15.4.1 Network Security

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

Network Security 0:00-0:58

Let's spend a few minutes talking about how you can reduce your Linux system's exposure and network attacks. It would be wonderful if we all lived in a Utopian world where we could all connect our networks together, and we would be able to trust each other and not worry about people trying to break into our systems.

Unfortunately, such a world does not exist. If your Linux systems are connected to a network, then you need to be very concerned about network attacks. If your network is connected to a public network, such as the internet, then you need to be extremely concerned about network vulnerabilities.

Network security is a huge topic that can take up an entire course all by itself, and we really don't have the time or the space here to really do the topic justice.

The good news is, however, that there are a few simple things that you can do that will really reduce the threat to your Linux systems from network attacks. We're going to talk about those in this lesson.

Unload Unnecessary Services 0:59-8:00

First, we're going to talk about unloading unnecessary services. We're going to talk about installing updates, and the last thing we're going to look at in this lesson is that of enabling a host firewall.

One of the easiest things that you can do to reduce the threat of network attacks is to simply unload network services that are running on your system that you don't need.

Depending on how your distribution was installed, it is very possible that you have several or even many services running on your system that you don't even know are there and that you don't use. You don't need them. You can view a list of installed services and see whether or not they're running using one of these two commands here.

If you're running an older init based Linux distribution, then you can run the chkconfig command at the shell prompt, and this command will list each service and also its status.

If you're running a newer systemd based distribution, then you use this command instead: systemctl list-unit-files. When you do, you see a list of all the different services that are installed on the system and you see whether or not that service is enabled or not.

For example, we have the Bluetooth service right here and on this system it is currently enabled. A quick word of caution. Don't actually go about disabling a service until you know what it actually does.

I say this because it's very likely that there are services that are running on your system that you don't recognize but are actually really necessary in order for the system to work properly.

If you run one of these two commands and you run across a particular service and you have no idea what that service does--for example, let's say this one, avahi-daemon.service. If I don't know what this daemon does, what I really should do before I disable it is go do some research on the internet, take a look at the man utility, maybe the info utility.

Find out what it is, what it does, and if it's necessary or not. If it is necessary, leave it alone. If it's something that doesn't have to be there, shut it down.

In addition to checking for running services, you can also use an extremely useful network security tool called nmap to evaluate your system. Specifically, we use nmap to view a list of open IP ports on the Linux system.

This information is exceptionally useful. Understand that every port that's open on your Linux system represents a potential vulnerability.

Understand that some open ports are necessary. They need to be there for the system to function. However, there may be other ports that are open and listening for connections that don't need to be there. This represents a security risk.

Every port you have open on the system represents a way an attacker can get into that system. The idea is you want to minimize the number of open ports to just that which is necessary for the system to run and provide its network function--and nothing else.

If you run across a port that is currently open in the system that you don't want to be open, you can close that port by stopping the service that's using it, and then disabling that service so it doesn't automatically start when the system is rebooted.

The syntax for using nmap is shown here in this first example running a TCP scan on the system. We run nmap -sT to indicate TCP protocol, and then the IP address of the system we want to check.

Remember IP can use both TCP or UDP ports, so you need to scan not only for TCP ports, but UDP ports as well. The only difference is you change the -s parameter to a U from a T, and then the IP address of the system you want to scan again, and that will check for any open UDP ports.

An example of running a TCP scan against a host with an IP address of 10.0.0.3 is shown here. Notice that I have a list of all the ports that nmap was able to find open on the system. This system was a server, and it does have a lot of ports open because it provides a whole bunch of different services all at the same time.

Then you can take a look and see what the state of each of those ports is--whether it's open, closed, and also what service is using that port. In this case, port 22 TCP is open, and the SSH daemon is the one that opened that port and is using it. You can use this output to decide what should and shouldn't be left running on the system.

In order to disable a service, you can either use its init script in your init directory--if you're running an older distribution-- or you can use the systemctl command to disable that service. Really, what you need to do is two things.

First, shut the service itself down now, so it doesn't keep listening on that port. Then use the appropriate command either chkconfig or systemctl to configure that service to not automatically start when the system boots up.

There is one other thing I want to point out here. Notice that in this command I'm actually using nmap to scan a remote system. That's a good thing to do because by scanning this system across the network from a different computer, I see what an attacker would see on this system.

You should do that. You should also run nmap locally. I would actually go to 10.0.0.3 and run nmap on it as well.

The reason we do this is because the only thing that we can see in the output when we run a remote scan here are the ports that are allowed through the firewall. There could be other services running on the system. We just can't see them currently because the firewall is blocking them.

That's why you go to the system itself and run the nmap command against itself. By doing that we may see additional ports that are currently open on the system, and the host-based firewall is preventing us from seeing them.

The reason we do this is because if we have ports that are running that we can't see from a remote scan like this, and then somehow the firewall got turned off or was compromised in some way, there could be other services with ports open that we didn't know about that could, again, expose the system to an attacker. We want to look at it from both of those perspectives.

In addition to nmap, you can also use the netstat utility to scan for open ports. The syntax is shown here. We run netstat followed by the option that we want to use.

