15.2.2 User Restrictions

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

User Restrictions 0:00-0:40

Let's spend some time talking about user restrictions. You can increase the overall security of your Linux systems by limiting user logins and also limiting their access to resources on those systems.

For example, you can impose limits on how many times a user can log in. You can limit how much CPU time they can consume, you can limit how much memory they can use on the system, and so on.

There are two general ways in which you can implement these limits. First of all, you can use the pam_limits to restrict access to resources and you can also use the ulimit command to restrict access to resources. We're going to look at both of those in this lesson.

pam_limits in /etc/security/limits.conf 0:40-1:08

Let's begin by looking at the first option--using pam_limits. You can limit user access to system resources using a pluggable authentication module called pam_limits.

It's configured in the file that you see here: /etc/security/limits.conf. This file contains resource limits that are configured using the syntax that is shown here in the comment part of the file.

domain 1:09-1:51

We specify the domain, the type, the item, and the value. The domain right here describes the entity to which the limit applies, and you can specify a particular user. An example of that is shown right here, we have an ftp user.

You can also specify a group name, so right here we are specifying a group instead of a user, which makes the limit apply to all members of that group. Or you can also specify a star (*) to indicate that this limit applies to everybody.

Notice right away here, that all of these limits are examples in the file, and they're commented out. So they're not actually applied. If you were to remove one of these pound signs (#), it would apply that limit.

type 1:52-2:15

The next part of the file is the type. In the domain we specified who the limit applies to, and under type we define whether it is a hard or a soft limit. A hard limit, such as this one, cannot be exceeded at all. It is hard. It's fixed. A soft limit on the other hand, is one that we do allow the user to exceed temporarily.

item 2:16-2:29

The next column in each limit is the item. This specifies what exact resource is being limited. There are actually many different things that you can restrict using pam_limits.

value 2:27-4:47

For example, you can specify core in the item field to restrict the size of core files. Specifying data will restrict the size of the program's data area in RAM, effectively limiting how much RAM the user can consume. fsize restricts the size of files that are created by users.

The nofile item restricts the number of data files the user can have open at the same time. The stack item restricts the stack size in the CPU. The cpu item restricts the amount of CPU time that a single process can consume.

nproc restricts the number of concurrent processes the user can run at the same time. maxlogin sets the maximum number of simultaneous logins for a given user. priority sets the priority to run user processes with.

The locks item sets the maximum number of locked files that will be allowed. And then finally we have nice, which sets the maximum nice priority a user is allowed to raise a process to.

For each item we have to specify a value, which simply specifies a value for that limit. For example, down here we have @student, which specifies the student group. We're going to set a max login of 4.

Let's take a look at an example. Suppose we want to configure the rtracy user, that is me, with a soft CPU limit of 15 minutes. To do this, we'd open up our /etc/security/limits.conf file, and then we would add this line right here to the file.

This would be a useful limit that we could impose if that rtracy user is running some CPU-intensive program that's hogging cycles away from other users on the system. Likewise, we could also create another restriction down here for the same rtracy user that would limit me to 2 maximum logins at the same time. For that, we're going to set a hard limit this time.

Because we set the CPU limit as a soft one, I would be allowed to temporarily exceed the value that we specified over here. Because this one is a hard limit, I get two logins and two logins only.

Using ulimit 4:48-7:18

In addition to using pam_limits, you can also limit user access to system resources using the ulimit command at the shell prompt. I don't actually use ulimit very often, and here is why.

It doesn't matter if you're logged in using a graphical interface or whether you're working from the shell prompt, pam_limits applies across the board; ulimit only applies to commands launched from the command prompt.

The syntax for using ulimit is shown here. We type ulimit, followed by the options that specify what limit we want to set, and then a value for that limit. The options you can use with ulimit are shown here.

The -c option sets a limit on the maximum size of core files, and this is done in blocks. If you set this option -c to a value of 0, then core dumps on the system are effectively disabled.

-f sets a limit on the maximum size in blocks of files that are created by the shell. -n sets a limit on the maximum number of open file descriptors, so it effectively sets the limit on the number of files the user can have open.

-t sets a limit on the maximum amount of CPU time, in seconds, that a process can use. The -u option sets a limit on the maximum number of processes that are available to a particular user. -d sets a limit on the maximum size of a process data segment in RAM, in KB.

-m sets a limit on the maximum resident size of a process in RAM, and it's also specified in KB. -s sets a limit on the maximum stack size, again in KB. -H is used to set a hard resource limit, while -S is used to set a soft resource limit.

You can also use the -a option with the ulimit command. And if you do that, it just simply displays all the current resource limits that have been set. An example of that is shown here. We have our core file size, data segment size, scheduling priority, file size, and so on.

An example of using the ulimit command is shown here. If we wanted to set a soft limit that would limit the user to a maximum of 50 processes, we could enter 'ulimit -S' to set a soft limit, and then '-u' to set a maximum number of processes, and then the value in this case, '50'.

Summary 7:18-7:30

That's it for this lesson. In this lesson, we discussed how to set user restrictions. We first looked at how you can use pam_limits to restrict access to resources, and then we talked about how you can use the ulimit command to restrict access to resources.
