



# Access Controls






Cybersecurity  
Linux 1 Day 2



# Class Objectives

---

By the end of today's class, you will be able to:

-  Edit configuration files using `nano`.
-  Audit passwords using `john`.
-  Elevate privileges with `sudo` and `su`.
-  Create and manage users and groups.
-  Inspect and set file permissions for sensitive files on the system.

# Activities

---

In the activities throughout this week, we are auditing a malfunctioning server.



In previous exercises, we collected important files, stopped malicious processes and installed tools to facilitate the audit.



Today, we'll first look at nano. Then we'll look at user access controls, beginning with passwords.



In the previous class, we made a copy of `/etc/shadow`, which contains the passwords on a system.



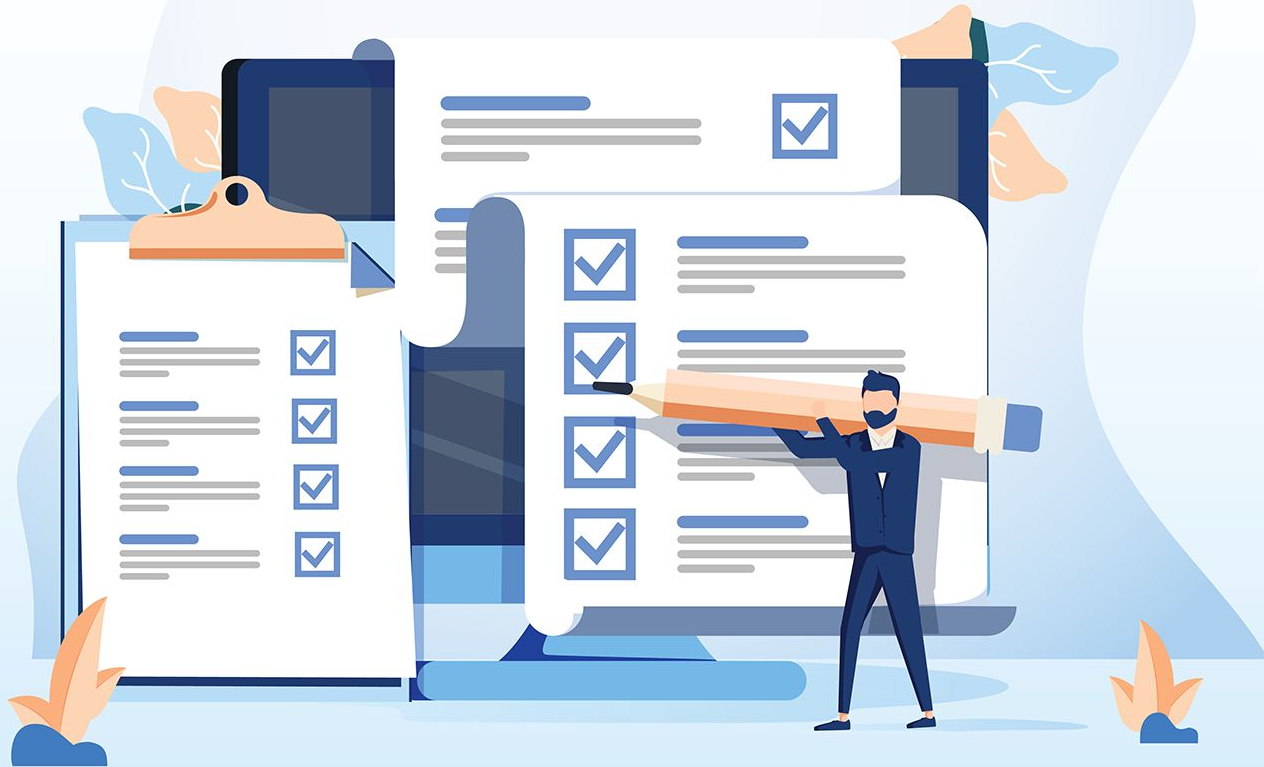
These passwords are obfuscated with a **hashing function**, meaning we can't read them in plain text...yet.

# Installing Nano



In this section of the day, we'll make changes and add new files using **Nano**.

After investigating files, stopping malicious processes, and installing tools to help us audit a system, we need a method for **keeping lists of our findings** and **editing system and configuration files**.



# Nano

---

Since many Linux distributions lack a GUI, editing files using the command line is a common task and essential skill for administrator.



