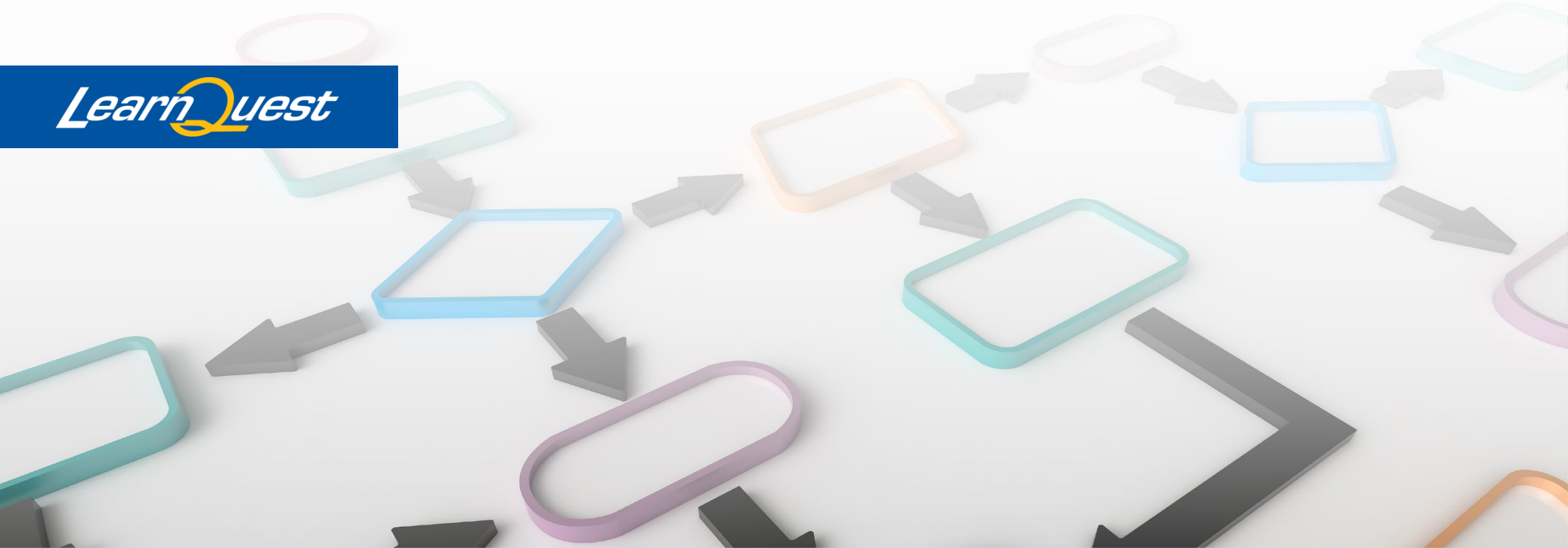
Securing Linux Systems

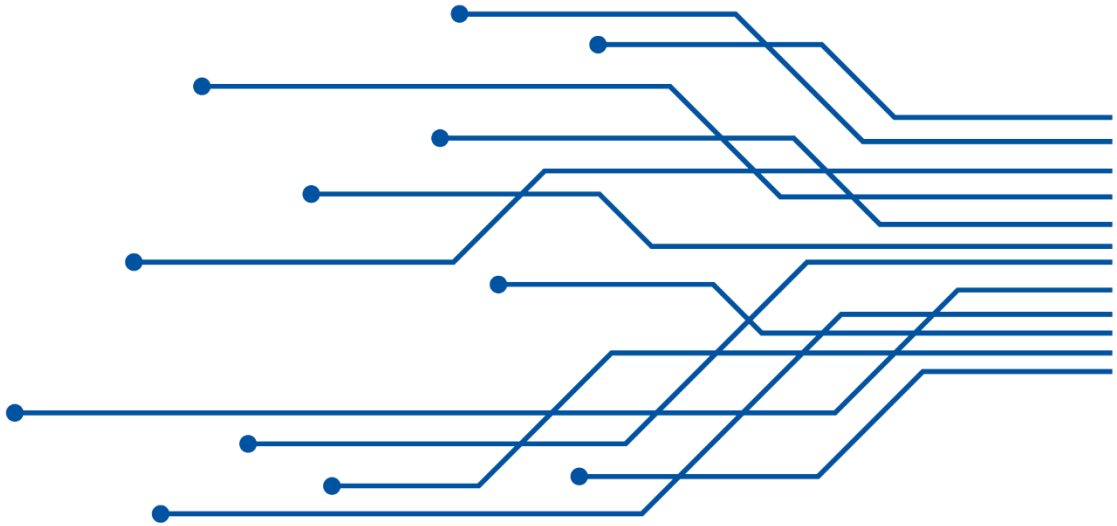
- 3rd Course in Linux Foundations Specialization



