



Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permitted without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Security Essentials: Network, Endpoint, and Cloud (Security 401)"
at <http://www.giac.org/registration/gsec>

Internet daemon (inetd) – what it is and securing it.

Wimpie du Plessis

Dimension Data Security

What is inetd or internet daemon?

If we were to continuously run every daemon, we would have little computer resources left to do anything else. The internet daemon inetd, controls access to network services. *Inetd* “listens” to many ports for incoming service requests. If a request is received, inetd starts the daemon for that service. After the service is provided, inetd shuts down the daemon again. To know which services are available inetd will look at the inetd.conf file in /etc. This happens when the workstation or server boots up. We will be looking at securing inetd and if you need to use a service in inetd to secure it with tcp wrappers. There is no reason for a normal workstation to provide services in the inetd configuration file, /etc/inetd.conf. The format of this file is as follows :

```
<service_name> <sock_type> <proto> <flags> <user> <server_path> <args>
```

Security issues with inetd.

If there is no need for any of the services available in the inetd.conf configuration file disable inetd completely. This can be done on Redhat linux as follows :

```
[root]# /etc/rc.d/init.d/inet stop  
[root]# /sbin/chkconfig inet off
```

Or you can do the following to remove it from the startup scripts :

```
[root]# /bin/rm /etc/rc.d/rc[345].d/*inet
```

Why is inetd services not secure? Lets look at the services in inetd that we definitely don't want to allow and why.

Services we are going to look at :

ftp, telnet, finger, echo, chargen, talk, ntalk, tftp, systat, netstat, time, exec, shell & login

File tranfer protocol (FTP)

FTP suffers from most the same problems as telnet. It allows remote users to transfer files to/from a host using ftp. Since user pass words are easily sniffable (they are trasmitted over the wire in cleartext), it is recommend that you disable the service and instead use a secure file transfer mechanism which encrypts the entire session (such as kerberized ftpd or SSH). If ftp access is a must, the service should be wrapped. Another method would be to allow FTP only by trusted clients via firewalling using IP Chains. If you have to run FTP make use of programs like WU-FTPD, ProFTPD or the BSD FTPD implimentation.

Crackers, when they compromise a system's FTP daemon like to use it as a repository for warez and pornography.

ftpd arguments in inetd.conf :

If you run FTP always make use of TCP Wrappers. If you are using the WU-FTPD there is a number of arguments that you can use in inetd.conf.

Example:

```
ftp stream tcp nowait root /usr/sbin/tcpd in.ftpd -l -i -a
```

Where:

- l causes each FTP connection to be logged.
- L causes each FTP command to be logged. This will include connections as well as commands.
- i causes logging of each put request, it is logged to the xferlog file. -I is overridden if ftpaccess is used.
- o causes the logging of output.
- X modifies -i and -o to cause logging to be done via syslog instead of being logged to the xferlog file.
- u specifies the default mask
- W suppresses the logging of each user logged in under ftp in a wtmp file. Its use is discouraged.
- r causes FTPD to do a immediate chroot to the directory for all users, not just the anonymous or ftp user.
- d causes debugging information to be logged via syslog.
- a causes the /etc/ftpaccess file to be used to control FTPD's configuration.

telnet:

Allows remote users to connect to a host via telnet. Since user passwords are transmitted over the wire in plain text and can therefore be easily sniffed, it is recommended that you disable the service and use a secure terminal access mechanism which encrypts the entire session (such as kerberized telnetd or sshd). Users are sometimes quite surprised if you show them their telnet username and password from a tcpdump or snoop. Another problem with telnet is that it is vulnerable to a man in the middle attack. Anybody can intercept a telnet session using a program called "hunt", you don't even need to have any hacking skills at all. Yet another problem with telnet is that crackers can guess telnet passwords as many times as he/she wants because telnetd does not track the number of incorrect guesses. If telnet is needed use a SSL-wrapped telnet solution.

finger:

Finger is used remotely to obtain information about a user. These include:

- Terminals the user is currently logged into
- Shell he uses
- Login directory path
- If there is unread email for the user
- Contents of his/her .plan and .project files

Finger gives out a lot of information that a cracker can use to his/her advantage. He can see the username and then he only needs to guess the password. As an alternative you can make use of TCP Wrappers to restrict access to the finger daemon by using the hosts.allow file and hosts.deny file. Considered highly insecure. Rather use something like cfinger. **If you don't need to make use of finger don't enable it at all.**

echo:

The echo port echoes back all data sent to it to test connectivity. These port can be corrupted by crackers. The UDP version of echo can be used for DoS attack against a third system.

chargen:

Chargen, outputs a rotating sequence of printable characters for testing tty devices and other situations where known and varying sequence was desired. A cracker can have system A telnet to system B's chargen port and the leave while B sends unlimited data to A using up bandwidth of whichever of them has a smaller bandwidth to the internet.