# Nano

---

While there are many text editors available, we will be using Nano for the following reasons:

01

## Default Text Editor

Nano is the default text editor across most Linux distributions.

02

## Learning Curves

Other editors, such as emacs and vim, while powerful, have steep learning curves and are more appropriate for seasoned professionals.



# Demo Scenario

---

We will create a message displayed to all users who log into our system.

The senior administrator has asked that we set up a message alerting users that our system will be “down for maintenance.”

This way, we can conduct our audit with out users making further changes.

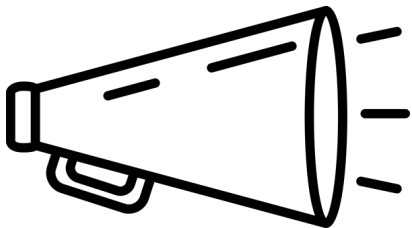


# Demo Scenario

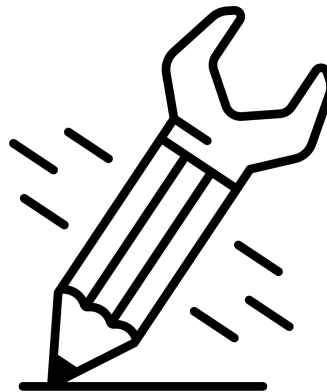
---

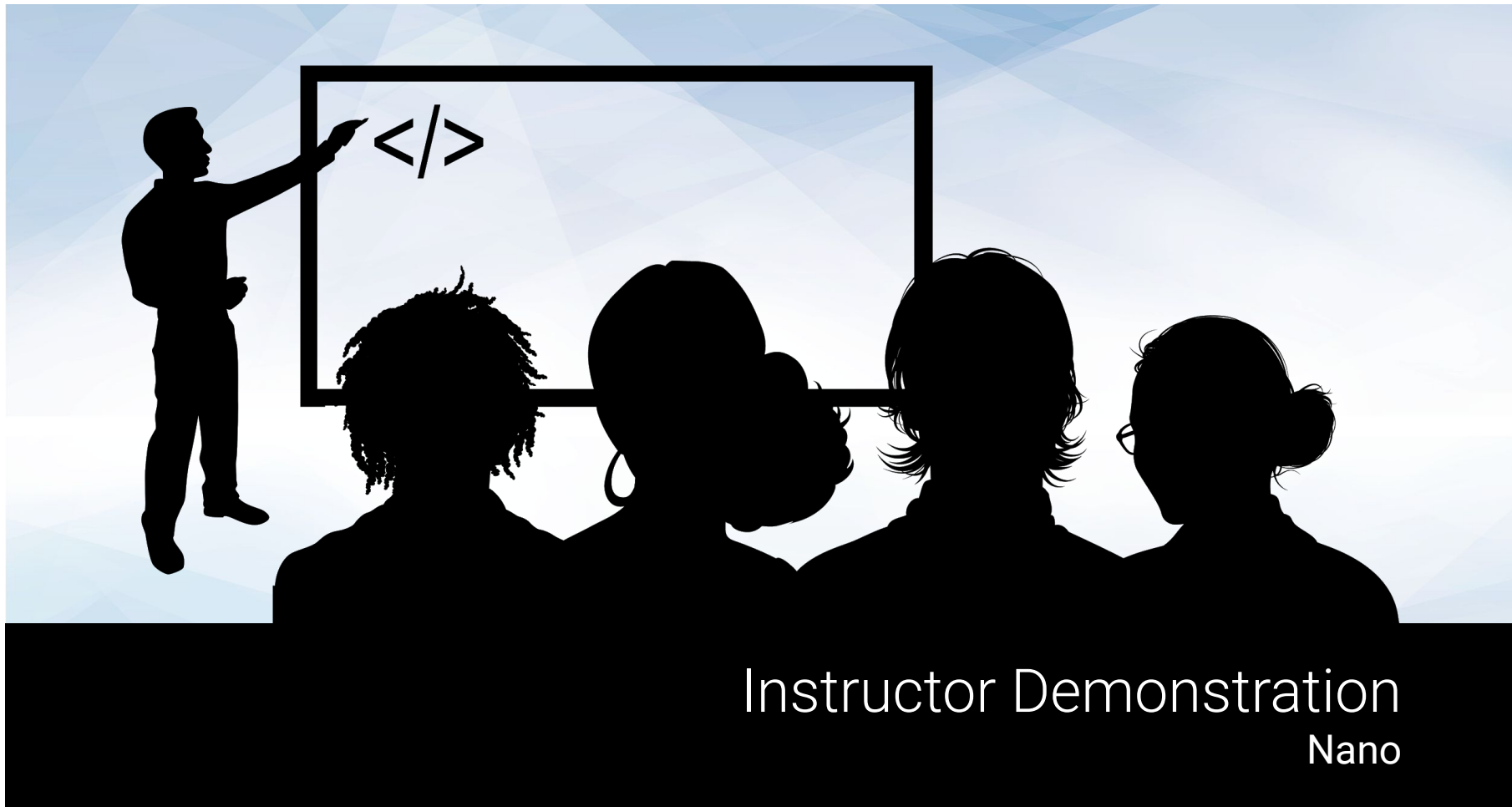
In the following demo, we will need to use Nano to create and edit the `/etc/motd` file.