Ownership and Permissions

In this module, we look at security and the Linux operating system. We will start by applying permissions to files and directories

1



Learning Objectives

Ownership and Permissions

Upon completion of this module, learners will be able to:

- Apply Ownership and Permissions to Files
- Describe Access Control Lists
- Apply Context-Based Permissions
- Describe Linux Account Types

Lesson 1

File and Directory Permissions

In this lesson we look at how to apply permissions to files and directories

Linux File and Directory Ownership

- Linux uses a three-tiered approach to protecting files and directories:
  - Owner: Within the Linux system, each file and directory is assigned to a single owner.
  - Group: The Linux system also assigns each file and directory to a single group of users.
  - Others: This category of permissions is assigned to any user account that is not the owner nor in the assigned user group.

Chown Command

The root user account can change the owner assigned to a file or directory by using the chown command.

Example Usage:

- chown aspeno file1.txt
- chown –R aspeno /home/john

Options:

- -R : recursively changes the owner of all files under the specified directory.

Chgrp Command

The root user account can change the group assigned to a file or directory by using the chgrp command.

Example Usage:

- chgrp staff file1.txt
- chgrp –R staff /home/john

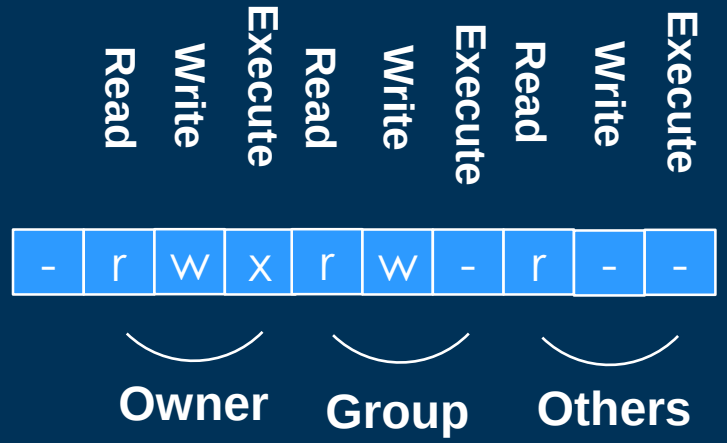
Options:

- -R : recursively changes the group of all files under the specified directory.

Permissions

Linux uses three types of permission controls:

- Read : The ability to access the data stored within the file or directory
- Write : The ability to modify the data stored within the file or directory
- Execute : The ability to run the file on the system, or the ability to list the files contained in the directory



Chmod Command

The root user account or the owner of the file or directory can change the assigned permissions by using the chmod command

Example Usage:

- chmod 666 file1.txt
- chmod ug=rwx file1.txt

Options:

- -R : recursively changes the permissions of all files under the specified directory.

Chmod commands

Octal	Value	Permission Meaning
0	---	No permissions
1	--x	Execute only
2	-w-	Write only
3	-wx	Write and execute
4	r--	Read only
5	r-x	Read and execute
6	rw-	Read and write
7	rwx	Read, write, and execute

Special Permissions

The Set User ID (SUID) bit is used with executable files. It tells the Linux kernel to run the program with the permissions of the file owner and not the user account running the file.

When a directory has the GUID bit set, any files users create in the directory are assigned the group of the directory and not that of the user.

The sticky bit is used to protect a file from being deleted by those who don't own it, even if they belong to the group that has write permissions to the file. The sticky bit is denoted by a t in the execute bit position for others: rwxrw-r-t.

Lesson 1 Review

One user can own a file

One group is assigned to every file

The octal 777 represents read/write and execute for the permissions

Lesson 2

Access Control Lists

In this lesson we look at Access Control Lists (ACLs) in Linux.

Linux ACL

You can only assign permissions for a file or directory to a single group or user account.

Linux developers have devised a more advanced method of file and directory security called an access control list (ACL).

ACL permissions use the same read, write, and execute permission bits, but can be assigned to multiple users and groups.

Setfacl Command

The setfacl command allows you to modify the permissions assigned to a file or directory. You define the rule with three formats:

- u[ser]:uid:perms
- g[roup]:gid:perms
- o[ther]::perms

Example Usage:

- setfacl –m g:staff:rw file1.txt
- setfacl –x g:staff file1.txt

Options:

- -m: modify permissions
- -x: remove permissions

Getfacl Command

The getfacl command allows you to view the ACLs assigned to a file or directory

Example Usage:

- getfacl file1.txt

Options:

- -d : show directory default ACL entries.

