Special Permissions and Managing Services

Cybersecurity

Linux 1 Day 3

Class Objectives

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

Inspect and set file permissions for sensitive files on the system.

Set file permissions with the **`SUID`** or **`GUID`** bits.

Manage and monitor services on the system, and remove unused services.

Create and assign users for services.

Access Controls

Like Google Docs,
Linux has **access controls**,
granting permission to access
documents and files on a host.

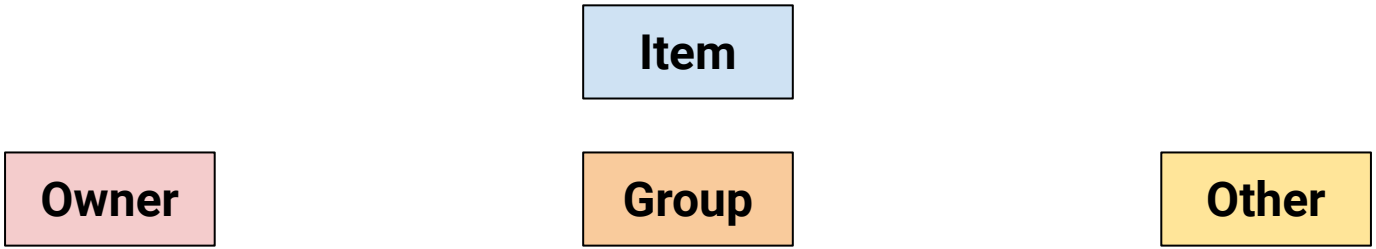
Managing Access Controls in Linux

Linux treats users, programs, and files as **items**.

Item

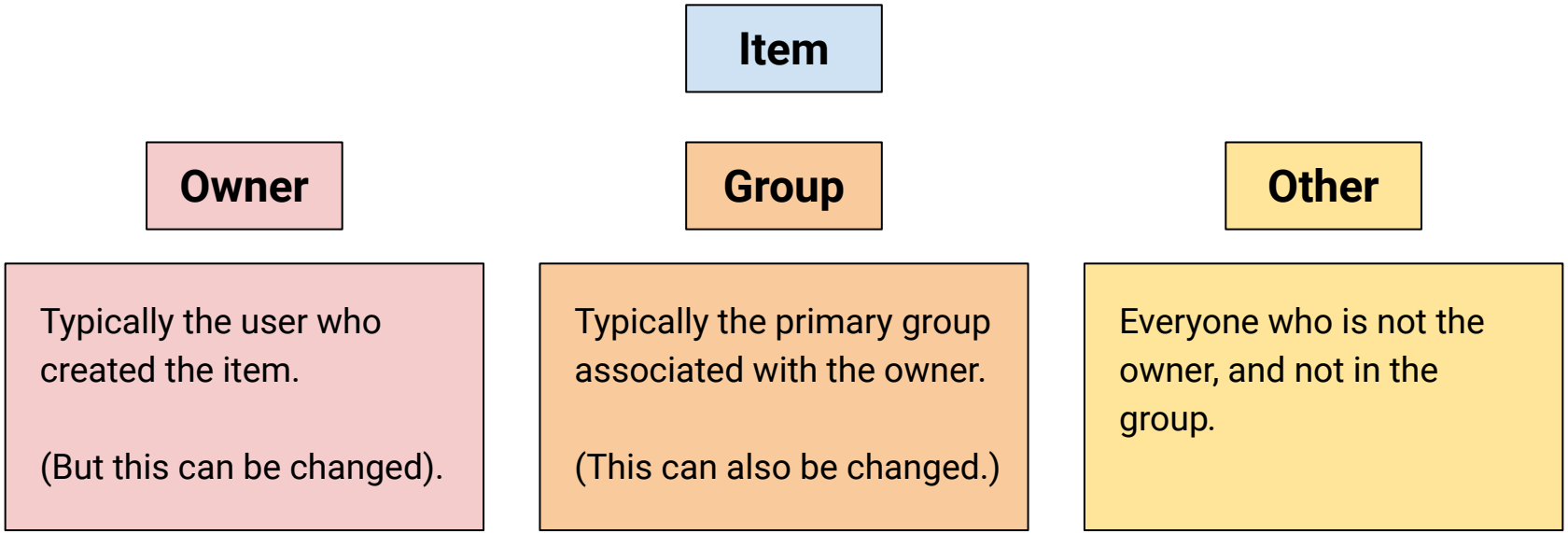
Managing Access Controls in Linux

Managing access controls in Linux involves assigning privileges to the *owner* of the item, the *group* associated with the item, and *others*.



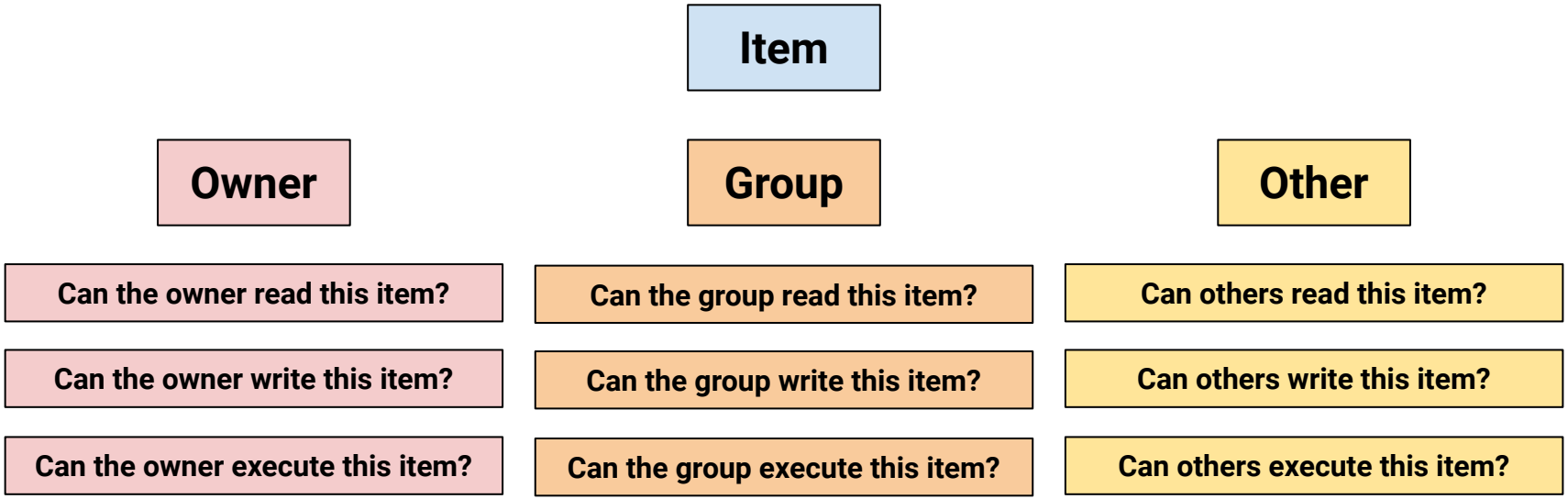
Managing Access Controls in Linux

Managing access controls in Linux involves assigning *owner* of the item, the *group* associated with the item, and *others*.



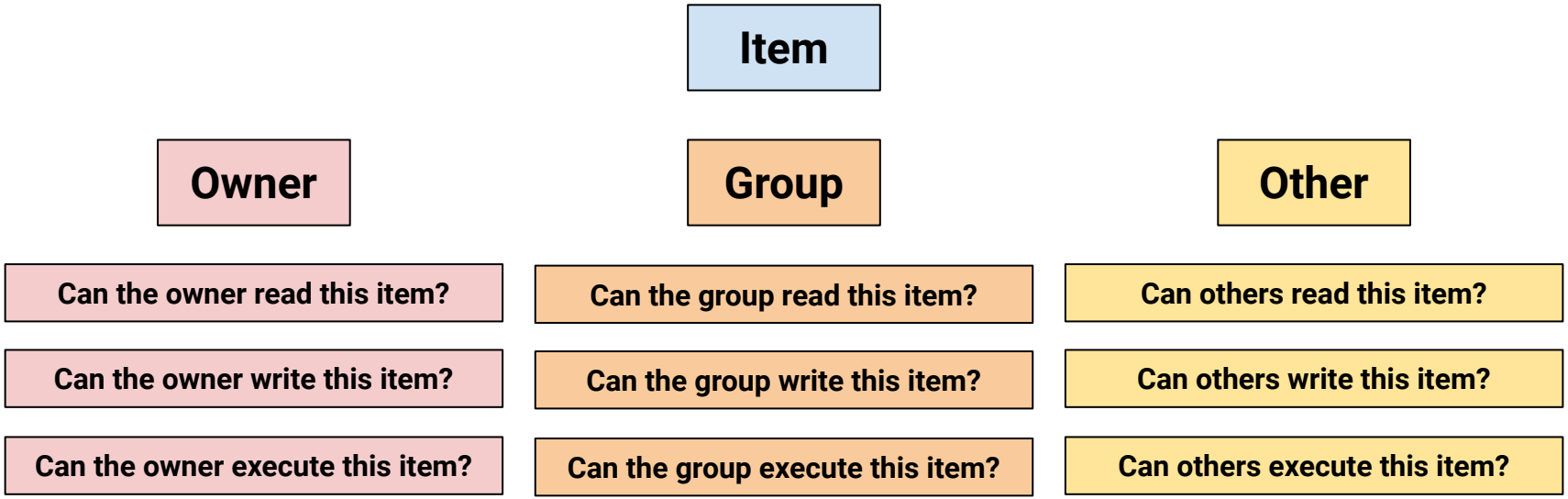
Managing Access Controls in Linux

For each of these categories, there are three actions that we can allow or prevent: **read, write, execute.**



Managing Access Controls in Linux

Assigning these permissions is called **Discretionary Access Control (DAC).**
It is *discretionary* because permissions can pass from one item to another.

