to CERT (Computer Emergency Response Team), to keep them informed of the progress as well.

Since automatic reverse fingering had proven useful in the past, Wietze decided to integrate the reverse fingering tool with the TCP Wrapper. To this end, the access control language was extended so that arbitrary shell commands could be specified. Now that the decision to execute shell commands was based upon the service and the host name, it became possible to automatically detect some types of suspicious traffic. For example, remote access to network services that should only be accessed from local systems.

For several months, Wietze had noticed several *tftp* requests from remote sites. This protocol does not require a password, and is often used for downloading systems software to diskless workstations, or to dedicated network hardware. The protocol could also be used to read any file on the system. For this reason, it was still widely used by crackers. In response, Wietze set up access control tables (Figure 3) such that *tftp* requests would be handled in the usual manner, while remote *tftp* requests would be refused. Instead of the requested file, a finger probe would be sent to the offending host.

```
/etc/hosts.allow:  in.tftpd: LOCAL:  .win.tue.nl
```

```
/etc/hosts.deny:  in.tftp:  ALL:  /usr/ucb/finger -l @%h 2>&1 | /usr/ucb/mail wswietze
```

**Figure 3: An example of a booby trap on the tftp service. The entry in the first access control file says that tftp connections from hosts within its own domain are allowed. The entry in the second file causes the TCP Wrapper to perform a reverse finger in all other cases. The %h sequence is replaced by the actual remote host name. The result is sent to an electronic mailbox.**

One example of logged *tftp* activity that Wietze received appeared as follows:

# TCP WRAPPERS

## THE COMPLETE STORY

By Mark E. Donaldson  
bandwidthco.com

```
Jan 4 18:58:28 svin02 tftp: refused connect from E40-008-8.MIT.EDU
Jan 4 18:59:45 svin02 tftp: refused connect from E40-008-8.MIT.EDU
Jan 4 19:01:02 svin02 tftp: refused connect from E40-008-8.MIT.EDU
Jan 4 19:02:19 svin02 tftp: refused connect from E40-008-8.MIT.EDU
Jan 4 19:03:36 svin02 tftp: refused connect from E40-008-8.MIT.EDU
Jan 4 19:04:53 svin02 tftp: refused connect from E40-008-8.MIT.EDU
```

Due to the nature of the *tftp* protocol, the refused request was repeated every 77 seconds. The retry interval is implementation dependent and can give some insight about the remote system.

According to the reverse finger results, only one person was active at the time, and apparently the login came from a system in France. France was able to tell Wietze that the cracker came a NASA terminal server. Apparently, this cracker liked to cross the Atlantic from NASA to France, cross the Atlantic again from France to MIT, and yet cross it again from MIT to the Netherlands.

A considerable amount of information and protection was enabled from a very simple program. Unfortunately, the cracker was never arrested. However, his work led to the development and refinement of the TCP Wrappers programs. This paper will now take a closer look at *tcp\_wrappers*.

### TCP Wrapper Installation

Every major Linux distribution comes with TCP Wrappers installed as part of the networking package. This can be verified by checking the /etc directory for two files called hosts.allow and hosts.deny. These are the configuration files used by the TCP Wrapper daemon, tcpd. There should also be a line in the /etc/inetd.conf file which reads:

```
telnet stream tcp nowait root /usr/sbin/tcpd \ in.telnetd
```

The telnet option /usr/sbin/tcpd indicates that whenever a client tries to telnet to your machine, they will first connect to the TCP Wrapper. For other versions of UNIX, information on how to download and install TCP Wrappers can be found in **Appendix A: TCP Wrappers and Information sources**.

Once properly installed, the TCP Wrapper program gives the system administrator a high degree of control over inbound TCP connections. The program is started by inetd after a remote host connects to your computer. Depending upon the configuration options selected and how they are configured, TCP Wrapper will perform one or more of the following functions:

# TCP WRAPPERS

## THE COMPLETE STORY

By Mark E. Donaldson  
bandwidthco.com

- Send an optional banner to the connecting client. The banner made used to display legal messages, advisories, or other warnings.
- Perform a double reverse lookup of the IP address, verifying the DNS entries for the IP address and host name match. If they do not, the finding is logged and the connection is immediately dropped.
- Compares the incoming host name and requested service with an access control list to see if this host or combination of host and service has been explicitly denied.
- Uses the inetd protocol (RFC 1413) to determine the username associated with the incoming connection.
- Runs a command to obtain a list of users on a computer that is attempting to contact the local host.
- Passes control of the connection back to inetd, or to another program for further screening, evaluation, or action.
- Logs all results with the **syslog** facility

In the words of Wietze, there are two ways to install and configure TCP Wrappers:

1. **The Easy Way.** This involves moving network daemons to some other directory, filling the resulting holes with copies of the wrapper programs. This approach involves no changes to the system configuration files, so the risk of system damaged is minimized.
2. **The Advanced Way.** With this method, the network daemons remain undisturbed and the inetd configuration files is modified. This paper will assume usage of advanced mode.