Lesson 2 Review

ACL allows you to apply a more specific set of permissions to a file or directory

ACL allows this without changing the base ownership and permissions of a file or directory

ACL uses the same permission bits

Lesson 3

Context-Based Permissions

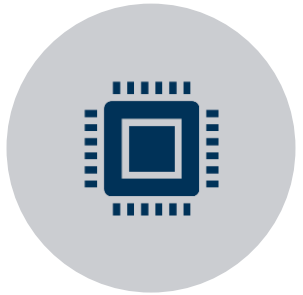
In this lesson we look at how to apply Context-Based Permissions in Linux

DAC vs MAC

The permissions method and the ACL method of assigning permissions to files and directories are called discretionary access control (DAC) methods.

The permission is set at the discretion of the file or directory owner.

An administrator cannot prevent users from granting full permission to others on all the files in their directories.



MAC methods allow the system administrator to define security based on the context of an object in the Linux system to override permissions set by file and directory owners.

MAC Implementations in Linux

There are currently two popular MAC implementations in Linux:

- SELinux for Red Hat–based systems
- AppArmor for the Debian based system

SELinux

The Security-Enhanced Linux (SELinux) application is a project of the United States National Security Agency (NSA)

SELinux implements MAC security by allowing you to set policy rules for controlling access between various types of objects on the Linux system, including users, files, directories, memory, network ports, and processes.

When a user or process attempts to access an object on the Linux system, SELinux intercepts the attempt and evaluates it against the defined policy rules

Security Context

SELinux labels each object on the system with a security context.

The security context defines what policies SELinux applies to the object.

The security content format is as follows:

- user:role:type:level

The user and role attributes are used only in the multilayer security mode and can get quite complex.

Systems running in the default targeted security mode only use the type attribute to set the object security type and control access based on that.

The level attribute sets the security sensitivity level and clearance level. Optional under the targeted security mode.

Viewing and Setting Security Context

To view the security context assigned to objects, add the -Z option to common Linux commands such as id, ls, ps, and netstat.

The semanage utility allows you to view and set the security context for user accounts on the system.

The Linux system sets their security context for files and directories at the time they are created, based on the security context of the parent directory.

You can change the default security context assigned to a file by using the chcon or restorecon utilities.

The chcon format is the following:

- chcon -u newuser -r newrole -t newtype filename

Booleans

SELinux uses a method of enabling and disabling individual policies without having to modify a policy file.

A Boolean is a switch that allows you to enable or disable a policy rule from the command line based on its policy name.

To view the current setting of a policy, use the getsebool command with the –a option to list all policies or pass the policy in. For example:

- getsebool antivirus_use_jit

To change the Boolean setting, use the setsebool command. For example:

- setsebool antivirus_use_jit on

AppArmor

Debian-based Linux distributions commonly use the AppArmor MAC system.

AppArmor is simpler than SELinux as it only allows you to control the files and network ports applications can access.

AppArmor defines access based on profiles.

Profiles are defined for each application in the /etc/apparmor.d directory.

An application package installs its own profiles.

The profile is a text file that defines the files and network ports the application can communicate with and the access permissions.

Lesson 3 Review

MAC methods allow the system administrator to define security based on the context of an object