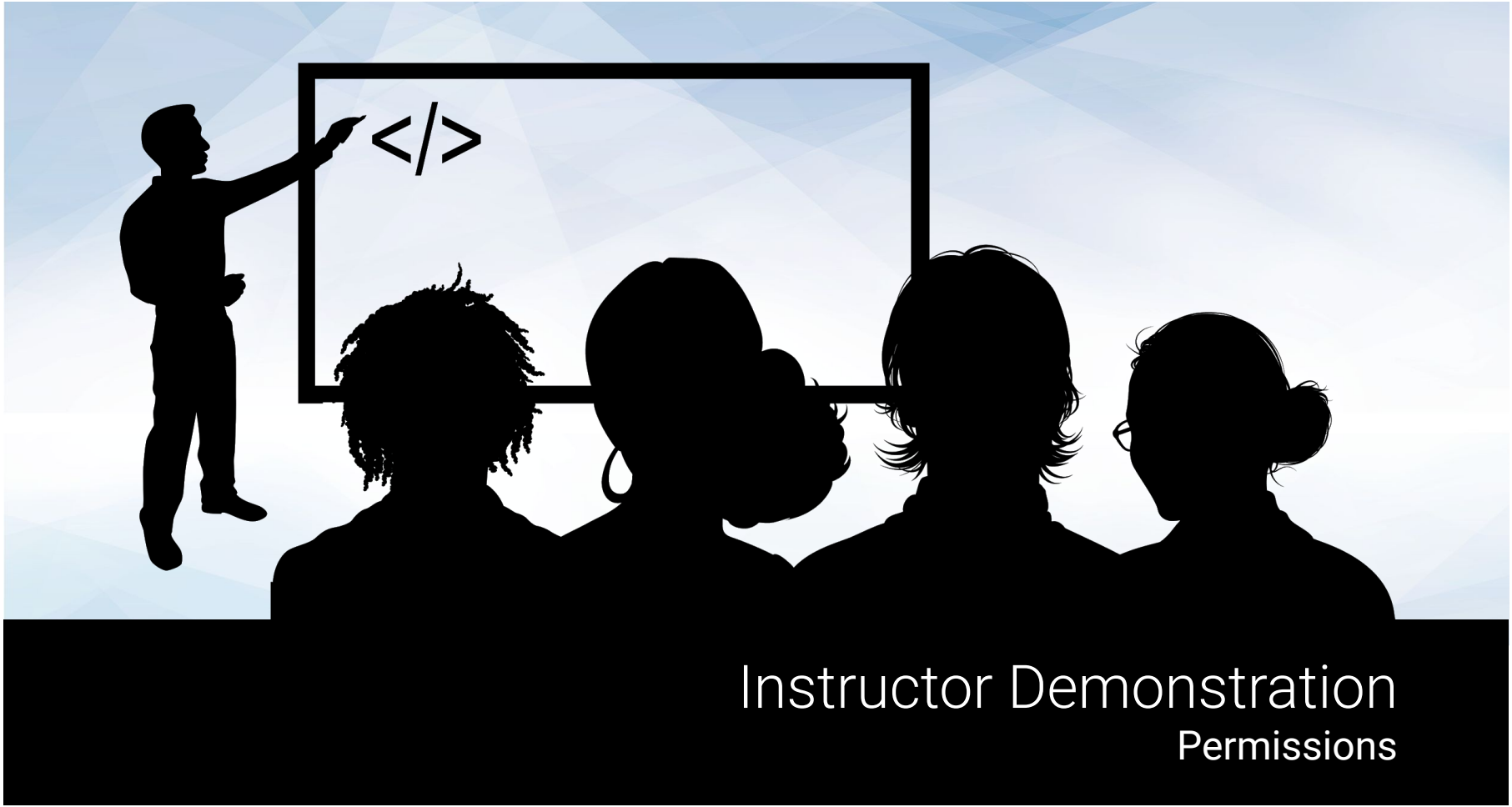


Permissions Demo

In the upcoming demo, we'll create a file and a directory, observing default permissions. Then, we will change the permissions to deny certain groups and users access.

To read and manipulate these file permissions, we'll use these commands:

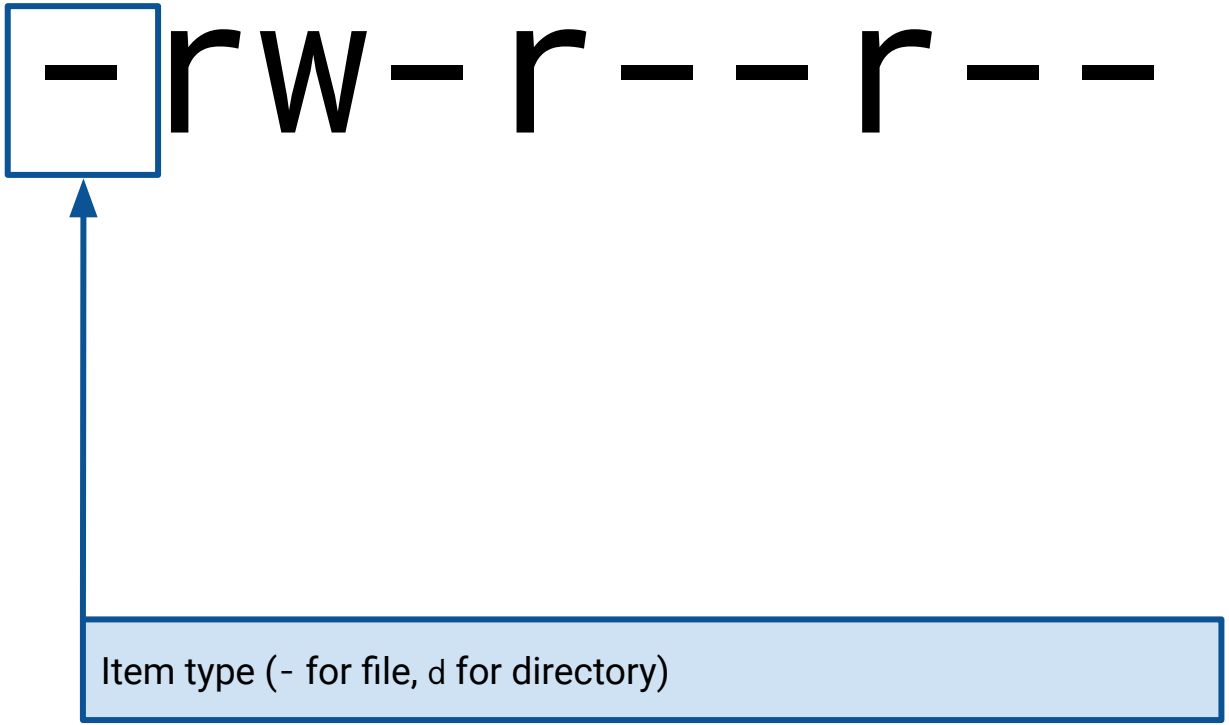
`ls -l`	Show the permissions info.
`chmod`	Change the permissions info.
`chown`	Change the owner and group of a file.