`motd` stands for “message of the day” and will display a message to any user who logs in successfully.



We will also add a message to the `motd` file that warns users not to use or alter the system while it is down for maintenance.





Instructor Demonstration

Nano



# A Brief Introduction to Hashes and Password Cracking

# Password Hashes

---

A hash is a cryptographic function that takes data as input and translates it to a string of different, random-looking data.

## My Plain Text Password

ApPles20rang3s93

## My Password Hash

579de3a38386c62a1eca4600e3882b8b

A hash will always output the same string for the same input data.

# Password Hashes

The hash is stored in the shadow file.

When a user logs in, the hash of the submitted password is compared to the hash stored in `etc/shadow`.

-----

If the hashes match, the user's logged in.



# Password Cracking

---

A hash is a cryptographic function that takes data as input and translates it to a string of different, random-looking data.

## My Plain Text Password

ApPles20rang3s93

## My Password Hash

579de3a38386c62a1eca4600e3882b8b

Password cracking tools **cannot** reverse a password hash.

# Password Hashes

---

Password cracking tools **cannot** reverse a password hash.

Instead, they use a wordlist of potential passwords and create hashes for each one.

-----

This form of password hacking is called a **brute force attack**.







The more random and  
lengthy the password, the  
longer it will take to crack.

# How secure is my password?

---

Go to [howsecureismypassword.net](https://howsecureismypassword.net).

HOW SECURE IS MY PASSWORD?

ENTER PASSWORD

Sponsored by [Dashlane](#): never forget another password

# Secure Password Takeaways

---

01

If a system requires passwords of only sixteen characters, the password will be relatively strong, even if it contains words.

02

Add a few extra characters and it will become exponentially more secure.

03

In contrast, if using all random characters, a password still must be at least 10 characters to be very effective.

# Managing Passwords as Administrators

---

As administrators, we can do the following to control and secure passwords:



We may not be able to see a user's password, but we can see when they last reset it.



Making the password expire after a set time, so users will have to reset it.



We can also set length, number, and character requirements.



The only tools we need are **chage** (to change when a password expires) and **nano** (to edit the settings inside the `/etc/security/pwquality.conf` file).

# chage

---

`chage` allows us to change the number of days between password changes for each user.

The system will then use this information to determine when a user must change their password.

Some common options to use with `chage`:

<code>chage -l</code>	Shows the account aging info.
<code>chage -M</code>	Sets the number of days between password expirations.
<code>chage -W</code>	Sets the number of warnings the user gets before the change must be made.
<code>chage -d 0</code>	Forces an immediate password change.

# Demo Scenario

---

In the following demo, we will:



Change the maximum number of days between password changes to 90.



Change the number of days of warning before a password expires to 10.



Force Billy's current password to expire immediately.



Ensure that the new password differs from an old password by at least one character.



Ensure that the new password does not contain common dictionary words.



Set the minimum character length of new passwords to 12.



Set the minimum character types in a password to three.



# Instructor Demonstration

## Password Expiration Demo



## Activity: Let's Talk to John

In this activity, you will continue your role as a junior administrator auditing a system. Now, the focus is on passwords.

- You will use `john`, which we installed in the previous class, to crack the password hashes for all of the users on our system.
- Then, you will use the appropriate `chage` commands to set some password expirations and edit the `pwquality.conf` file to force stronger passwords.

