15.3.1 Login Blocking

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

Login Blocking 0:00-0:51

In this lesson, we're going to discuss login blocking. Understand that from time to time you may need to completely disable all logins to a Linux system. For example, you may have a very serious issue that needs to be resolved and while you're trying to fix things, you just need to keep everybody out while you try to figure out what's going on.

To do this, two things have to happen. First of all, you need to log out all current users. Then you also need to prevent any other users, new users from logging in. This gives you the window of time you need to get things fixed.

Be aware that the order in which you do things might change. On a lightly used system, you can go ahead and log everybody out and then block all logins, but on a heavily used system, you might need to do just the opposite. You may need to first block all logins and then log out all current users.

Log Everyone Off 0:51-1:51

Let's first look at how you log everybody out of the system. Now in order to do this you first have to find who's currently logged in. One command you can use to accomplish this is the w command. Its command name is very simple.

All you do is just 'w' and hit Enter. When you do, a list of all the currently logged in users is displayed. On this system, I have two users logged in. They are the rtracy user and the ksanders user. Now at this point we need to get these two users to log out.

Ideally, you should be able to send a text to these logged in users and ask them to log out, maybe send them an email and ask them to log out or maybe even call them on the phone and ask them to log out. Hopefully they'll get your message and actually do what you asked and log out of the system.

Unfortunately, this doesn't always work. That's because sometimes users leave their systems logged in while they are away. Maybe they are in a meeting, or maybe they're on a conference call or something like that, and they're not paying attention. In this situation, you may need to brute force log out users.

Use pkill 1:52-2:29

This is done using a command called, pkill. It stands for process kill. The syntax is to enter 'pkill', and then we specify the signal we want to send, which is the kill signal. We enter '-KILL'. Then we specify which user we want to log out of the system.

In this case we're going to force log out the ksanders user. Now after doing this you should have everybody logged off of the system. You can now work and do what it is you need to do to resolve whatever issue you are having.

But you probably also want to disable all future logins, so that someone else doesn't come and try to log into the system while you're busy trying to fix it.

Block Logins 2:30-4:29

Blocking logins on Linux is actually extremely easy to do. All you have to do is go into the /etc directory and create a new file called nologin. That's it. That's all you have to do. There doesn't even have to be any text in the file.

As long as this file exists in /etc, then nobody except for the root user is allowed to login to the system. You can see in this example all I did was run the touch command to create a new file in /etc called, nologin. It's blank, there's nothing to it but because that file exists, all logins are blocked.

Here's a neat feature about the nologin file. If you do decide to go ahead and put text in the nologin file, in the /etc directory, then the text you enter will actually be displayed to the end user when they do try to login.

In this example, I've put in some text that says, "The system will be down until 9:00 pm for maintenance." I put this in using the vi editor and save the changes.

Then when a user tries to log in, the error message that you entered will be displayed. This is really a good idea because if a user tries to log in while you've got logins blocked, all they're going to see is an error message. It's really not very informative.

If you haven't communicated with the users to let them know what's going on, then they're going to think something is wrong. They're going to start calling you wanting you to fix the login problem while you're actually busy trying to fix the real problem with the system.

Putting a little text up there explaining what's going on will save you the aggravation. The users will know that the system is down, and they won't assume that something is broken.

The functionality of the nologin file is actually configured in the /etc/pam.d/login directory. This behavior of the nologin file is actually configured in the /etc/pam.d/login file.

Use PAM 4:30-5:08

The acronym PAM itself stands for Pluggable Authentication Modules. Basically what it does is allow any Linux system that uses PAM to use a variety of different authentication mechanisms.

By default, it's set up to use just a username and a password, but you could reconfigure PAM such that it required a biometric login like a thumbprint or a fingerprint. You could configure it to require a retina scan, or you could require it to use a security certificate or some type of card reading device, whatever it is you want to use.

That's why we call it pluggable authentication modules, because it's plug and play. We can take different modules out and put new modules in and allow authentication to be customized to whatever it is you want to do.

Use /etc/pam.d/login file 5:09-5:55

In this case, we have the /etc/pam.d/login file. Right down here there's a line that says, account required pam_nologin.so. Basically this line causes PAM to check whether the file name nologin exists in the /etc directory. If it does, PAM will not allow regular users to login.

It'll allow root to log in to fix whatever is going on in the system and blocks everybody else. Once you're done, and you're ready to allow people to log back into the system, you can re-enable logins by simply deleting or if you want to, just renaming the nolongin file.

For example, you could use the move command to rename the nologin file to nologin.back in which case, PAM will allow people to log into the system again.

Summary 5:56-6:10

That's it for this lesson. In this lesson, we discussed how you can block user logins. We first reviewed how you can identify who's currently logged into the system. We talked about how to force logged users out. Then we ended this lesson by discussing how you can prevent additional users from logging in using the /etc/nologin file.