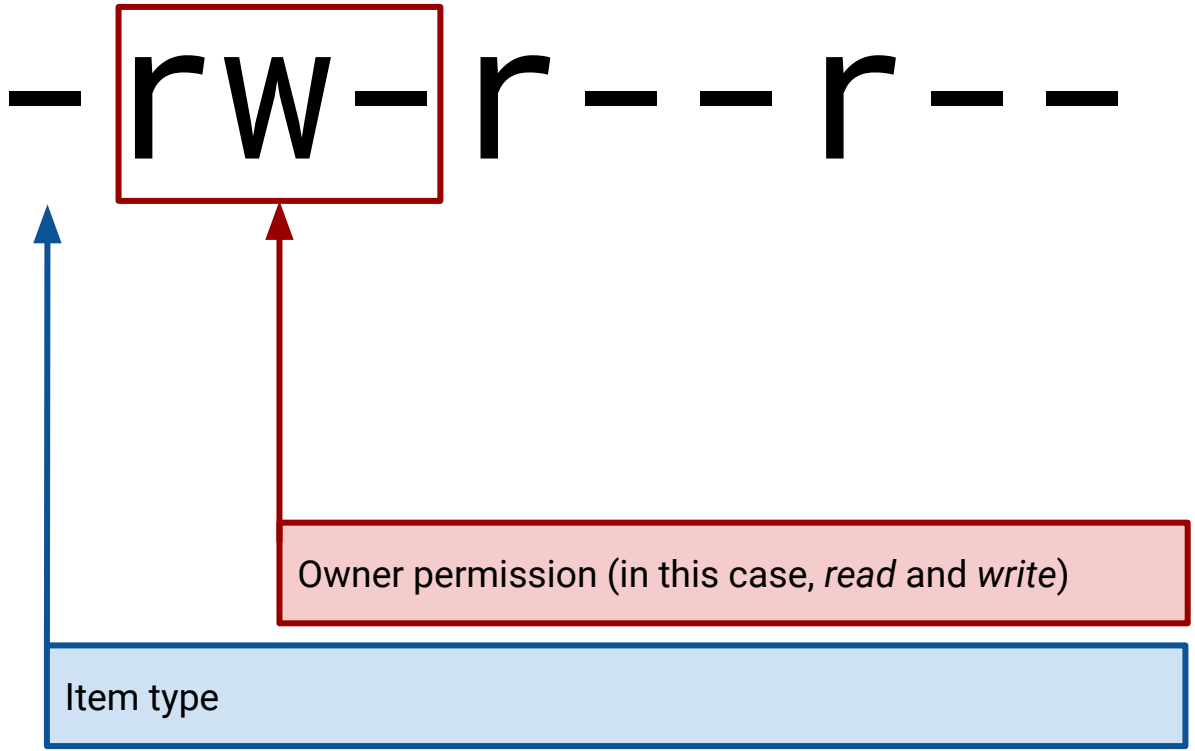
Instructor Demonstration

Permissions

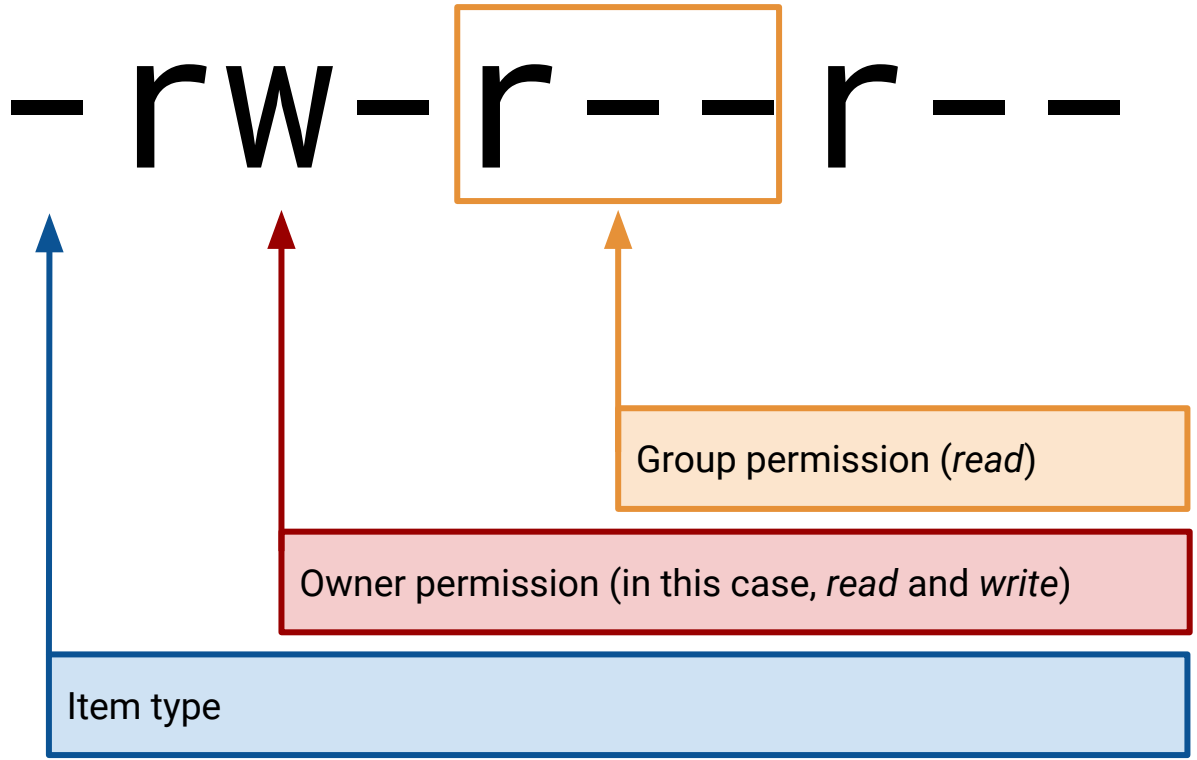
Inspecting File Permissions



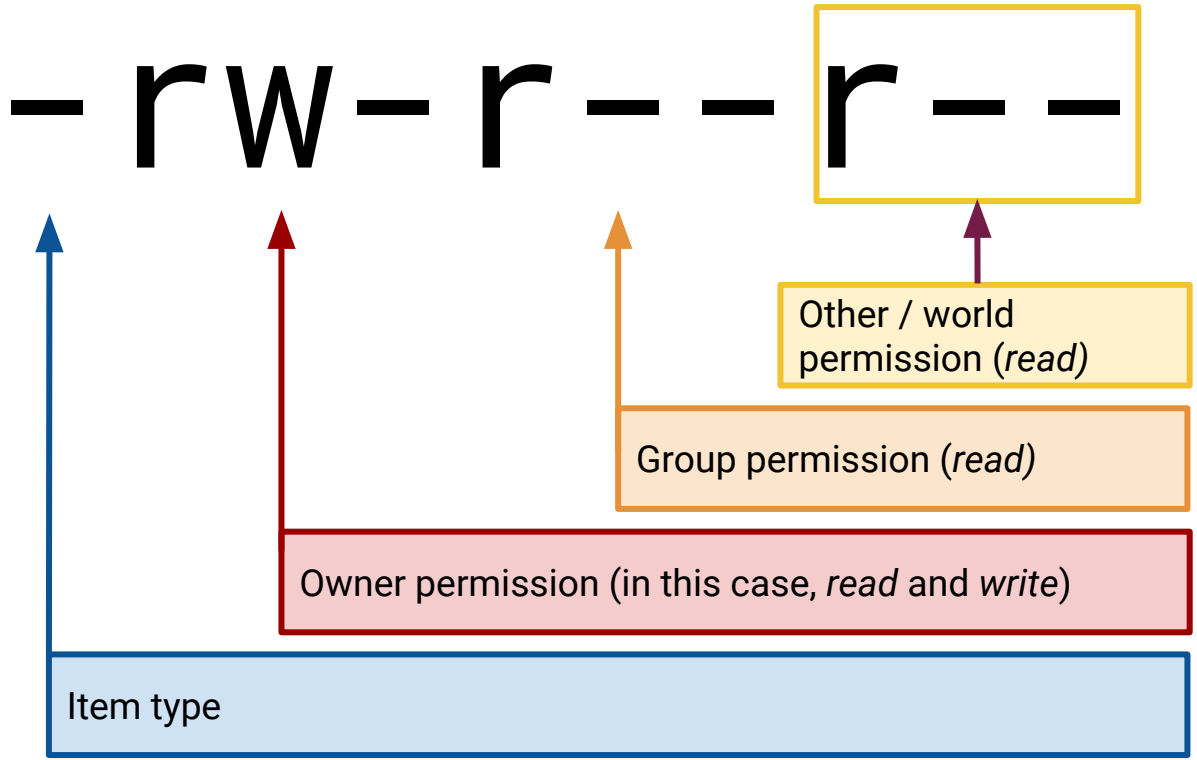
Inspecting File Permissions



Inspecting File Permissions



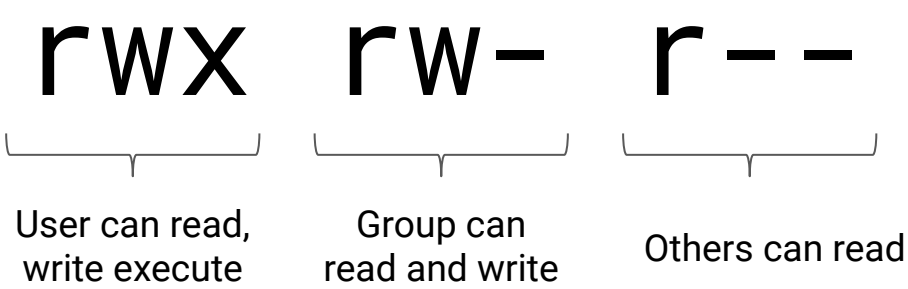
Inspecting File Permissions



Changing File Permissions

File permissions can be set using two different notations: **symbolic** and octal.

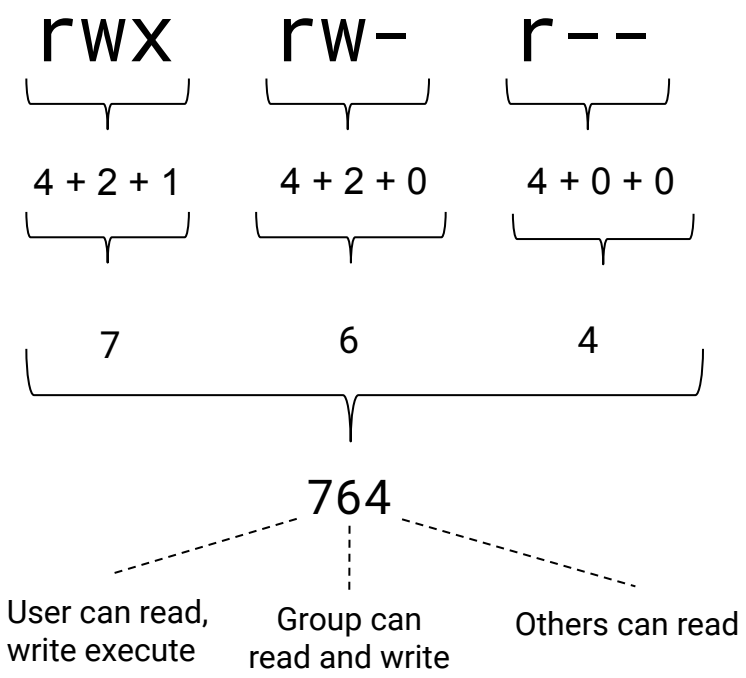
Symbolic Notation	
`r`	read
`w`	write
`x`	execute

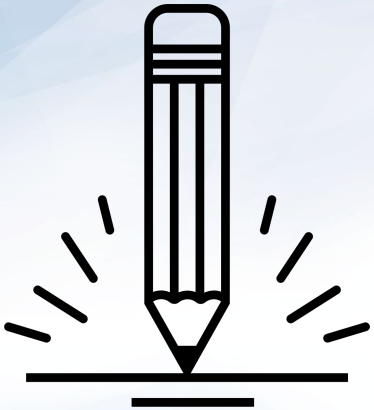


Changing File Permissions

File permissions can be set using two different notations: symbolic and **octal**.

Octal Notation				
	4	2	1	
0	-	-	-	No permission
1	-	-	x	Only execute
2	-	w	-	Only write
3	-	w	x	Write and execute
4	r	-	-	Only read
5	r	-	x	Read and execute
6	r	w	-	Read and write
7	r	w	x	Read, write, and execute





Activity: Access Controls and Permissions

In this activity, you will inspect and set file permissions on a few of the most sensitive items on a Linux system.

- In the previous activity, you cleaned up the system groups and removed all the malicious users from the system.
- Now you need to make sure a few items are locked down by checking and modifying their permissions.

Suggested Time:
15 Minutes

Times Up! Let’s Review.

Access Controls and Permissions Review

In order to complete this activity, we needed to:

01 Set permissions `600` on `/etc/shadow` (`rw` for `root` only.)

02 Set permissions `600` on `/etc/gshadow` (`rw` for `root` only.)

03 Set permissions `644` on `/etc/group` (`rw` for `root` and `r` for everyone else.)

04 Set permissions `644` on `/etc/passwd` (`rw` for `root` and `r` for everyone else.)

05 Verify all accounts have passwords.

06 Verify that no users other than `root` have UIDs of `0`.

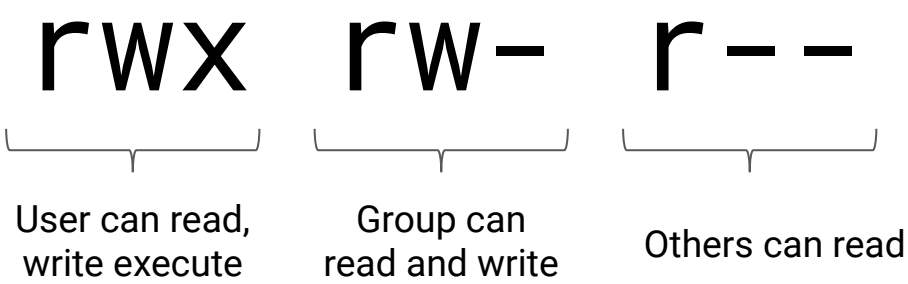
07 Provide a list of your findings in the research directory.