Figure 4 shows a sample of what the `/etc/inetd.conf` file may look like after modification. As an example of advanced mode, when a **ftp** request arrives, **inetd** will run the wrapper program **tcpd** with a process name **in.ftpd**. This is the name the wrapper will use when logging the request and when scanning the optional access control tables. **in.ftpd** is also the name of the server program that the wrapper will attempt to run once the connection is made. Any arguments are transparently passed on to the server program.

# TCP WRAPPERS

## THE COMPLETE STORY

By Mark E. Donaldson  
bandwidthco.com

ftp	stream	tcp	nowait	root	in.ftpd
telnet	stream	tcp	nowait	root	in.telnetd
smtp	stream	tcp	nowait	root	smtpd
shell	stream	tcp	nowait	root	in.rshd
login	stream	tcp	nowait	root	in.rlogind
exec	stream	tcp	nowait	root	in.rexecd
ftpd	stream	tcp	nowait	root	in.ftpd

Figure 4: Sample Lines From /etc/inetd.conf

## TCP Wrapper Configuration

### Permit or Deny

Before TCP Wrappers can be configured, one must decide if their machine is open (default permit), or closed (default deny). Permit mode assumes that most services will be available to other computers on the Internet. This configuration is used when only a few untrusted or troublesome sites must be blocked. The deny assumes that most services will be blocked. This configuration provides the greatest amount of security.

### Access Control Lists

When compiled with **DHOSTS\_ACCESS**, the wrapper programs support a simple form of access control. Access can be controlled by host, by service, or by a combination thereof. The software provides hooks for the execution of shell commands when an access control rule fires. This feature may also be used to install booby traps. Access control can be used to connect clients to the right service. **Right** depends on the requested service, the origin of request, and the host address to which the client connects. Access control is enabled by default. It can be deactivated by editing the **Makefile**, or by providing no access control tables.

With some network applications such as **RSH** and **RLOGIN**, the client host name plays a critical role in the authentication process. Host name information can be reliable when lookups are done from a local hosts table. Authentication control for distributed name services is less reliable because the security of the local machine may depend on a DNS outside the local control area. Here, the wrapper programs verify the client host name returned by the address name DNS server by asking for a second opinion. The name and addresses returned by the DNS server may be of an entirely different requesting host. If any name or address discrepancies are found, or if the second opinion is not available, the wrappers assume that one

# TCP WRAPPERS

## THE COMPLETE STORY

By Mark E. Donaldson  
bandwidthco.com

of the two name servers is lying and that the client host is pretending (spoofing) by using another machines host name.

When compiled with **DPARANOID**, the wrappers will always attempt to look up and double check the client host name, and will always deny service when a host name or address discrepancy occurs. Without **DPARANOID**, the wrappers still perform host name lookup by default. Host name and address irregularities can be filtered with the **PARANOID** wildcard.

Two files need to be present for access control to be utilized:

- Access will be granted when a daemon-remote client pair matches an entry in the */etc/hosts.allow* file.
- Access will be denied when a daemon-remote client pair matches an entry in the */etc/hosts.deny* file.

If neither files exist, daemon access is granted to all hosts.

The format of these files is as follows:

```
daemon_list : client_list [ : shell_command ]
```

daemon\_list (service) is a list of one or more daemon process such as *ftpd*, *ftpp*, and *telnetd*, or any of a number of legal wildcards. Client\_list (host) is a list of one or more hostnames, host addresses, patterns, or legal wildcards.

### Default Permit

In permit mode, the */etc/hosts.allow* file is normally left empty, while blocking instructions are placed in the */etc/hosts.deny* file. For instance, if access to *telnet* and *rlogin* is to be denied to anyone connecting from the machine cracker.nasty.net, and anyone in the domain cracker.org, the following line in the */etc/hosts.deny* file would be necessary:

```
in.telnetd in.rlogind : cracker.nasty.net \ .cracker.org
```

The leading "." in front of cracker.org signals tcpd to deny access to any machine in that network domain. Since crackers at these sites probably know how to exploit most network services, access can be denied to all services using the wildcard **ALL**:

# TCP WRAPPERS

## THE COMPLETE STORY

By Mark E. Donaldson  
bandwidthco.com

**ALL : cracker.nasty.net .cracker.org**

Other wildcards that can replace specific host names include **LOCAL**, **UNKNOWN**, **KNOWN**, and **PARANOID**. **LOCAL** matches any name without the "." designation. **KNOWN** and **UNKNOWN** refer to hosts either found or not found in the Domain Name Service. **PARANOID** matches any host name that does not match its stated network address. This option is not often used since wrappers are compiled to reject access to any host that matches this condition before checking the hosts.allow and hosts.deny files.