For example, -a lists all listening and non-listening sockets. -i displays statistics for your network interfaces. -l displays just listening sockets. -s displays summary information for each protocol enabled on the system. -r shows your routing table.

An example of using the netstat command is shown here. In this situation, I use the -l option to view a list of listening sockets on the Linux host. And here you can see that I have a socket open for ssh. I have a socket open for ntp.

I actually have three sockets open; it looks like for ntp. By using this output, I can see what is running on the system.

Install Updates 8:01-9:54

In addition to unloading unneeded services, you also need to make sure that your Linux operating system remains current by installing operating system updates. One of the key problems that we have to deal with in the IT world is that software isn't written perfectly, and it is released with bugs in it.

You'll find that most programs, most services--even the operating system itself--has some defects. Some of these defects aren't a big deal. Some are annoying. Others represent really serious security risks that need to be fixed.

As software is released and used, usually these defects are found by system administrators, by users, and unfortunately by hackers. As they're discovered, updates are written and released that fix these defects.

With most distributions you can configure the operating system to go out on the internet and periodically check for available updates, and that's what's happened here in this situation.

You can also manually update the packages on your system using your package manager. Which one you use depends upon which distribution you're using. For example, if you're running Red Hat, you can run this command right here: yum update. An example of that is shown down here.

Because we did not specify a particular package with the yum command in this example, yum goes out and tries to update all packages on the system.

If you did need to update just one particular package, you would provide its name right here. In which case, yum would just grab updates for that particular package. By not providing a package name, we update everything.

If you're running an openSUSE

system, you use the Zypper command. The syntax is exactly the same: zypper update. It will accomplish the same task. If you're using a Debian based distribution, such as Ubuntu, then you use the 'apt-get dist-upgrade' command. This will update all the installed packages on a Debian based system.

Enable a Host Firewall 9:55-14:16

Finally, you need to make sure that you have a host-based firewall running on your Linux system. This is so critical because today, most organizations connect their corporate networks to the internet. While this is really useful from an end user's perspective, it's also terrifying from a system administrator perspective because this exposes your network to a serious security threat.

If a user can go out on the internet, it also means that an uninvited from the internet can come back in and get on your computer. You need to take measures to keep this from happening.

To do this you need to implement some type of host-based firewall on each computer system and by default most distributions come with a host-based firewall installed and configured, but never trust. Always verify that this is the case.

A host-based firewall acts kind of like a gatekeeper between your computer, here, and the external network. This firewall right here monitors all the traffic that flows in both directions between the computer and that external network.

You configure this firewall with a whole list of rules. These rules define what is allowed through and what is not allowed through.

For example, if someone out here in the internet--probably a nefarious person--tries to establish a connection uninvited with my computer down here, that connection request is going to hit that firewall. The firewall is going to look at it and say, "Ah, no. You're not allowed." And that connection is going to be blocked.

On many distributions, such as Fedora, the firewalld daemon is used to implement a host-based firewall. Other distributions might use a different firewall package. You'll just need to check with your particular distribution vendor to see which one it uses.

What you should do is use this command right here just to make sure that firewalld has been installed on the system, and that it is currently loaded, active and running as shown in the output right here. We enter systemctl status firewalld.

With the firewall running, there are several different commands that you can use to manage it. These are shown here.

We can run the firewall-cmd --state command to check the status of the firewall. You can enter firewall-cmd --get-active-zones to display the default firewall zone configuration.

This particular firewall package comes with several predefined firewall zones that you can use going from extremely secure to not very secure at all. You can actually pick whichever one it is that you want to use.

Once you have the firewall running and you have a default zone set up, there may be situations in which you need to open up just a particular port in that firewall to support a particular service. For example, if the system is set up as a web server, then you've got to let web traffic through. Blocking the web traffic with the firewall negates the whole purpose in setting up the web server in the first place.

What you would need to do in this situation is open up port 80 in the firewall to allow traffic through. To do this, you would use the command shown here. We enter firewall-cmd --permanent --zone= followed by the name of the zone.

Then you enter --add-port= followed by the port and the protocol that you want to open up in the firewall. An example of that is shown here. We have firewall-cmd --permanent --zone=public --add-port= 80/tcp. This would allow web server traffic through the firewall.

You can accomplish the same thing using this command right here, where instead of specifying a port and protocol, we specify --add service= followed by the name of the service that we want to allow through the firewall. We can accomplish the same thing that we used in the previous command allowing web server traffic through to the web server using this command here, firewall-cmd --permanent --zone= whatever the name of the zone is, in this case public --add-service this time equals HTTP instead of --add-port=80/TCP.

Both of these commands would accomplish the exact same thing.

Then down here we have the systemctl restart firewalld command to restart the firewall service. This is important because if you make any changes to the firewall configuration, you need to restart the firewalld daemon in order to have that change take effect.

Summary 14:17-14:37

That's it for this lesson. In this lesson, we reviewed several measures that you can take to protect your Linux systems from network attacks.

We first talked about unloading unneeded services. We talked about how to scan the system to find out what is currently running on the system that has ports open. We talked about making sure the system stays updated with updates from your distribution vendor and then we ended this lesson by talking about implementing a host-based firewall.