To recap

Permissions

How permissions apply to each specific file and folder with `r`, `w`, and `x`.

Symbolic Notation	
`r`	read
`w`	write
`x`	execute



Permissions

How to view and apply permissions to an item's user, group, and other.

Users

Every file and program on a Linux system has permissions.

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

Groups

Users can be placed in groups, which can have special permissions that apply to all members of the group.

Root

File and program permissions apply to all users in a system, *except* the root user.

The root user (or super user) has complete access and can perform any action.

Permissions

We can use **sudo** user to invoke the **root** user and bypass any permissions.

`ls -l`	To show the permissions info.
`chmod`	To change the permissions info.
`chown`	To change the owner and group of a file.

Permissions

We can assign **sudo** for a specific command for a specific user.

`whoami`	To determine the current user.
`su`	To switch to another user, in this case the root user.
`sudo`	To invoke the root user for one command only.
`sudo -l`	To list the `sudo` privileges for a user.
`visudo`	To edit the `sudoers` file.

Special Bits

Can anyone think of a scenario where all the users on a system would need to run a specific program as `root`?

Password Command

The **passwd** command allows a user to change their password.

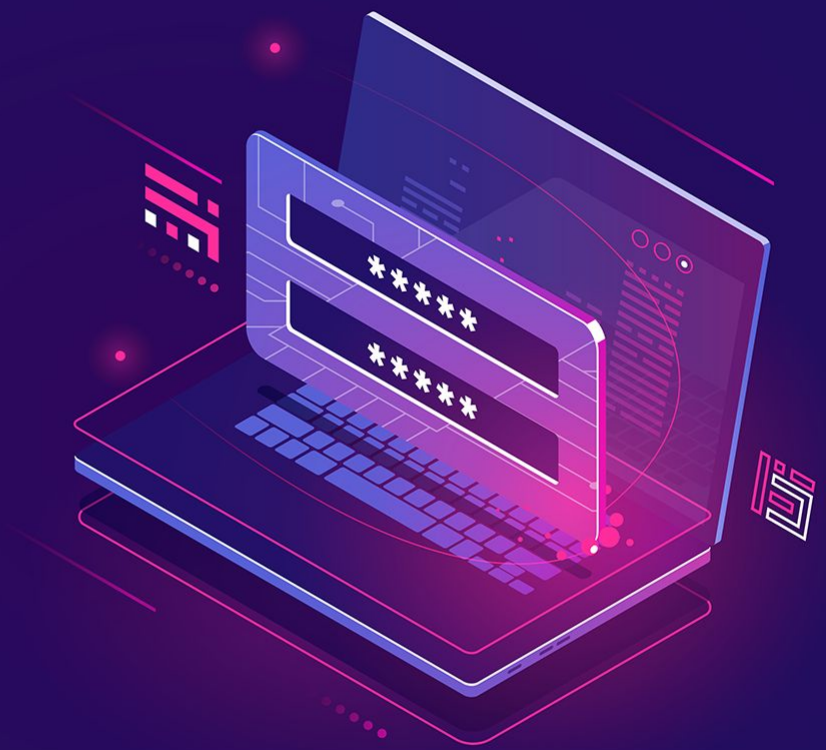
`/etc/shadow`	In order to change a password in the system, the `passwd` program has to edit the `/etc/shadow` file where the password hashes are stored.

passwd must have root privileges in order to edit the **/etc/shadow** file and update the password hash.

Remember, we never edit the **/etc/shadow** file manually. It is only edited by the system when a user is created or removed, or when a user's password is changed and it is owned by the **root** user.

Password Command

Rather than manually giving all users `sudo` access for this command and making sure every new user also gets `sudo` access, we can have `passwd` set to always run on `root`, regardless of the user running the command.



Special Permissions

We use special permissions for three basic scenarios:

01

Allowing multiple users to make changes to the same files inside a shared directory.

02

Limiting users to only making changes to their own files within a shared directory.

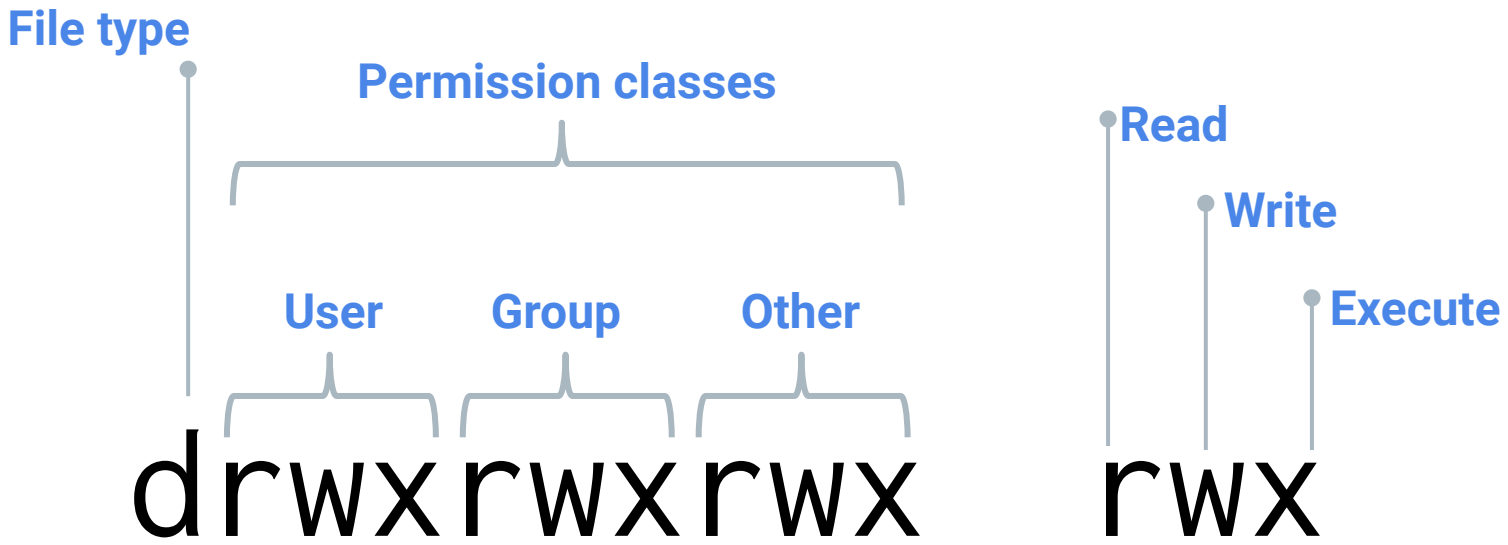
03

Allowing any user on the system to run a specific program as the root user.

For each of these three scenarios, there is a different **special bit** setting that we can add to an item's permissions.

Special Permissions

The `r`, `w`, and `x` bits are the standard characters for permissions. The **special bits** are those we add to permissions for the three scenarios mentioned before.



Special Permissions

Special bits are displayed alongside the permissions of an item, in the place of the **execute** bit.

- In these cases, the **x** setting is still present, it's just not displayed.
- The special bit and the execute settings are used in conjunction with each other. To use a special bit, we would first allow execute permissions before setting the special bit.
- There are three special bit options, one for the owner, group, and other permissions.

Special Permissions

We can enable or disable these bits by using the same **chmod** command we learned in the last class. We can use either symbolic or octal notation.

`ls -l`	To show the permissions info.
`chmod`	To change the permissions info.
`chown`	To change the owner and group of a file.

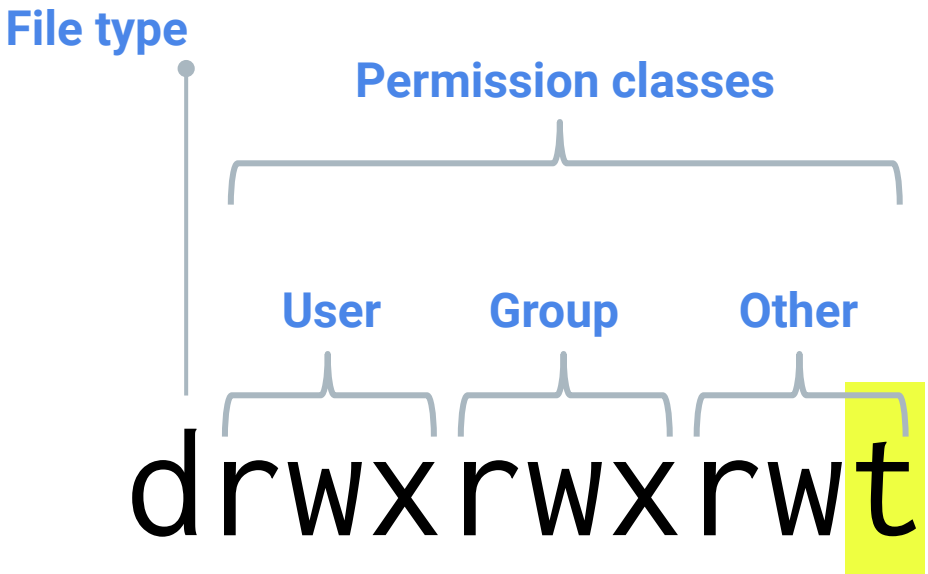
The Sticky bit lives in the execute position for the *other* permissions.

Sticky Bit

Restricts users from making changes to other users' files inside a shared directory.