**Suggested Time:**  
15 Minutes



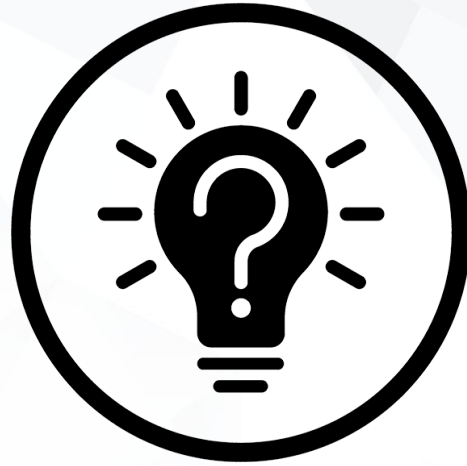




**Times Up!** Let's Review.



Privileges, root, sudo, and su



**We've used `sudo` for several commands in this unit.  
Why do you think some commands require  
`sudo` and others don't?**

# Privileges and Users

---

Why do we need to use sudo for some commands and not others?

## Users

Every file and program on a Linux system has permissions.

These permissions tell the system which users can access a file or run a program.

## Groups

Users can be placed in groups, which can have their own permissions.

## Root

File and program permissions apply to all users *except* the root.

The root user (or super user) has complete access to the system and can perform any task.

# Root Access

---

When attackers try to gain access to a system, they often try to gain **root access**.



Secure Linux systems do not allow anyone to log in as the root user on the system.



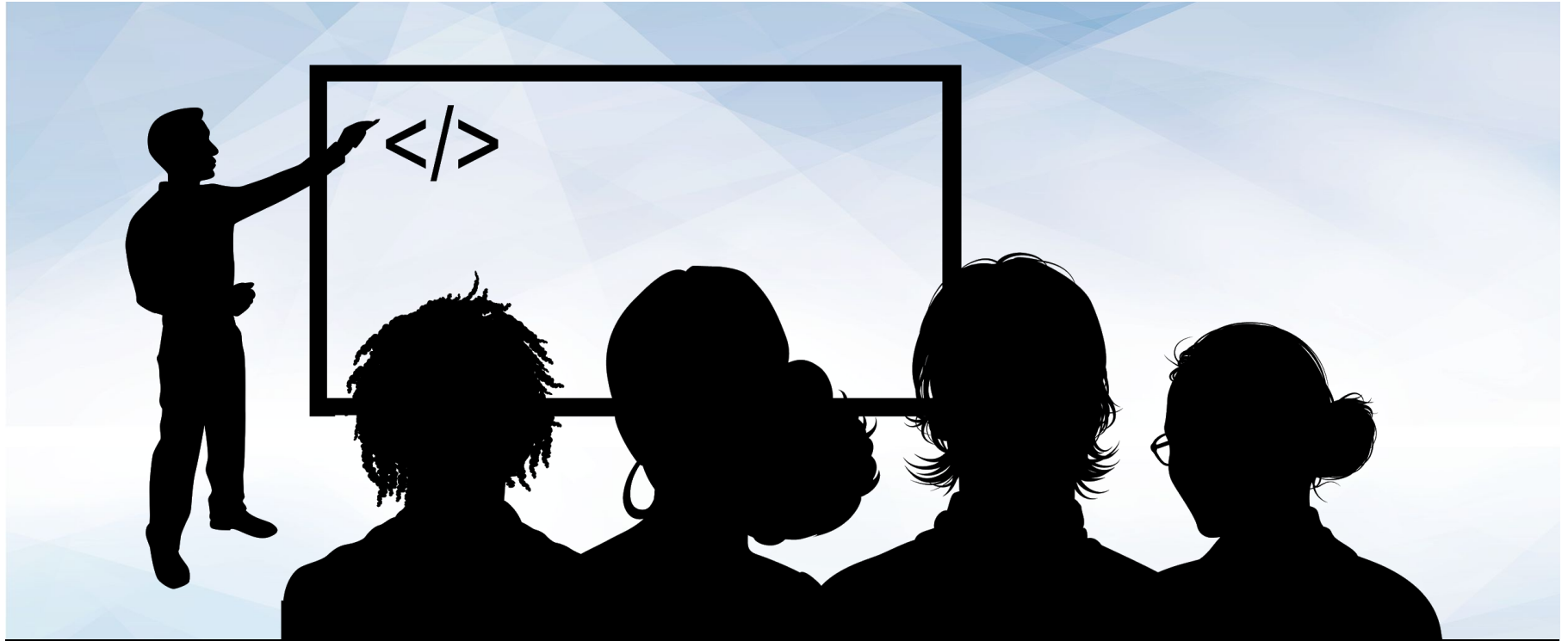
`sudo` (superuser do) can grant a user root privileges for one command.



