The Linux daemon on steroids

XINETD

If you are already familiar with inetd from UNIX or the earlier versions of Linux, just think of xinetd as inetd on steroids; it can do what inetd does plus a whole lot more.

xinetd in a nutshell

xinetd is the main TCP/IP server and it controls the majority of network connections to your host, allows connections to be logged, provides general access controls and time-based access control. It also allows specific services to be bound to a specified interface to allow balance of network traffic to the host. It can also be used to forward services to another host as a sort of DIY fail-over service. xinetd's job though, is mainly to determine what daemon should start for each incoming connection, like Telnet, FTP or rsh. We won't look at all these features just yet.

xinetd is shipped with most Linux distributions. Depending on your set-up you could just have one file, */etc/xinetd.conf*, but this configuration can be a bit of a headache to administer. Most vendors now split the configuration up into many files, one file for each service and one main default configuration file, as shown in Figure 1.

Be advised, if any changes are made to the xinetd configuration files the xinetd daemon must be restarted by either:

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

Or alternately:

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

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

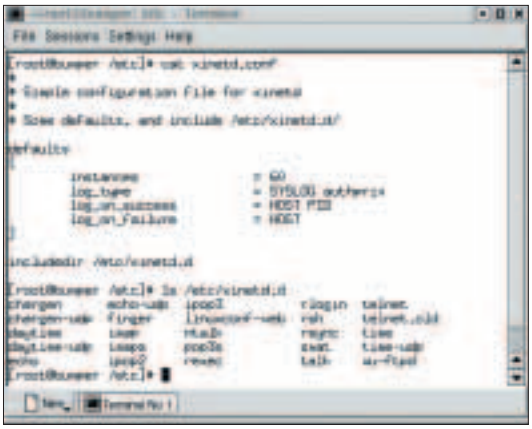

Figure 1: Listing of xinetd.conf and /etc/xinet.d directory

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

Or alternately:

```
$ /usr/bin/killall -TERM xinetd
```

The configuration basics

The main default file, */etc/xinetd.conf*, lets you set the defaults for logging successful and failed connections. In Figure 1, the instances is set to 60, this is the number of requests that a service can handle at a time. If the Linux machine is part of a big network, pump it up to 90. The log_type is how and where the logging will occur, as this is set to SYSLOG authpriv, then syslog will handle the logging, which is the default. Being logged as an authpriv means the content maybe privileged information, like usernames or IP addresses, but not the passwords. Depending on your Linux flavour these messages will either be logged into */var/log/messages* or more probably to */var/log/secure*.

Lets now look at a typical service configuration file – as Telnet is a commonly used service this makes it a good choice.

The flags option of REUSE, lets the TCP/IP socket (that's the protocol Telnet uses) to be well, reusable, in simple terms all this means is that the service can be restarted on the fly. The socket_type STREAM is the type of TCP/IP used, stream is used for both Telnet and FTP connections. When a connection for Telnet is requested xinetd will either

In a previous article we looked at TCP Wrappers and how you can protect incoming TCP-based connections, like FTP and Telnet. In this month's article David Tansley looks a bit more closely at securing your server with xinetd, or the Extended Internet Service Daemon to call it its proper name

operate a multithreaded (unlike MS Windows) or single service. The wait option says NO, so for every new connection a new instance of in.telnetd daemon will be created. If wait had been set to YES, then the incoming Telnet connection would wait until the in.telnetd daemon had finished serving the previous request, before it would service the next request. The user is ROOT, this means the service will run as root User ID. The actual server will be the Telnet daemon */USR/SBIN/IN/TELNETD*. It would be good to log all failures to syslog, so log_on_failure will log the User ID as well as the IP Address of the failed connection. When xinetd is initially shipped it comes pretty much secure, with Telnet disabled, so if your machine is on a network and you cannot connect to your host simply change the disable entry from YES to NO.

Controlling services

To disable a service there is no need to go around every services file located in */etc/xinetd.d* and then edit the particular services file you wish to disable, this can be done globally through the defaults file */etc/xinetd.conf*. Here's how: simply put the service you want disabled on a new line that contains the following:

```
disabled = <service to disable> <service to
disable> <?.>
```

So, to disable say Telnet and FTP you would create a entry like the following:

```
disable = telnet ftp
```

Notice that a space separates the services.Figure 2, shows the defaults file with FTP and Telnet disabled. As mentioned before, for the effect to take place you'll need to restart xinetd.