talk & ntalk:

These daemons allow two users on different systems to communicate with each other. This allow a minor DoS by allowing people on remote systems to send output your user's screens. If you want to use it make use of TCP Wrappers, otherwise turn it off. Most users now make use of ICQ and AOL, which introduces additional security problems which is beyond the scope of this document.

trivial ftp (tftp):

Allows remote users to transfer files from a host without requiring login. Used primarily by X-terminals and routers. Considered insecure. It is recommended that you disable the service. If tftp access is desired, it is recommended that the "-s" option be used and that the service be wrapped.

systat:

System status (systat), invokes /bin/ps -auxww, which will show you processes running as well as the user it runs as. This service should be disabled or be implemented with TCP Wrappers or blocked with IP Chains.

netstat:

Network status (netstat), invokes /bin/netstat -a, which will show which ports a machine is listening on. This should be disabled or implemented with TCP Wrappers or blocked with IP Chains.

time:

Used for clock synchronization. It is recommended that you disable the service and using ntp to synchronize your system clock.

exec:

Allow remote users to execute commands on a host without needing to log in. Exposes remote user passwords on the network, thus highly insecure. It is recommended that you disable this service.

shell:

Allows remote users to run arbitrary commands on a host via the Berkeley rsh utility (via the trusted hosts mechanism using the .rhosts file). Considered highly insecure. It is recommended that you disable the service and use SSH instead. If rsh access is a must, the service should be wrapped.

login:

Allows remote users to use the Berkeley rlogin utility to log in to a host without supplying a password (via the .rhosts mechanism). Considered highly insecure. It is recommended that you disable the service and use SSH instead. If rlogin access is a must, the service should be wrapped.

Lets have a look at TCP Wrappers and see what it can do for us.

TCP Wrappers

TCP Wrappers allows administrators to wrap a security layer between the raw TCP and UDP socket layer administered by /etc/inetd and the application which implements each service. This can then be controlled by files that will determine who has access to services in inetd.conf, namely /etc/hosts.allow and /etc/hosts.deny. TCP Wrappers also

tums off any possible source routing for TCP sockets (but not UDP sockets) to eliminate most TCP protocol-level spoofing.

In the `/etc/inetd.conf` file, the sixth field on a line is the program to invoke to supply the requested service. Many of these programs names begin with `in.whatever`. Some common ones are `imapd`, `ipop3d`, and `in.telnetd`. Following this field are any arguments that are to be passed to the program. TCP Wrappers works by replacing this program with the name of the TCP Wrappers daemon, usually `/usr/sbin/tcpd`. The `tcpd` program looks at `/etc/hosts.allow` and `/etc/hosts.deny`.

How to enable TCP Wrappers :

Edit `/etc/inetd.conf`

In order to wrap a service, you simply change its entry in `/etc/inetd.conf` from

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

to

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

You change the full pathname to a daemon to the pathname to `tcpd`, leaving everything else untouched.

Blocking access via TCP Wrapper using `hosts.allow` and `hosts.deny`:

`/etc/hosts.allow`:

```
in.telnetd: hostname.your-domain.com hostname.customer-domain.com
```

```
in.smtpd: ALL
```

```
ftpd: .your-domain.com.customer-domain.com
```

```
nntpd: 192.168.2.
```

`/etc/hosts.deny`:

```
ALL: ALL
```

Inetd Trojan example:

It is easy to create a Trojan once a cracker has root access. All that needs to happen is something like this:

```
echo ingreslock stream tcp wait root /bin/sh -i > /tmp/bob  
/usr/sbin/inetd /tmp/bob  
/bin/rm /tmp/bob
```

The ingreslock service is for locking parts of the Ingres database. It uses port 1524 TCP and UDP. Many administrators allow this service by mistake. Any unused service can be used in the above example, and since the attacker needs root access to do this he/she can add any new service in /etc/services or alter the port number of an existing service.

To detect this on your system you need to use ports or netstat -atuvp.

References:

1. Amett, Matthew. "Inside TCP/IP, Second Edition" New Riders 2001
2. Toxen, Bob. "Real World Linux Security : Intrusion Prevention, Detection, and Recovery" 1995
3. Indiana University. "Configuring inet.conf securely." 16 May 1999. URL: <http://www.uwsg.indiana.edu/security/inetd.html> (23 March 2001).
4. Warfield, Michael. "Securing Linux, Part 1, Elementary security for your box", "Securing Linux", July 1999 <http://linuxworld.com/linuxworld/lw-1999-05/lw-05-ramparts-4.html> (23 March 2001)
5. Scambray, Joel. "Hacking Exposed: Network Security Secrets & Solutions - Second Edition." McGraw-Hill 2001

© SANS Institute 2000 - 2002, Author retains full rights.