To allow access to all network services to computers on the local network or subnetwork, the following line can be placed in the /etc/hosts.allow file:

**ALL : LOCAL**

### Default Deny

In the closed or default deny mode, all services to all outside machines can be blocked with the following entry in the hosts.deny file:

**ALL : LOCAL**

This configuration is effective since the hosts.allow file is checked first by the system, then the hosts.deny file, and finally, access is granted only if no match is found in the hosts.deny file.

Again under the deny configuration, only those services intended for use by foreign network machines are listed in the hosts.allow file. Of course, access must be left open to local machines. Also, suppose it may be necessary to telnet into your own computer from an Internet Service Provider. Or, suppose you wish to allow all users to finger local accounts. The rules placed in the /etc/hosts.allow file to achieve this would appear as:

**ALL : localhost**  
**in.telnetd : respective.isp.net**  
**in.fingerd : ALL**

To keep the before mentioned crackers from using finger to obtain information, the following entry would be substituted instead:

**ALL : localhost**  
**in.telnetd : respective.isp.net**

# TCP WRAPPERS

## THE COMPLETE STORY

By Mark E. Donaldson  
bandwidthco.com

**in.fingerd** : **ALL EXCEPT .carcker.org**

### Advanced Options

TCP Wrappers can be compiled to include powerful extensions beyond the normal access controls. For example, the line in the configuration files:

**service** : **host**

can be replaced with the line:

**service** : **host** : **option** : **option** : **option** ...

where **option** can be **allow**, **deny**, or several other advanced options. To enable the advanced options, TCP Wrappers must be compiled with **DPROCESS\_OPTIONS**. With the advanced options, the */etc/hosts.deny* file can be eliminated since the option for **allow** and **deny** can be added to any rule. The advanced options also allow a change in the logging level and the priority or niceness of the network service. So too, they enable use of the *ident* protocol and banners feature.

Since the */etc/hosts.deny* file is no longer needed, only the rules in the */etc/hosts.allow* file need to be rewritten. Entries in the file may appear as follows:

```
ALL      :      localhost      :      allow
in.telnetd :      respective.isp.net :      allow
in.fingerd :      ALL EXEPT .cracker.org :      allow
in.tfpd   :      cracker.nasty.net   :      spawn \
          (/usr/sbin/safe_finger -l @%h | \
          /bin/mail -s %d-%h root) &

ALL      :      ALL      :      deny
```

This configuration would result in the following:

- All services on the local machine would be permitted.
- telnet is allowed on the ISP respective.isp.net.
- finger is allowed from everywhere except by hosts in the .cracker.org domain

# TCP WRAPPERS

## THE COMPLETE STORY

By Mark E. Donaldson  
bandwidthco.com

- tftp would not be allowed from cracker.nasty.net. If they were to try, a booby trap would be released for identification.
- All other services from anywhere else would be denied.

### Logging Information

TCP Wrappers writes a message into the system logs every time a connection is requested, whether or not it is granted. The messages are written through the standard syslog facility, and by default, are sent to the mail transaction location as well. A refused telnet request may place an entry into the **/etc/syslog** (/var/admin/messages for Linux) that reads as follows:

```
Aug 14 18:30:00 ads in/telnetd [15604]: refused connect from bullwinkle.ucr.edu
```

All telnet login attempts can be assembled with a simple **grep** command:

```
grep telnetd /etc/syslog or grep /var/admin/messages
```

TCP Wrappers produces even more information when booby traps are used.

### TCP Wrappers Limitations

TCP Wrappers cannot control access for services that start at boot time and remain running until the system is shutdown. For example, **SENDMAIL** and **HTTPD** are always listening to their respective ports and require their own access controls.

TCP Wrappers are also vulnerable to host name spoofing. Services such as **RSH** and **RLOGIN** depend on the host name being correct. If a distant or remote DNS server is used, it is possible for an attacker to spoof the name lookup by hiding the true client name. This can simply be overcome by placing an entry for the host name and address in the **/etc/HOSTS** file so authentication is not dependent on a foreign DNS lookup. However, the **/etc/HOSTS** file must be kept current, or accesses will be denied as addresses change. Also, and as already mentioned, when wrappers are compiled with the **PARANOID** option, they not only check the network address by looking up the name, but also by looking up the address. If the two do not match, access is automatically denied.

Another weakness of TCP Wrappers involves source routing, where a computer from an outside address claims to be a trusted computer on the inside network. However, TCP Wrappers can be compiled with **KILL\_IP\_OPTIONS** to disable source routing.