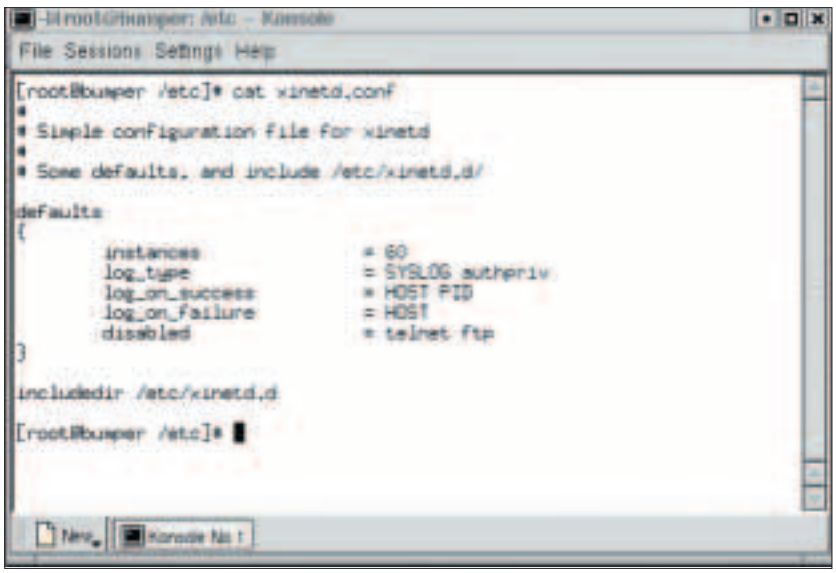

Figure 2: Listing of /etc/xinetd.conf with FTP and Telnet disabled

IP-based control access

You may want to allow connections to a service from the local network, but disallow it from any other network. This is accomplished using the keywords ONLY_FROM by just specifying the IP/Network address. If we want only Telnet to be accessible from the network address of 192.168.10.0. Edit the Telnet file in */etc/xinetd.d* directory and insert the following:

```
only_from = 192.168.10.0/24
```

The zero at the end of the IP address (192.168.10.0) is a wildcard. The /24 is the netmask.

Normal fully qualified hostnames can also be used, (such as bumper.somedomain.com) as long as they are resolvable. Let's now turn our attention to the FTP service. Imagine we have a local company who download extracts from their database and then FTPs it to our system, so that we can import it into our databases. The other company's host IP address is 192.168.8.23; as our company is very security conscious, we only want this specific IP address to use the FTP service. You are not bound to use network addresses in specifying the entry in ONLY_FROM, though it is much easier to. You can just use the actual IP as in the following:

```
only_from = 192.168.8.23
```

The following is an extract from the error log */var/log/secure*, informing us that a host with an IP address of 192.168.1.12 tried to FTP to our Linux machine and failed. It also tells us the date and time and the process number (PID).

```
Mar16 12:32:42 bumper xinetd[1380]: START: ftp
pid=1383 from=192.168.1.12
Mar16 12:32:42 bumper xinetd[1383]: FAIL: ftp
address from=192.168.1.12
Mar16 12:32:42 bumper xinetd[1380]: EXIT: ftp
pid=1383 duration=0(sec)
```

Being a systems administrator, one of your firsts tasks each day should be to check the logs. To quickly check on failed accesses, use *egrep*. The following one-liner will print out lines that contain either FAIL or Auth* (for Authentication) from files ending in .log.

```
$ egrep "FAIL | Auth*" *.log
```

To specify more that one host IP address, the proper format is to enclose the non-network part in curly brackets, separating them with commas. For example suppose we wanted to specify the following hosts: 192.168.1.8, 192.168.1.20, 192.168.1.22 and 192.168.1.50 on the 192.168.1.0 network. We would use the following to include those IP addresses:

```
only_from = 192.168.1.{8,20,22,50}
```

Similarly, to specify more that one network address, for instance to allow network addresses 192.168.8.0 and 192.168.10.0, use spaces to separate the entries, like the following:

```
only_from = 192.168.8.0/24 192.168.10.0/24
```

Time-based control access

You may have a security policy where FTP must be closed down when office hours are over. xinetd lets you specify in a HH:MM format when a service can be disabled. To disable FTP from 17:30 through to 09:30 the following morning, using the ACCESS_TIMES entry we could specify the following in the FTP (wu-ftpd) file:

```
access_times = 17:30-09:30
```

Being more adventurous we can also specify that the service is to be disabled at lunch break times:

```
access_times = 12:30-14:00  17:30-09:30
```

Unfortunately this format does not allow for a day number or day of week sequence. To disable it over a weekend you'll have to edit the defaults file and insert a disable entry, like we have done previously. A better solution however, would be to make a couple of copies of xinetd.conf, one for normal working (xinetd.live) and the other with the daemons you wish disabled (xinetd.disable), then use *cron* to automate it.