When the one `sudo` command is done, the user is reverted to their normal access.



`sudo` can also control which commands the user can run as root.



# Instructor Demonstration

sudo

# Sudo Demo Summary

---

In the previous demonstration, we covered the following commands:

<code>whoami</code>	Determines the current user.
<code>su</code>	Switches to another user, in this case the root user.
<code>sudo</code>	Invokes the root user for one command only.
<code>sudo -l</code>	Lists the <code>sudo</code> privileges for a user.
<code>visudo</code>	Edits the <code>sudoers</code> file.



## Activity: Sudo Wrestling

In this activity, you will continue your role as a junior administrator auditing the system:

- The senior administrator has asked you to audit the system for sudo and root access, making sure no users other than the admin user have access to any sudo use.
- You must log in as each user, check their privileges, edit the sudoers file, and look for anything else suspicious.

**Suggested Time:**  
15 Minutes







**Times Up!** Let's Review.

# Sudo Wrestling Review

---

In order to complete this activity, we needed to:

01

View the `sudo` privileges for each user on the system.

02

Find a user that has `sudo` access for the `less` command.

03

Log in as that user.

04

Verify that the user has the ability to drop out of `less` and into a root shell.

05

Audit and edit the `sudoers` file to remove any access that shouldn't be there.

# Users and Groups

# Users and Groups

As noted, users on a Linux system can be added to groups.

Linux has the ability to create groups of users for functions like file and service sharing.

If a company has different departments, like Sales, Accounting, and Marketing, a Linux admin can create a group for each department. **Only users in each group can access files owned by the group.**