# TCP WRAPPERS

## THE COMPLETE STORY

By Mark E. Donaldson  
bandwidthco.com

### Checking and Testing The Configuration

TCP Wrappers has considerable flexibility. However, with this flexibility comes the likelihood of error. For this reason, Wietse Venema provided two additional programs (**tcpdchk** and **tcpdmatch**) in TCP Wrappers that check the configuration and test its protection.

**tcpdchk** checks the configuration files for problems. Incorrectly used wildcards such as **ALL** or **LOCAL** are identified. Nonexistent host names in the access rules, or rules for services controlled by tcpd in the **/etc/inetd.conf** file, are also flushed by tcpdchk. For example, the output from tcpdchk for the deny configuration above may produce the following results:

```
# tcpdchk -v
```

```
Using network configuration file:    /etc/inetd.conf
```

```
>> Rule /etc/hosts.allow line 6:
```

```
daemons:    ALL
clients: localhost
access: granted
```

```
>> Rule /etc/hosts.allow line 7:
```

```
daemons:    in.telnetd
clients: respective.isp.net
warming:    /etc/hosts.allow line 7:    respective.isp.net
host not found
access: granted
```

```
>> Rule /etc/hosts.allow line 8:
```

```
daemons:    in.fingerd
clients: ALL EXEPT .cracker.org
access: granted
```

```
>> Rule /etc/hosts.deny line 10:
```

```
deamons:    ALL
clients: ALL
access: denied
```

The **tcpdmatch** program tests the configuration against a virtual request for a connection by determining if a connection would be allowed or denied when the daemon and host name are provided.

# TCP WRAPPERS

## THE COMPLETE STORY

By Mark E. Donaldson  
bandwidthco.com

### Conclusion

Recommended by CIAC (U.S. Department of Energy Computer Incident Advisory Committee), TCP Wrappers is inexpensive, effective, and widely used to detect security breaches when used with UNIX servers. However, it is not a stand alone security solution. The software is designed to be used in conjunction with other security devices. Router based packet filtering and firewalls work effectively with Wrappers. **SOCKS**, a package designed to control internal traffic to the outside world, will also function well in conjunction with TCP Wrappers.

# TCP WRAPPERS THE COMPLETE STORY

By Mark E. Donaldson  
bandwidthco.com

## APPENDIX A

### TCP Wrappers Software and Information Sources

#### Software Sources

TCP Wrappers may be obtained from the following locations:

- [ftp://ftpcert.org/pub/tools/tcp\\_wrappers/](ftp://ftpcert.org/pub/tools/tcp_wrappers/)
- [ftp://coast.cs.purdue.edu/pub/tools/unix/tcp\\_wrappers](ftp://coast.cs.purdue.edu/pub/tools/unix/tcp_wrappers)
- [ftp://ftp.win.tue.nl/pub/security/tcp\\_wrappers.txt.Z](ftp://ftp.win.tue.nl/pub/security/tcp_wrappers.txt.Z)
- [http://reaserach.att:/dist/internet\\_security/berferd.ps](http://reaserach.att:/dist/internet_security/berferd.ps)

#### Related Pages and Documents

Information related to TCP Wrappers may be found at the following locations:

- <http://www.ccd.bnl.gov/pds/9410-tcpwrapper.html>
- [ftp://ftp.win.tue.nl/pub/security/tcp\\_wrappers.ps.z](ftp://ftp.win.tue.nl/pub/security/tcp_wrappers.ps.z)
- [http://mis.saic.com/papers/tcp\\_wrapper.ps](http://mis.saic.com/papers/tcp_wrapper.ps)

# TCP WRAPPERS

## THE COMPLETE STORY

By Mark E. Donaldson  
bandwidthco.com

### APPENDIX B

## BIBLIOGRAPHY

Brotzman, Lee E. *Wrap A Security Blanket Around Your Computer*. Linux Journal Issue 40. August 1997.

Eckel, George and Chris Hare. *Building A Linux Internet Server*. 1995. New Riders Publishing.

Frisch, Aeleen. *Essential System Administration*. 1995. O'Reilly & Associates, Inc.

Garfinkle, Simson and Gene Spafford. *Practical UNIX and Internet Security*. 1996 O'Reilly & Associates, Inc.

Hunt, Craig. *TCP/IP Network Administration*. 1994. O'Reilly & Associates, Inc.

Kirch, Olaf. *Linux Network Administrator's Guide*. 1995. O'Reilly & Associates, Inc.

*TIS Internet Firewall Toolkit*. 1997 Trusted Information Systems, Inc. Glenwood, Maryland.

Venema, Wietse. *TCP/IP Wrapper 7.1 README.TXT*. February 1995.

Venema, Wietse. *TCP Wrapper: Network Monitoring, Access Control, and Booby Traps*. July 1992.

Welsh, Matt and Lars Kaufman. *Running Linux*. 1996. O'Reilly & Associates, Inc.