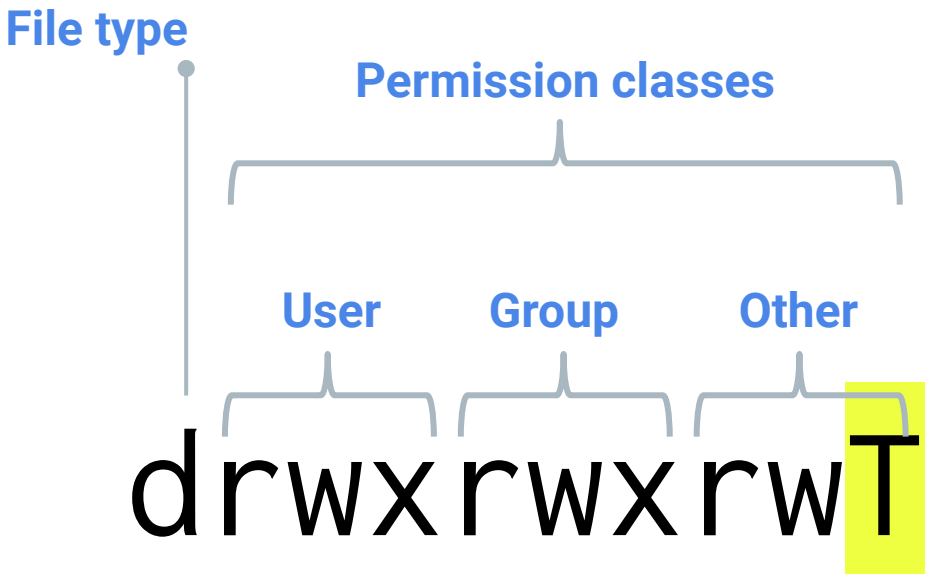
Sticky Bit

Displayed with a **`t`** in the place of the **`x`** for the *other* permissions: **`drwxrwxrwt`**.



Sticky Bit

If set without setting the `x` first, it will be indicated with a capital `T`: **`drwxrwxrwT`**.



Sticky Bit

Can be set with **`chmod o+t`** or **`chmod 1000.`**

```
# mkdir es_data
# mkdir es-data
# chmod o+t es_data
# chmod 1000 es-data


# ls -l
d--------T 2 1000 1000 4096 Jun  8 09:50 es-data
drwxr-xr-t 8 root root 4096 Jun  8 09:50 es_data
```

The Set Group ID (**`SGID`**) bit lives in the execute position for the *group* permissions.

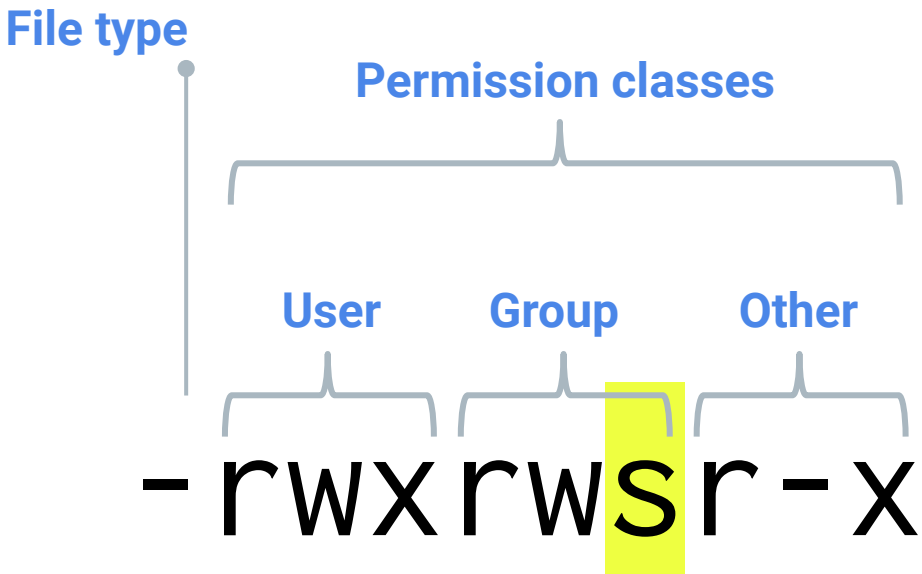
Set Group ID

Grants all users the same permissions as the group assigned to a shared directory.

Designed to allow multiple users to make changes to the same directory.

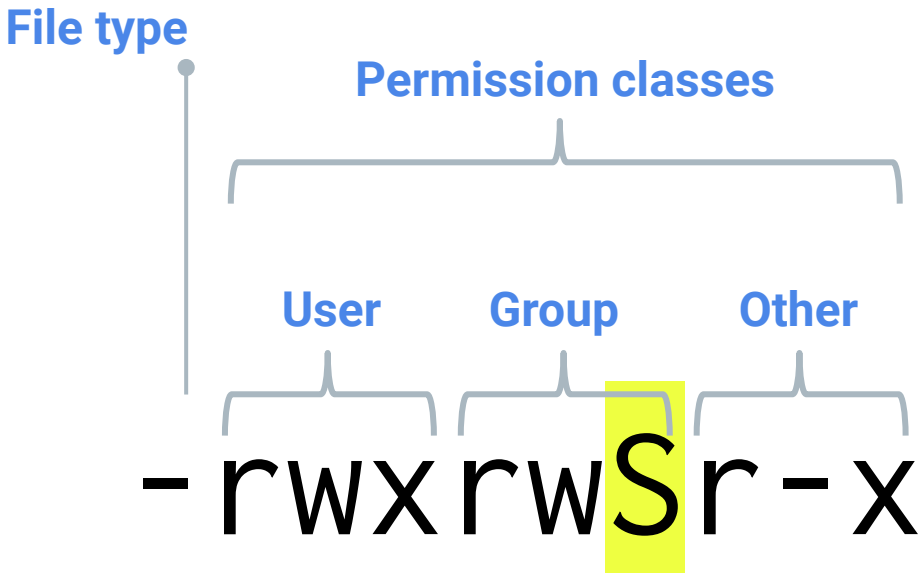
Set Group ID

Displayed with an **`s`** in the place of the **`x`** for the *group* permissions: **`-rwxrwsr-x.`**



Set Group ID

If set *without* setting the `x` first, it will be indicated with a capital `S`: `-rwxrwSr-x`.



Set Group ID

It can be set with **`chmod g+s`** or **`chmod 2000.`**

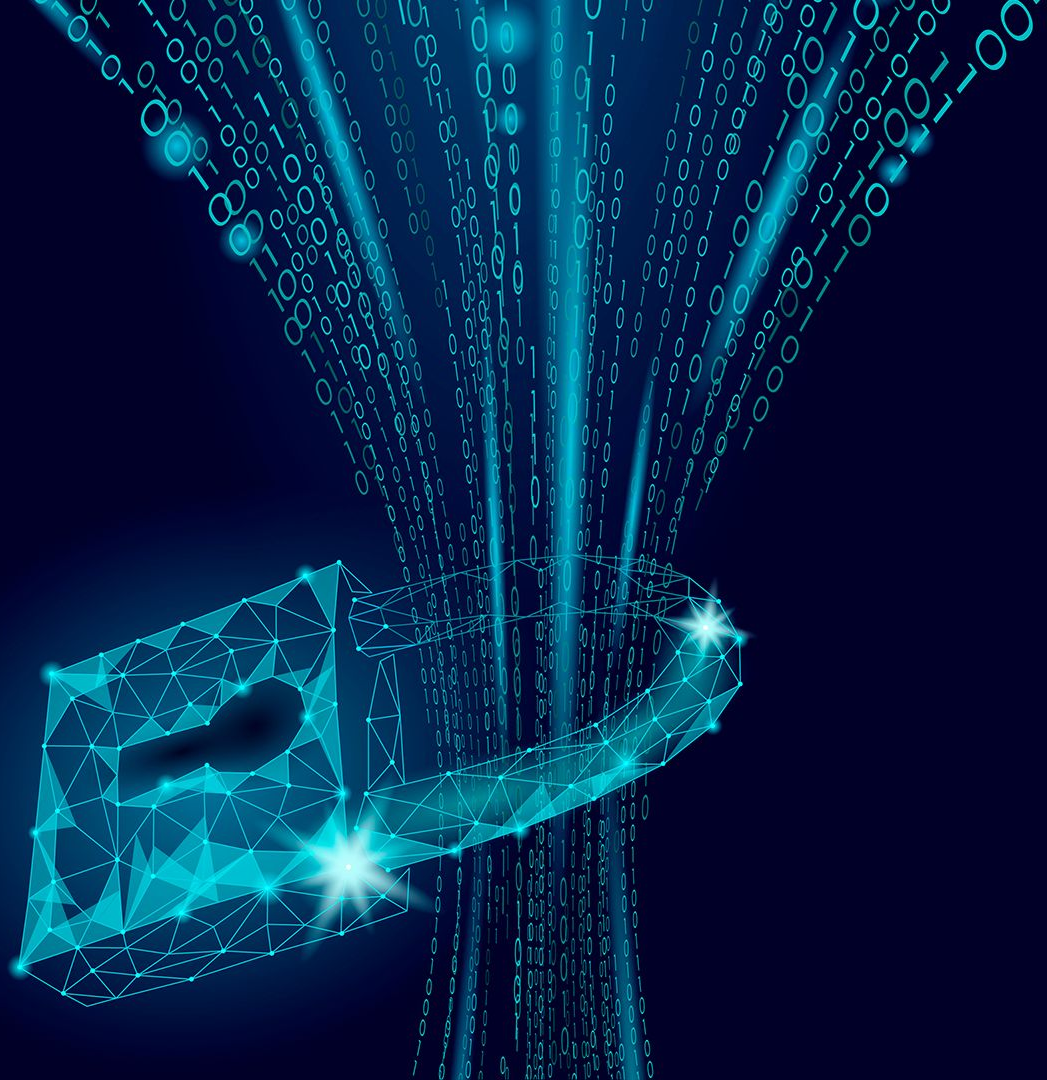
```
# mkdir es_data
# mkdir es-data
# chmod g+s es_data
# chmod 2000 es-data


# ls -l
d-----S--- 2 root root 64 Jun  8 09:50 es-data
drwxr-sr-x 8 root root 64 Jun  8 09:50 es_data
```

The `SGID` and sticky bits can be used together to allow a group of users to collaborate in the same directory.

When set together, they allow users to make changes to the same directory, but only to files they own.

The `SUID` bit is the most important bit for security.



Special Bits Demo

A web developer at your company asked for **`sudo`** access to run the **`find`** command on a web server. They want to search the entire system for misplaced configuration files.