# Users and Groups

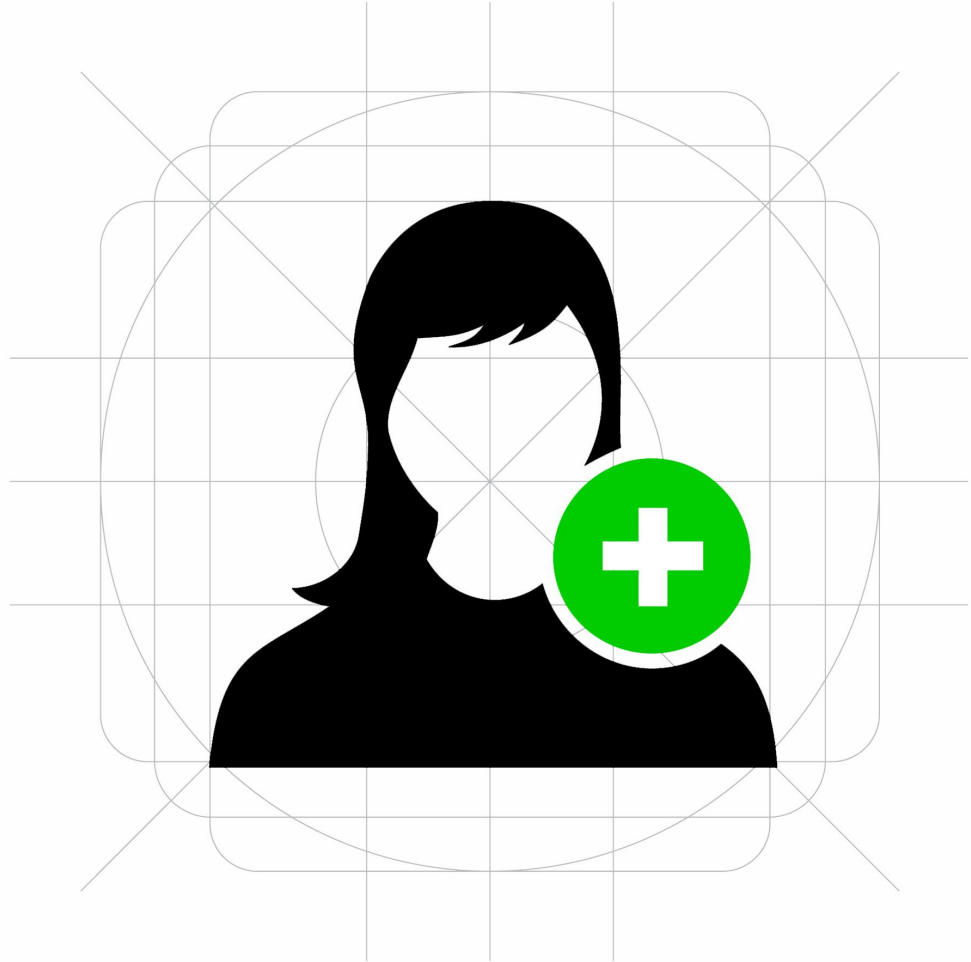
---

A system admin must know how to to add and remove users to a system, add and remove groups, and add or remove users from groups.

Soon we'll discuss commands specifically used for user and group management.



**But first, let's see how Linux identifies users and groups in the system.**



# id command

---

Linux identifies users and groups in the system using the **id** command:



Linux associates a specific number with each user, known as the **user ID (UID)**.



When Linux needs to identify a user, it uses the UID, not the username.



System users have a UID that is **less than 1000**.



Standard users have a UID that is **greater than 1000**.



The root user always has the UID of **0**.



Instructor Demonstration

id

# Users and Groups Demo Scenario

---

In the upcoming demo, we'll dive into more actions for user and group management using the following scenario:

**Your company recently made changes to the developer team.**



Mike, a lead developer,  
has left the company.



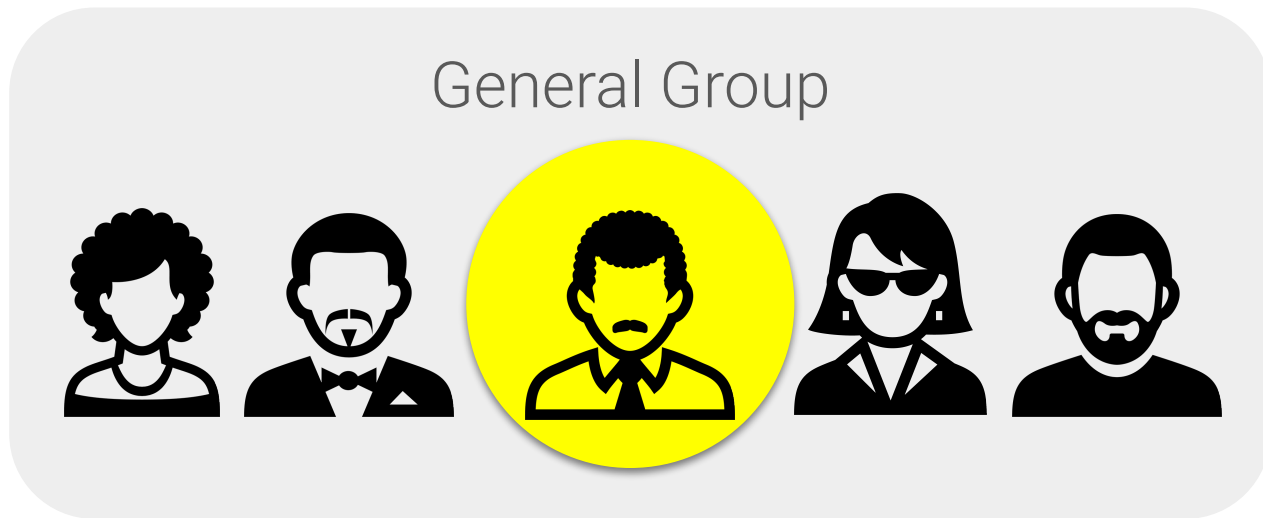
Joseph has joined as a  
new junior developer.



# Users and Groups Demo Scenario

---

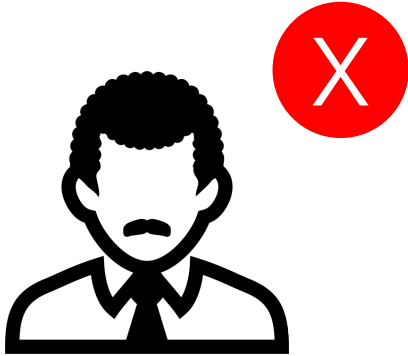
The company's Linux system has never been set up properly with a developers group. Instead, **Mike was part of the general group.**