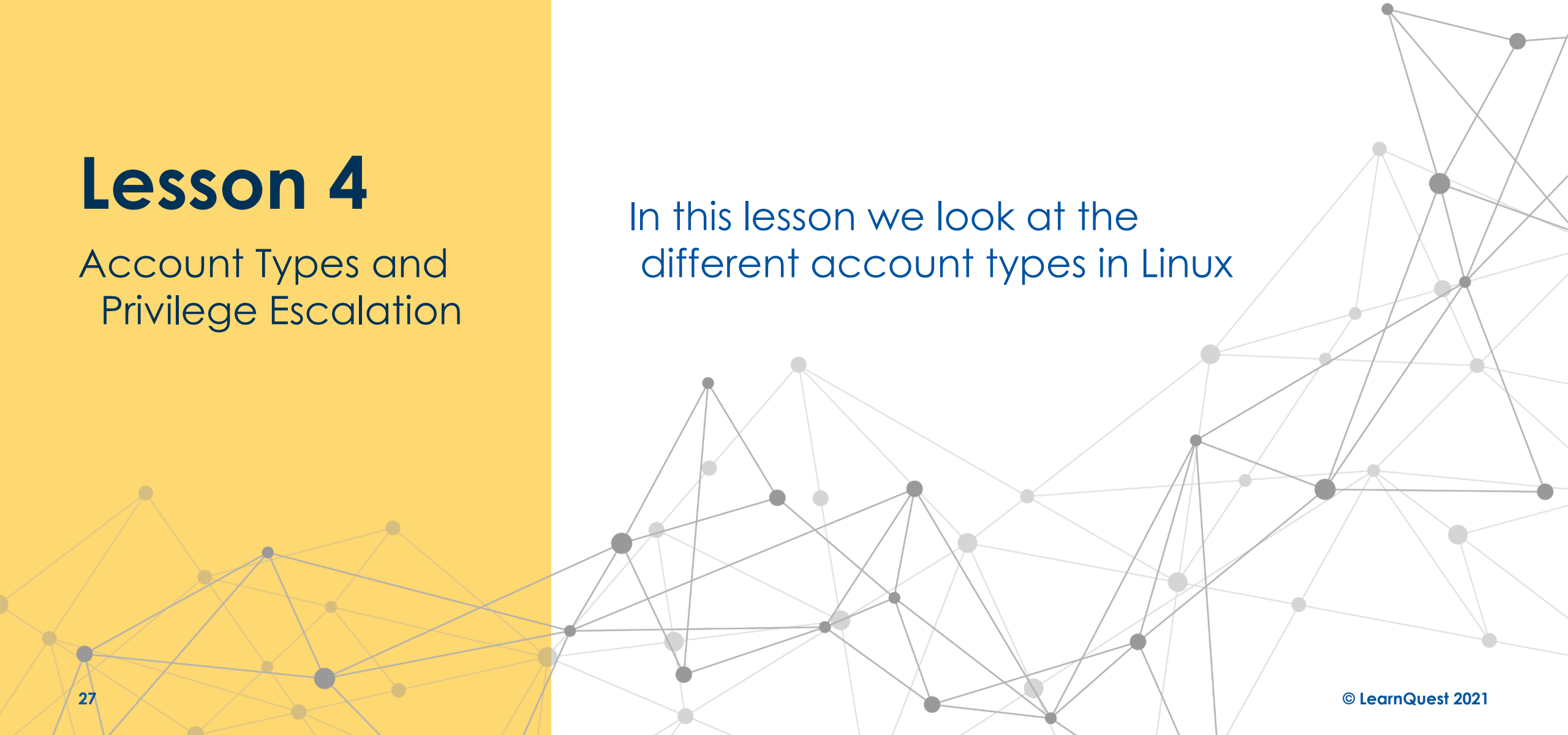
Red Hat based distros use SELinux for MAC

Debian based distros use App Armor for MAC

Lesson 4

Account Types and Privilege Escalation

In this lesson we look at the different account types in Linux



Account Types

Root : The root user account is the main administrator user account on the system. It is identified by being assigned the special user ID value of 0.

Standard : Standard Linux user accounts are used to log into the system and perform standard tasks, such as run desktop applications or shell commands. Standard Linux users cannot access files outside of their $HOME directory unless given permission by the file or directory owner. Most Linux distributions assign standard user accounts user IDs over 1000.

Service : Service Linux user accounts are used for applications that start in the background, such as network services. The password value in the shadow file to an asterisk so that the account cannot log into the system. The login shell is also defined in the /etc/passwd file is nologin to prevent access to a command shell. Service accounts normally have a user ID less than 1000.

Escalating Privileges

su : Short for substitute user. Allows a standard user account to run commands as another user account, including the root user account.

sudo : Short for substitute user do . It allows a standard user account to run any command as another user account, including the root user account.

sudoedit : The sudoedit command allows a standard user to open a file in a text editor with privileges of another user account, including the root user account.

Restricting Users

Two Commands for limiting what users can do, separate from file and directory permissions:

- Ulimit - Restrict access to system resources for each user account.
- Chage - Set how often user account passwords expire

Ulimit Options

-b The maximum socket buffer size.	-c The maximum size of core files created.	-d The maximum size of a process's data segment.	-e The maximum scheduling priority ("nice")	-f The maximum size of files created by the shell(default option).
-i The maximum number of pending signals.	-l The maximum size that can be locked into memory.	-m The maximum resident set size.	-n The maximum number of open file descriptors.	-p The pipe buffer size.
-q The maximum number of bytes in POSIX message queues.	-r The maximum real-time scheduling priority.	-s The maximum stack size.	-t The maximum amount of cpu time in seconds.	-T The maximum number of threads.
	-u The maximum number of processes available to a single user.	-v The maximum amount of virtual memory available to the process.	-x The maximum number of file locks.	

Lesson 4 Review

The root user account has permissions to access all files and directories on the system.

The sudo command allows a standard account to run a command as another

The ulimit command can restrict access for an account