They also said that other developers want the same privilege, and suggested setting the **`SUID`** bit on the **`find`** command so everyone can use **`find`** with root privileges.

Special Bits Demo

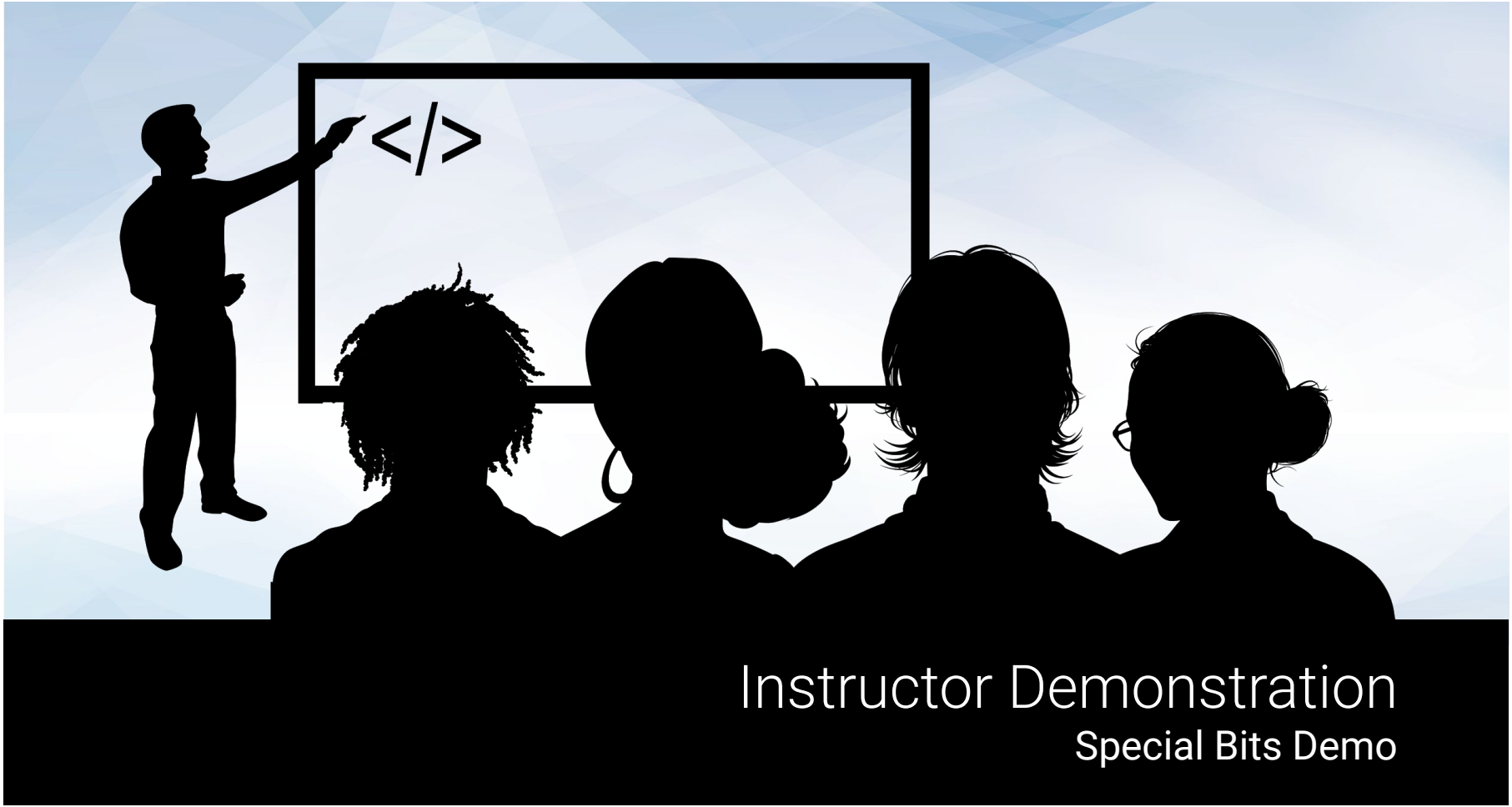
In the following demo, we will:

Turn on the **`SUID`** setting of the **`find`** command.

Confirm that this introduces a vulnerability to the system.

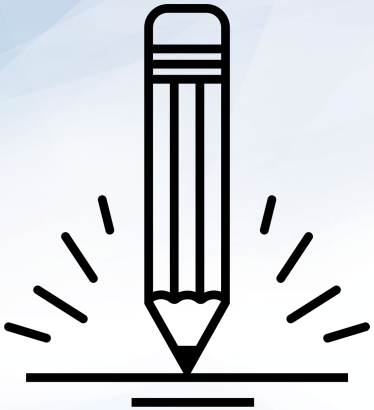
Turn off the **`SUID`** setting for the **`find`** command.

Search the system for other programs that have the **`SUID`** bit set.



Instructor Demonstration

Special Bits Demo



Activity: Shutting Out `SUID`

Your senior administrator is worried about the `SUID` bit being set on programs owned by `root`, and has asked that you audit the system for all files that have the `SUID` bit set for the root user.

You must audit the system for known programs that should not have this bit set.

Suggested Time:
15 minutes

Time’s Up! Let’s Review.

Shutting Out `SUID` Review

Completing the previous activity required the following steps:

01 Running the **`find`** command to **`find`** all **`SUID`** bit-enabled programs.

02 Comparing the lists and verifying one of the programs found is not on the first list.

03 Switching to another user and attempt to exploit the program found.

04 Running the correct command to remove the **`SUID`** bit and verifying this worked.


Break

Managing Services


Servers are computers that offer services to other computers.

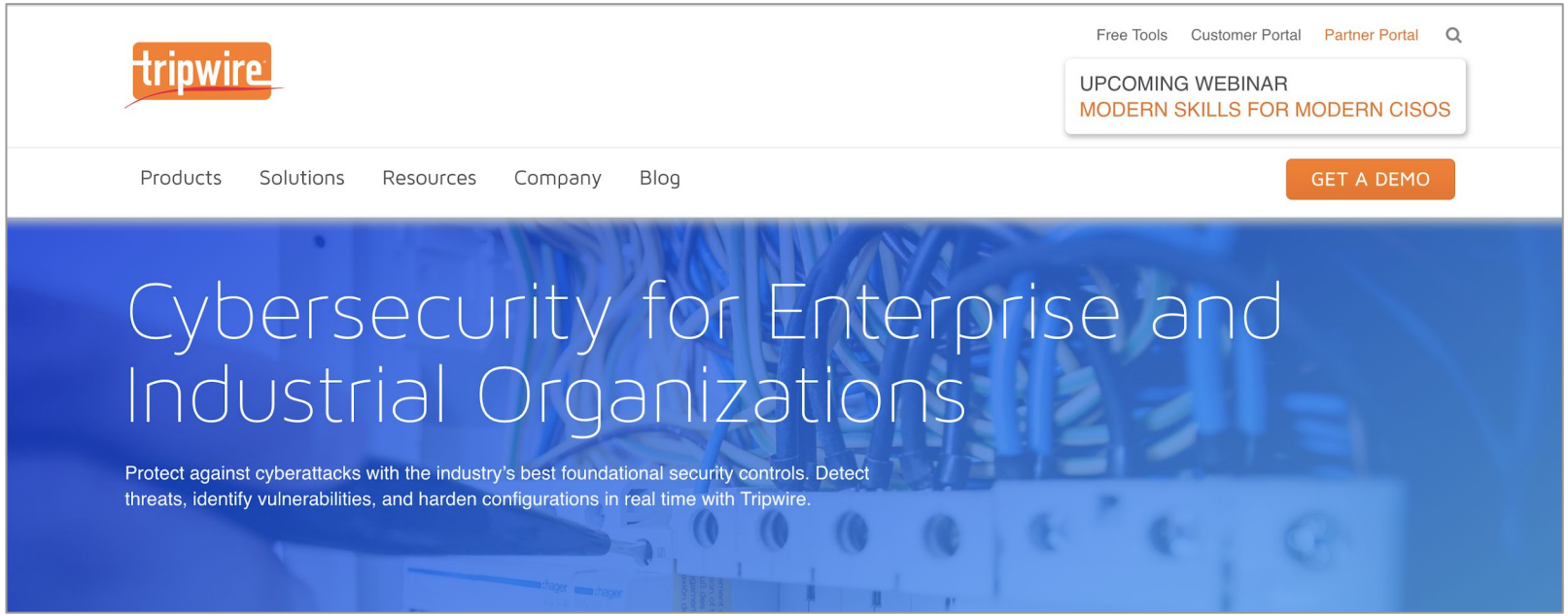
Managing Services

A service is a function or capability that a machine makes available to another.

For example, file sharing services allow computers to send and receive data.

Managing Services

Some services, like Tripwire, are only run locally on the server and are not provided to other computers. These services are packages that can be installed and removed just like other programs.



Services and Security

Services and Security

Attackers can manipulate services into doing things that they are not designed to do.

Services and Security

For example: Samba (SMB), the file sharing protocol, allows users to view, download, and store files remotely.

Services and Security

If a malicious user is able to gain access to a shared folder, they can exfiltrate, alter, or delete sensitive files.