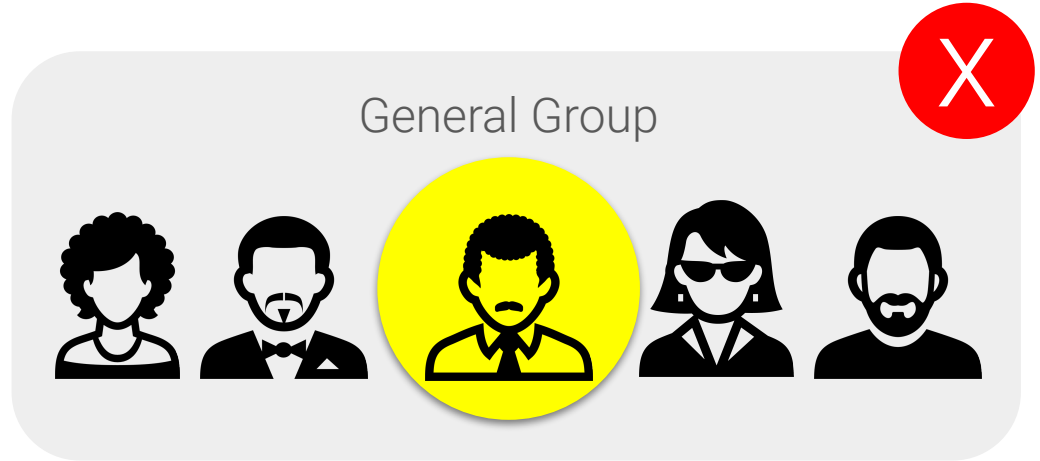
# Users and Groups Demo Scenario

---

As the sysadmin for this system, you need to remove Mike from the general group, remove the general group, and delete Mike from the system.



Mike, a lead developer, has left the company.



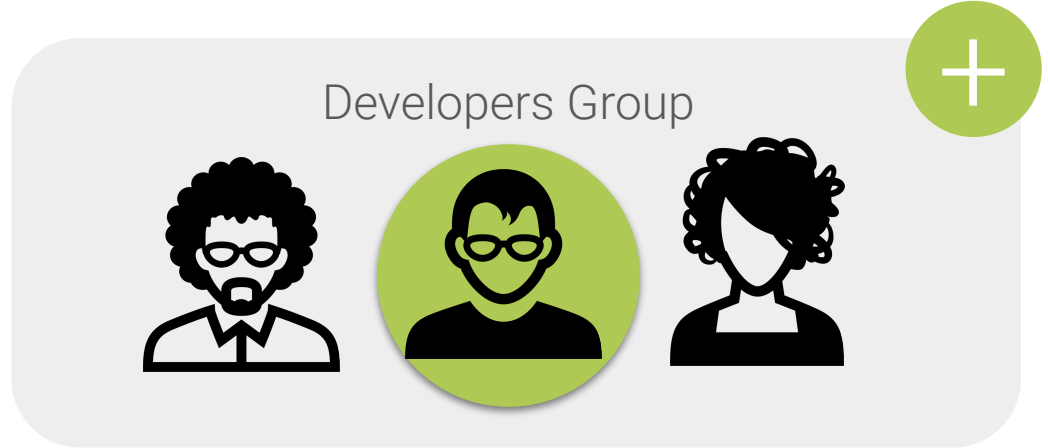
# Users and Groups Demo Scenario

---

Then, you need to add Joseph to the system, create a developers group, and add Joseph to this group.







Joseph has joined as a new junior developer.



# Users and Groups Scenario

To accomplish these tasks, we will use the following commands:

<code>groups</code>	Get group info for the user <code>mike</code> .
<code>usermod</code>	Lock Mike's account to prevent him from logging in.
<code>usermod</code>	Remove the user <code>mike</code> from the <code>general</code> group.
<code>deluser --remove-home</code>	Delete the user <code>mike</code> .  
<code>delgroup</code>	Delete the <code>general</code> group.
<code>adduser</code>	Create the user <code>joseph</code> .  
<code>addgroup</code>	Create a <code>developer</code> group.
<code>usermod</code>	Add the user <code>joseph</code> to the <code>developer</code> group.



# Instructor Demonstration

## Users and Groups



## **Activity:** Users and Groups

Your senior administrator has asked you to audit all the users and groups on the system.

- You must create a new group for the standard users and remove users from the sudo group.
- In the previous activity, you found some malicious users. Now, you will remove them from the system entirely.