The following crontab entries would on a Friday at 17:30 copy the xinetd.disable over to xinetd.conf, and on Monday at 07:30 copy the original (xinetd.live) back, ready for business.

```
30 17 *   * 5 /bin/cp /etc/xinetd.disable
/etc/xinetd.conf >/dev/null 2>&1
32 17 *   * 5 /sbin/service xinetd restart >
/dev/null 2>&1

30 7   * * 1 /bin/cp /etc/xinetd.live
/etc/xinetd.conf > /dev/null 2>&1
32 7 *  *  1 /sbin/service xinetd restart >
/dev/null 2>&1
```

A bit of redirection

xinetd offers redirection (of sorts). This function allows you to redirect a service to another machine. Why do this? Well suppose your FTP directory structure got blitzed or perhaps the performance of your current machine is under-achieving. You will want a quick solution to redirect all incoming connections to another host. All you need to do is specify the IP address and the port of the forwarding machine. Assume the local host has an IP address of 192.168.1.10. We wish to forward all FTP connections to a backup FTP server, which has the IP address of 192.168.1.15, the FTP port number is 21. To see what port numbers match what service check the */etc/services* file out.

Using the redirect entry, our FTP (wu-ftpd) file would look like Figure 3. Your file may look slightly different. When a host tries to establish an FTP connection to our host, their screen will display a 'Trying?192.168.1.10' message, then the re-direction will kick in and a connection will be established to the backup server (192.168.1.15).

Conclusion

xinetd by itself enables you to create a fairly secure policy from daemons that are launched from xinetd. We have demonstrated how you can control your daemons, based on access via hosts and IP addresses and how to enable/disable the daemons, as well as simple time-based access control. We've also shown how you can implement basic redirection of services to another host. What we've shown this month has been without the involvement of TCP Wrappers, so if you do not have TCP Wrappers installed you're not out on a limb security wise.

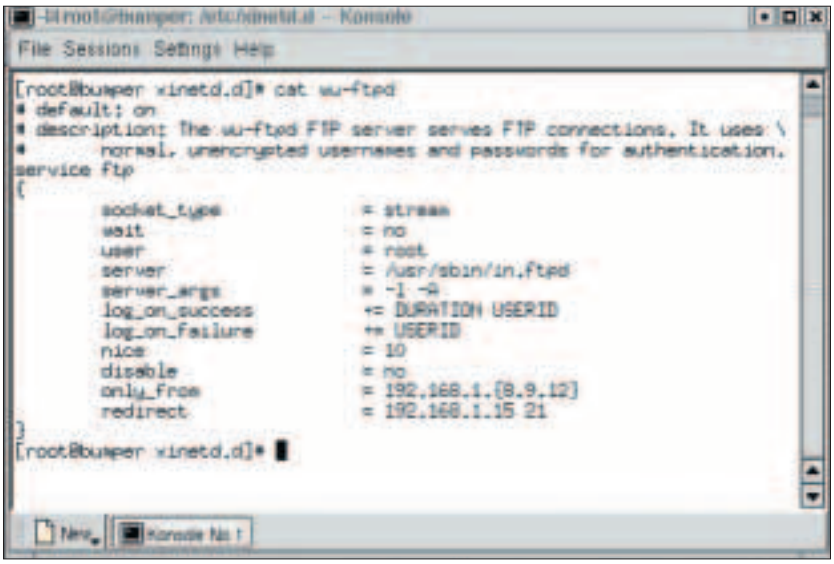

Figure 3. Redirect entry in the FTP (wu-ftpd) file

Info

Xinted homepage
http://synack.net/xinetd

The author

David has written two Linux-based books and several magazine articles and enjoys riding his motorbike when it's not raining. David is a Senior Systems Analyst at ACE Europe, a leading Insurance company.