Finding and Stopping SMB Demo

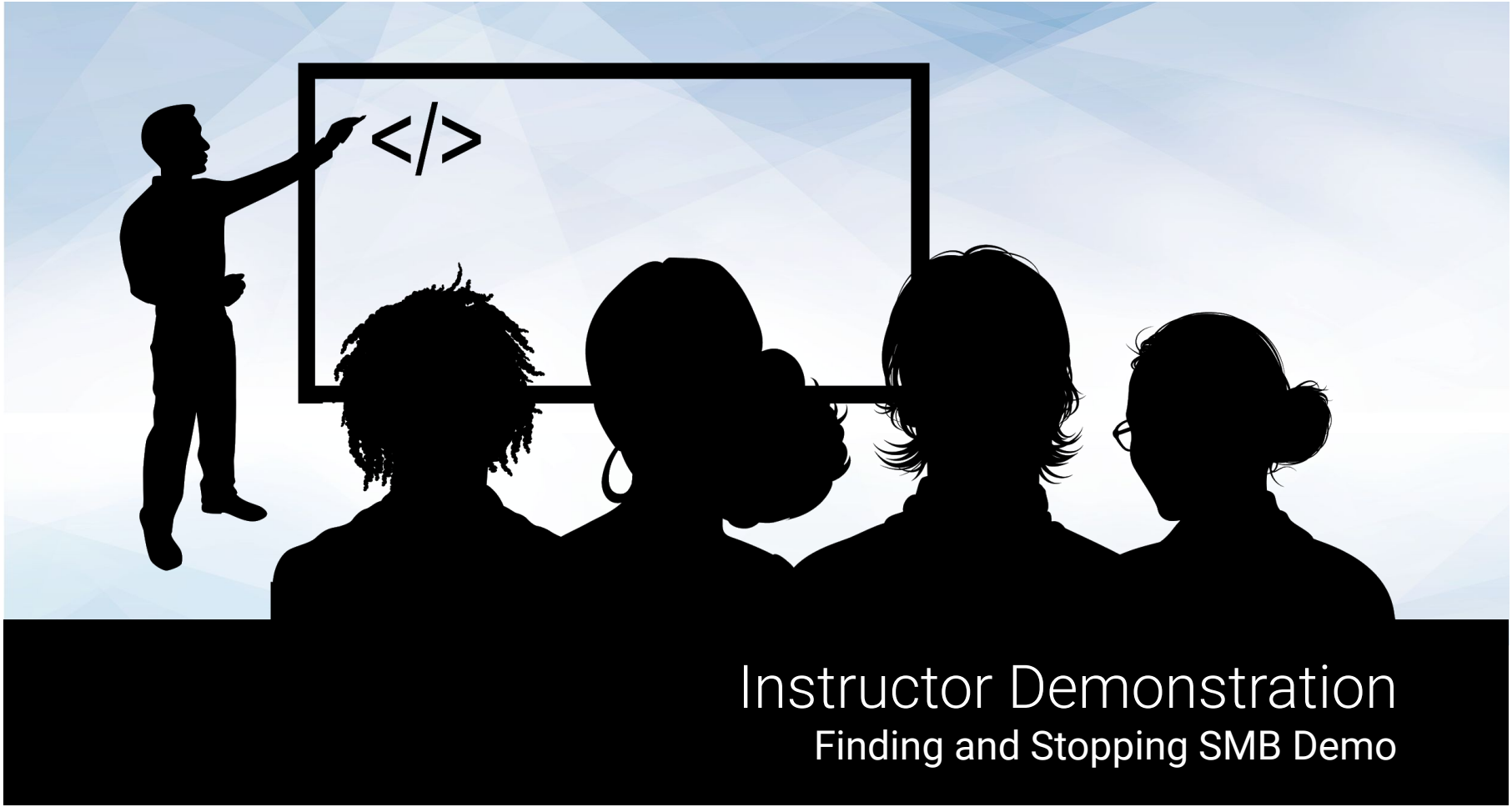
In this example, the server has already been compromised.

In the following demo, we will stop the SMB service, and then uninstall it from the system.

Finding and Stopping SMB Demo

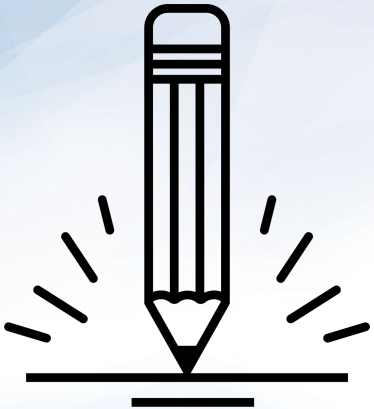
This will require the following steps:

- Listing all running services.
- Identifying the Samba service in the list to confirm it's running, then stopping it.
- Ensuring Samba doesn't start when the machine is started up.
- Ensuring Samba is no longer running.
- Uninstalling the Samba service completely.



Instructor Demonstration

Finding and Stopping SMB Demo



Activity: Managing Services

Your senior administrator wants you to audit the services being run by the server and shut down old and unused services.

They have provided a list of services to check.

Check this list of services, and shut down and remove any service that you find running.

Suggested Time:
15 minutes

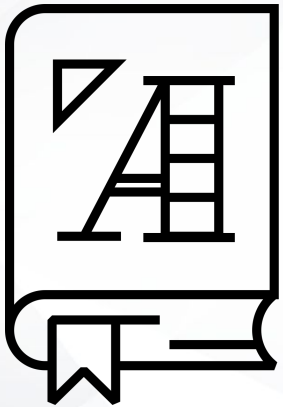
Time’s Up! Let’s Review.

Review: Managing Services

Completing this activity required the following steps:

- Identifying four services in the list that are installed/running on the machine.
- Stopping each service.
- Disabling each service.
- Uninstalling each service.

Service Users

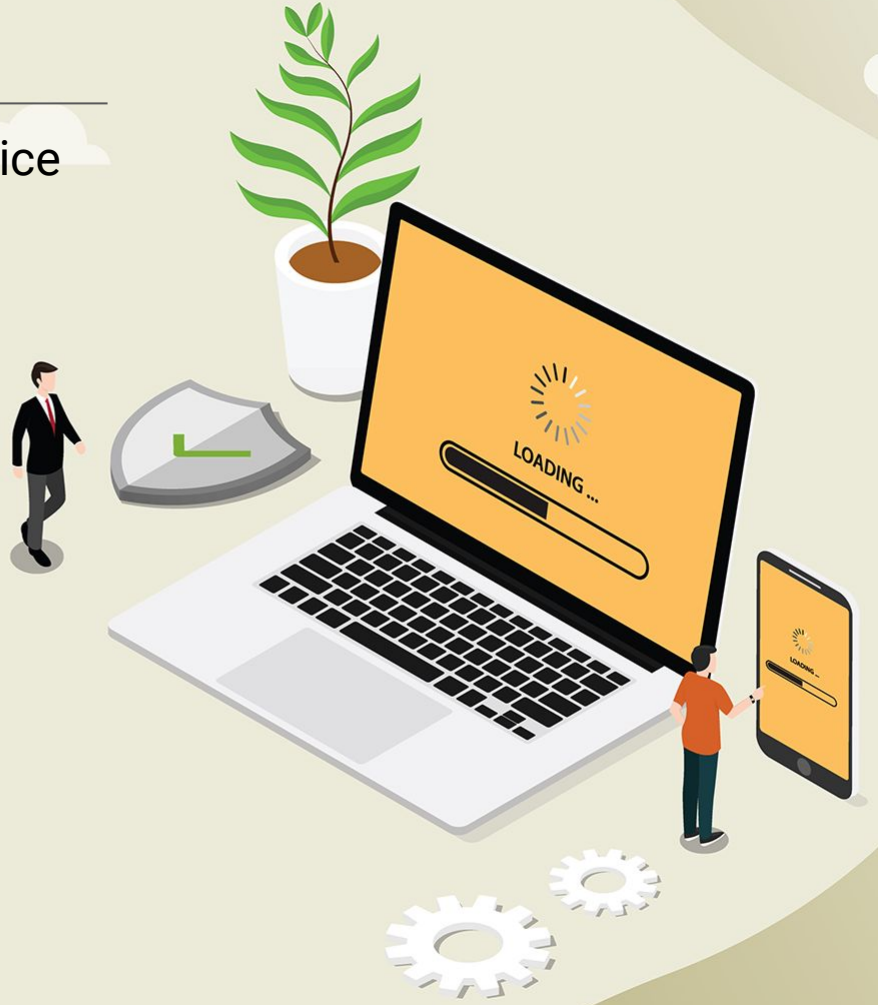


Some services are not run by real users. They are run by specific **service users** that are dedicated to running their own specific service.

Service Users

Typically, when you install a service with the package manager, a service user is automatically created and configured.

Running services under a dedicated user offers several security benefits. It makes it easier to start, stop, and manage the service, and control which files the permissions need to access.



A service user usually has a system **UID less than 1000** and cannot log in to use a shell.

Service Users

Since service users aren't humans who need to log into and interact with the machine, it's best practice to ensure that users cannot log into an interactive shell using a service username.

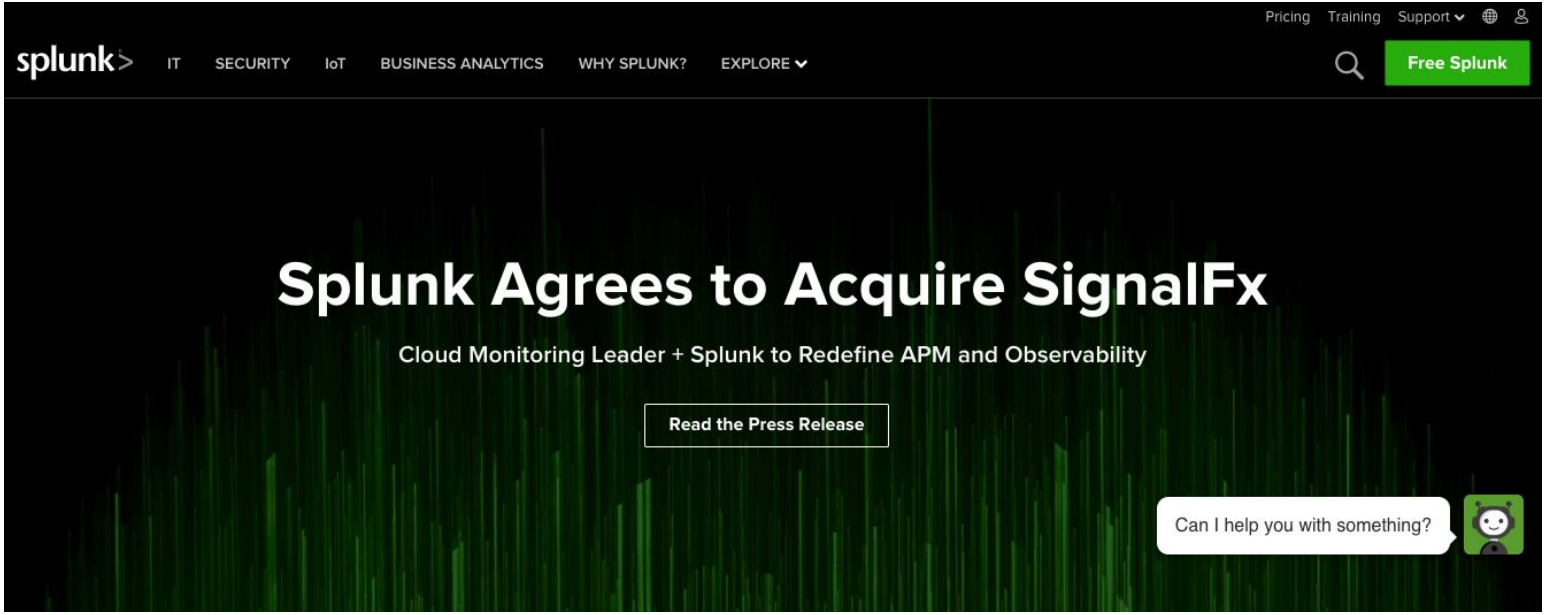
Setting Up and Adding Service Users Demo

Your senior administrator asked you to follow up on your uninstallation of unused services. You must now ensure the services' corresponding users have also been removed from the system.

