15.5.1 The xinetd Daemon and TCP Wrappers

Click one of the buttons to take you to that part of the video.

xinetd and TCP Wrappers 0:00-0:55

In this lesson, we're going to look at the xinetd super daemon. Understand that most Linux distributions, when they're initially installed, will include a variety of different network services--or daemons--during the initial installation process.

Most of these services, such as the time service or FTP service, are very handy and they provide a valuable service. But depending on upon how that system is deployed, they may or may not be needed most of the time.

What we need to do is provide a way for these services to be loaded when they're requested by a network host, but then be unloaded when they're not needed anymore. Doing this saves memory, it reduces CPU utilization, and it increases the overall security of the system.

In order to accomplish this, most Linux distributions include a special daemon called xinetd that is used to manage a number of different network services. Let's take a look at how it works.

---

xinetd Super Daemon 0:55-2:27

The xinetd daemon acts as an intermediary between the user requesting a network service and the daemons on the service that actually provide that service. In this situation, let's suppose that this workstation sends a request for a particular service to this server.

By example, let's say we are connecting to the server using the FTP protocol because we want to download a file from the server over here to the workstation. Well, when this FTP request arrives here at the server, it doesn't directly go to the ftp daemon.

Instead, this request is actually intercepted by the xinetd daemon. It is processed by xinetd before it ever gets to the ftp daemon.

The xinetd daemon takes a look at the request and recognizes that it's seeking to establish an FTP connection with the server, so it goes ahead and starts whatever daemon it is that's required for the requested service. In this case, it will start up the ftpd daemon and then forward that request onto it.

Then the daemon itself will fulfill the request. In this case, we are saying, "Please send a us a particular file." So the ftp daemon would respond and let the client download the file.

Then, once that request is complete, xinetd will actually unload the ftp daemon from the system. In this way, ftpd no longer consumes memory and it no longer consumes any CPU cycles.

Network Services Managed by xinetd 2:28-2:45

Understand that not all daemons on your system can be managed by xinetd, but there are many that can. A list of those that are managed by default by xinetd are shown here. For example, ftp, pop3, rsync, smtp, telnet, tftp, time, and vnc.

/etc/xinetd.conf 2:46-3:42

For this to work, you have to configure xinetd, and this is done using the files located in the /etc directory. The xinetd daemon itself is configured using this file right here: /etc/xinetd.conf.

This file usually, on most distributions I've worked with anyway, works fine using its default settings. I usually don't have to make any changes at all. If you do, that's where you would go.

One key parameter in this file that you need to be aware of is this one shown right here. At the very end of the file, you'll see a directive that reads, include dir /etc/xinetd. This line tells the xinetd daemon to use the configuration files for the individual services that are located in this directory. Basically, the files in this directory tell xinetd how to start each service when it's requested.

/etc/xnetd.d/vsftpd 3:40-6:16

For example, the vsftpd file in /etc/xinetd is used to configure the vsftpd FTP server daemon. We've used the cat command here to display the configuration file for this daemon.

Understand that this file that you're looking at does not configure the daemon itself. All it does is tell xinetd how to start this daemon.

The actual configuration file for the daemon is located somewhere else in the system. I believe it's /etc/vsftpd.conf. We're not worried about that file right now. What we are concerned about is how xinetd is going to start this daemon when someone sends an FTP request to this server.

One of the most important parameters in this file is this one right here, disable. The disable directive specifies whether or not xinetd is allowed to start this daemon when a request arrives at the system.

In this example, it is set to yes, which means the daemon will not be started when requested. If we wanted to enable this, we'd have to open this file in an editor and change this to a value of no.

The other parameter you need to be concerned with in this file is the server parameter. This specifies the daemon that xinetd should actually start. In this example, if an FTP request arrives at the system and is intercepted by xinetd, we're telling it to go ahead and load the /usr/sbin/vsftpd executable.

This is the executable that starts the vsftpd daemon. If you make any changes to any of the xinetd configuration files located in this directory--for example, if we went in here and enabled the vsftpd daemon--the change will not take effect until the xinetd daemon itself has been restarted. If you make any change to any of these files, be sure you restart the xinetd daemon itself so that it picks up the change that you just made.

I want to point out another mistake I see happen all the time when folks work with xinetd, and that is the fact that they may enable a particular service, like we've done right here, but then they forget to go into the host-based firewall of that system and open up the respective ports that are required for communications to occur.