**Suggested Time:**  
15 Minutes





**Times Up!** Let's Review.

# Users and Groups Review

---

To complete this activity, we had to:

01

Check every user's UID and GID.

02

Make sure that only the sysadmin account belonged to the sudoers group.

03

Remove any users from the system that shouldn't have been there.

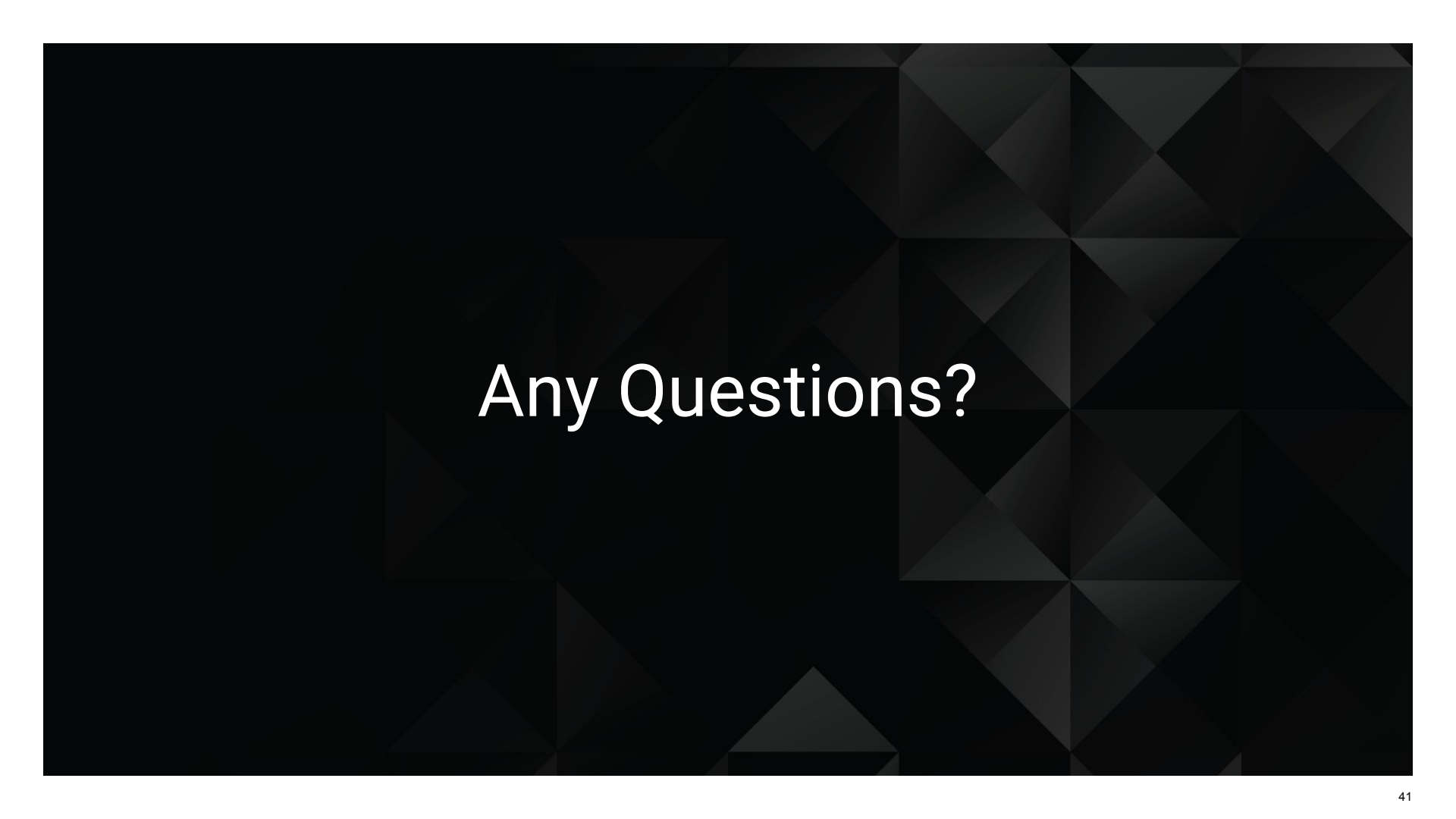
04

Verify that all non-admin users were part of the group developers.

05

Remove the suspicious hax0rs group.





Any Questions?