Previously, you uninstalled **apache2**, but its service user, **www-data**, still exists.

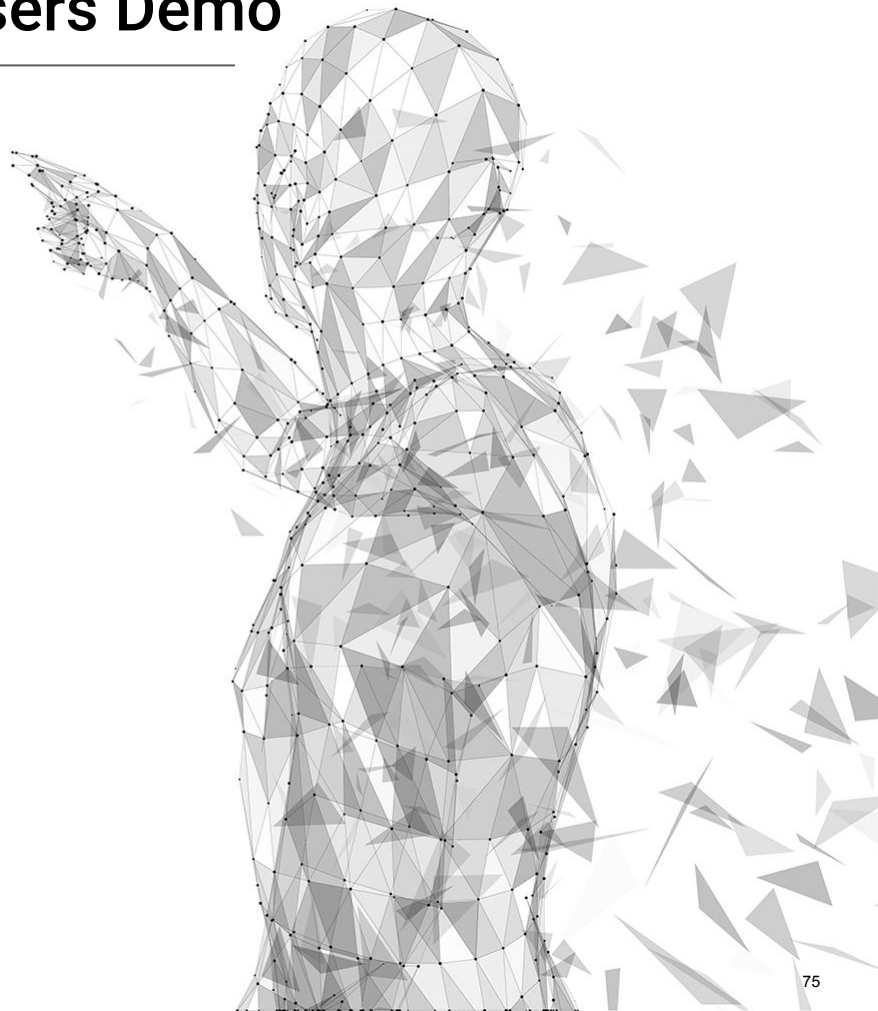
Setting Up and Adding Service Users Demo

Your senior administrator also plans to install a security service called Splunk to collect and analyze logs for suspicious activity. Like Tripwire, Splunk makes it easier for admins and security personnel to detect and stop malicious behavior.



Setting Up and Adding Service Users Demo

Your senior administrator told you that they'll handle the installation and configuration themselves, but have requested that you create a service user that they can use later.



Setting Up and Adding Service Users Demo

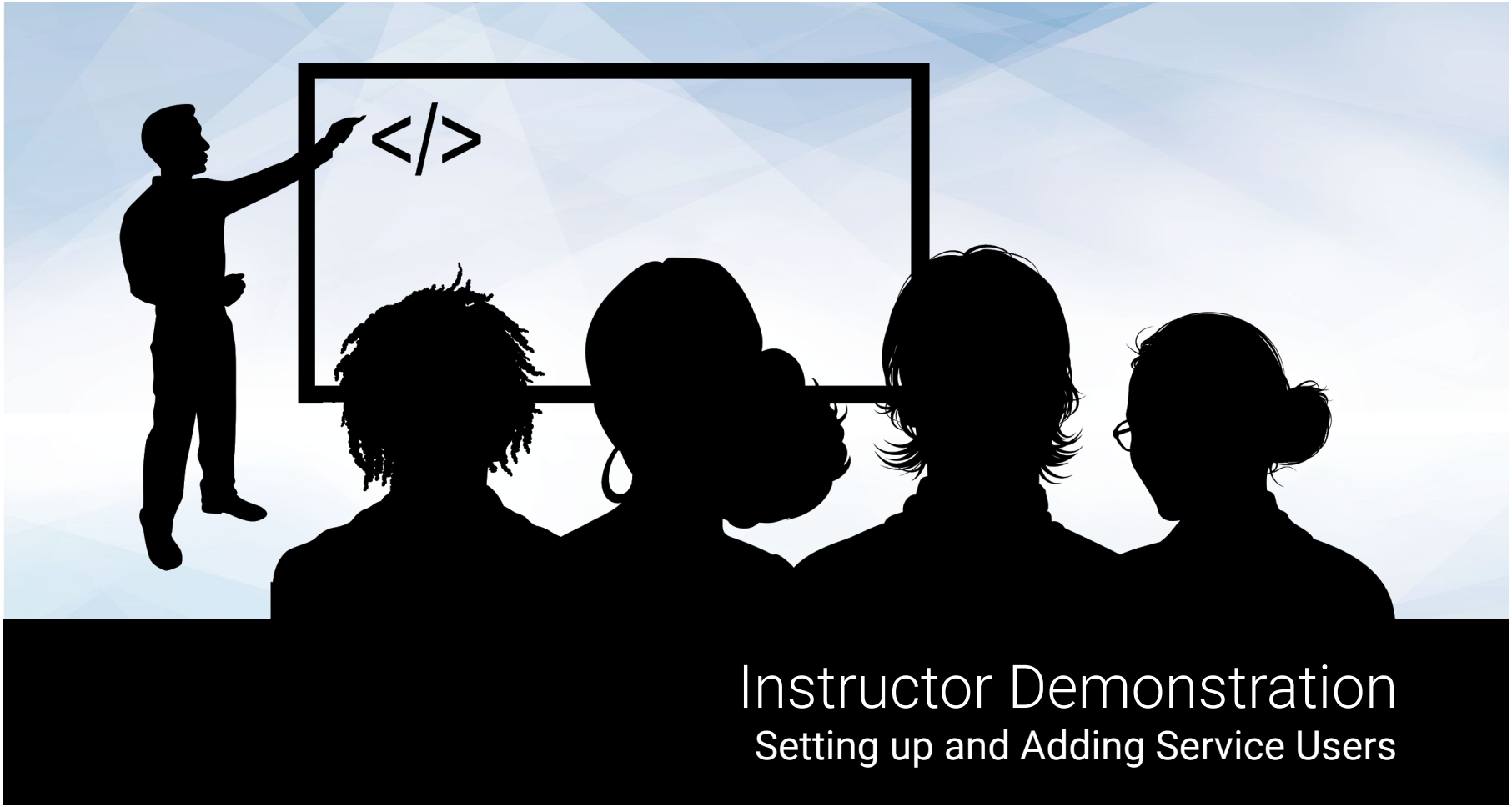
Completing this task will require the following steps:

01 **Delete**

- Deleting an old, unused service user with **`deluser/`**.

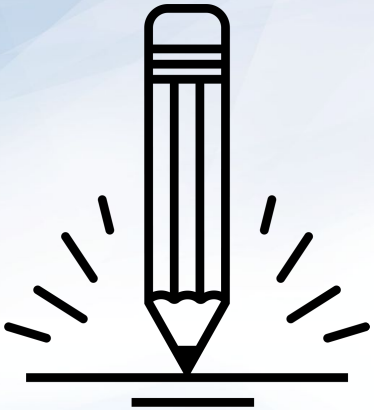
02 **Create**

- Creating and validating a new service user with **`adduser`**.



Instructor Demonstration

Setting up and Adding Service Users



Activity: Service Users

Your senior administrator would like you to remove any old service users from the system and create a new user dedicated to running Tripwire.

- Use **`adduser`** and **`deluser`** with the correct flags to clean up the system and create this new Tripwire user.
- Tripwire can only be run as **`root`** , so you must add a line to the **`sudoers`** file to allow this.

Suggested Time:
15 minutes

Time’s Up! Let’s Review.

Review: Managing Services

Completing this activity required the following steps:

Using **`deluser`** to remove four lingering users.

Using **`adduser`** with the correct flags to create a new Tripwire service user.

Editing the **`sudoers`** file to allow the Tripwire user to run the program with **`sudo`**.

Changing the **`tripwire`** user permissions to only allow the owner of Tripwire to run it.

Homework

In this week's homework, you will practice all the hardening steps we learned this week, this time on a new system.

You will also run a few new tools: **`chkrootkit`** and **`lynis`**.

Class Objectives

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

Inspect and set file permissions for sensitive files on the system.

Set file permissions with the **`SUID`** or **`GUID`** bits.

Manage and monitor services on the system, and remove unused services.

Create and assign users for services.

Questions?