In this case, I'd want to go into my host-based firewall and open up ports 20 and 21 to enable FTP communications through the firewall. If you don't, then the xinetd daemon is never going to see that FTP request come in from the client, and therefore will never start the vsftpd daemon.

TCP Wrappers 6:15-9:06

It's important that you realize that if you do enable a particular service using one of these configuration files in xinetd.d, then any network host can come in on the network and connect to that service through xinetd.

Depending upon on how your system is deployed, you may or may not want this to happen. For example, you may want to control access to these services. You may want to limit access to only a specific host or set of hosts and deny access to everybody else.

If this is the case, then you can configure xinetd to work with TCP Wrappers. TCP Wrappers works in conjunction with xinetd to start and run network services using a specific set of access control files that specify who can and who cannot access one of these services.

In order to use TCP Wrappers, the first thing we need to do is go in and enable TCP Wrappers within each service's configuration file in the /xinetd.d directory. Really, the first thing you need to do is go and make sure the tcpd package has been installed. This is the TCP Wrappers daemon itself. It's got to be there; otherwise, none of what we're going to do is going to work.

Then you open up the appropriate configuration file that you want to work on in a text editor. In this case, we're going to be working on the vsftpd configuration file and xinetd.d, and we're going to get down to the server line that we looked at before. Remember, previously for server, we specified the full path to the executable for the daemon.

We're going to use TCP Wrappers; we're going to change that. Notice up here that I've actually commented out the original server line, which originally told xinetd to unload the vsftpd daemon when a FTP request came in on the network.

Instead, what we're going to tell xinetd to do now when it receives an FTP request is to instead load the /usr/sbin/tcpd daemon--which, remember, is the TCP Wrappers daemon instead of the ftp daemon. Then we edit this line right here, server_args.

What we need to do at this directive is then tell tcpd which daemon we actually want to run when an FTP request arrives at the system. And notice we set it to /usr/sbin/vsftpd, which is the value we were using before for the server itself.

When an FTP request arrives at the system, we're going to load the TCP Wrappers daemon and then tell TCP Wrappers, using a server argument, to load the vsftpd daemon. Because we've made several key changes to this file, we would need to restart the xinetd daemon so it would pick up the changes.

TCP Wrappers to Restrict Access 9:06-12:32

With that configuration in place, we next need to specify our access controls. To do this, we use two different files. We use hosts.allow and hosts.deny. These files specify who can access the services loaded and managed by the TCP Wrappers daemon--tcpd.

As you might surmise, entries in host.allow allow access, and entries in host.deny deny access. The syntax for these files is pretty straightforward. We enter the name of the service, a colon (:), then we enter which hosts are allowed to request this services.

In this example, we're setting up access controls for the vsftpd daemon and we specify host 10.0.0.10 and 10.0.0.102, and we could put these in either host.allow or host.deny. If we put it in host.allow, it means those hosts are allowed. If we put it in host.deny, it means those hosts are denied.

Here's a very important thing that you need to remember. As these two files are processed by tcpd, tcpd--the TCP Wrappers daemon--will stop searching through these files as soon as it encounters a match. Anything else that comes after that match are not processed.

You need to be familiar with how the search process works. If you don't understand how the search process works, you're very likely to make a mistake in your host.allow and host.deny files. Here's what you need to remember. First of all, the host.allow file is searched first and if a matching entry for the service in question, such as vsftpd, is found in host.allow, then access will be granted according to whatever is configured here.

Say host 10.0.0.10 tried to access the FTP service running on our system, and we have this entry that you see here in host.allow. tcpd would start searching through host.allow.

It would come to vsftpd and then it would say, "Hey, the requesting host is 10.0.0.10. I've got a matching entry here for 10.0.0.10, so I'm going to go ahead and allow access and load the vsftpd daemon, and I'm not going to look in anything else in this file or in any of the other files, either."

On the other hand, if a request comes in--let's say this time it's like 10.0.0.5--and we search through host.allow and we do not find a matching entry in host.allow, then it will start searching through host.deny. It will try to find a matching entry for the service in question for the host that's making the request, 10.0.0.5.

If the match is found, then that host will be denied access. It is possible for a request to be made coming from a host like 10.0.0.22 that is not in host.allow and is not in host.deny. If this happens, then this host will actually be allowed access. You need to configure host.allow and host.deny files very carefully.

Summary 12:32-12:42

That's it for this lesson. In this lesson we discussed how to configure xinetd to manage network services, and we also reviewed how you can use TCP Wrappers to control access to services managed by xinetd.
