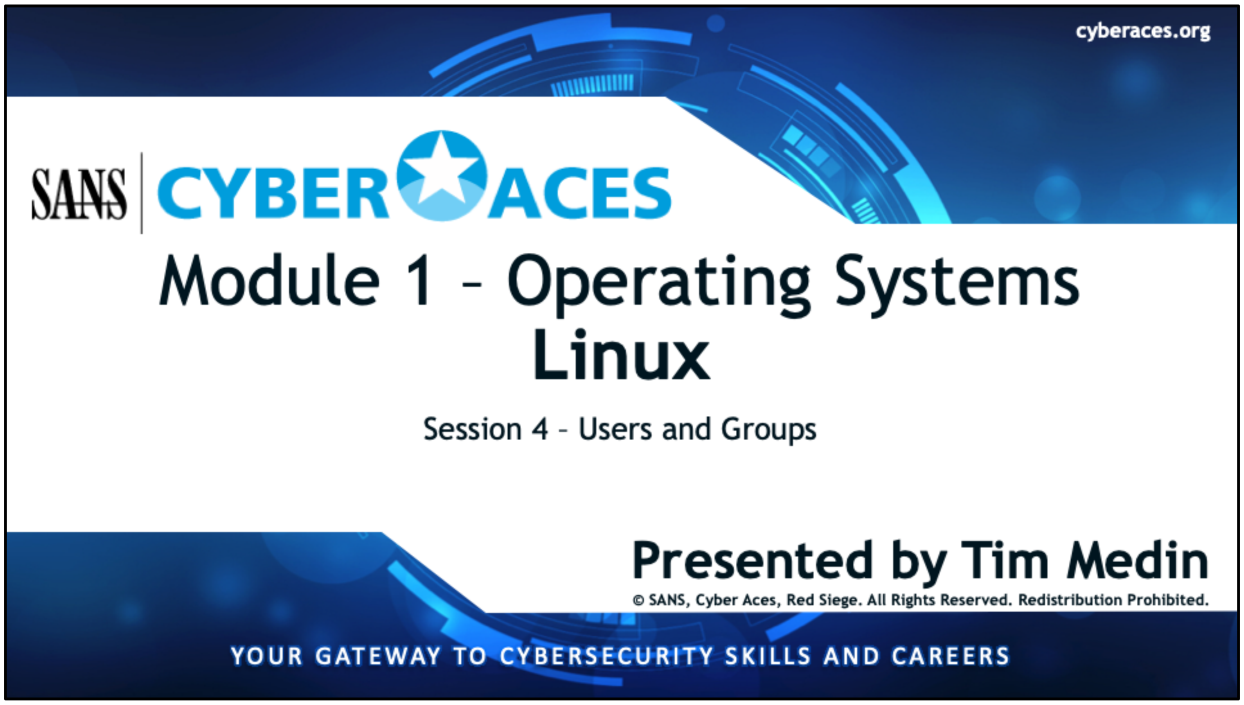

cyberaces.org

SANS | CYBER ACES

Module 1 – Operating Systems Linux

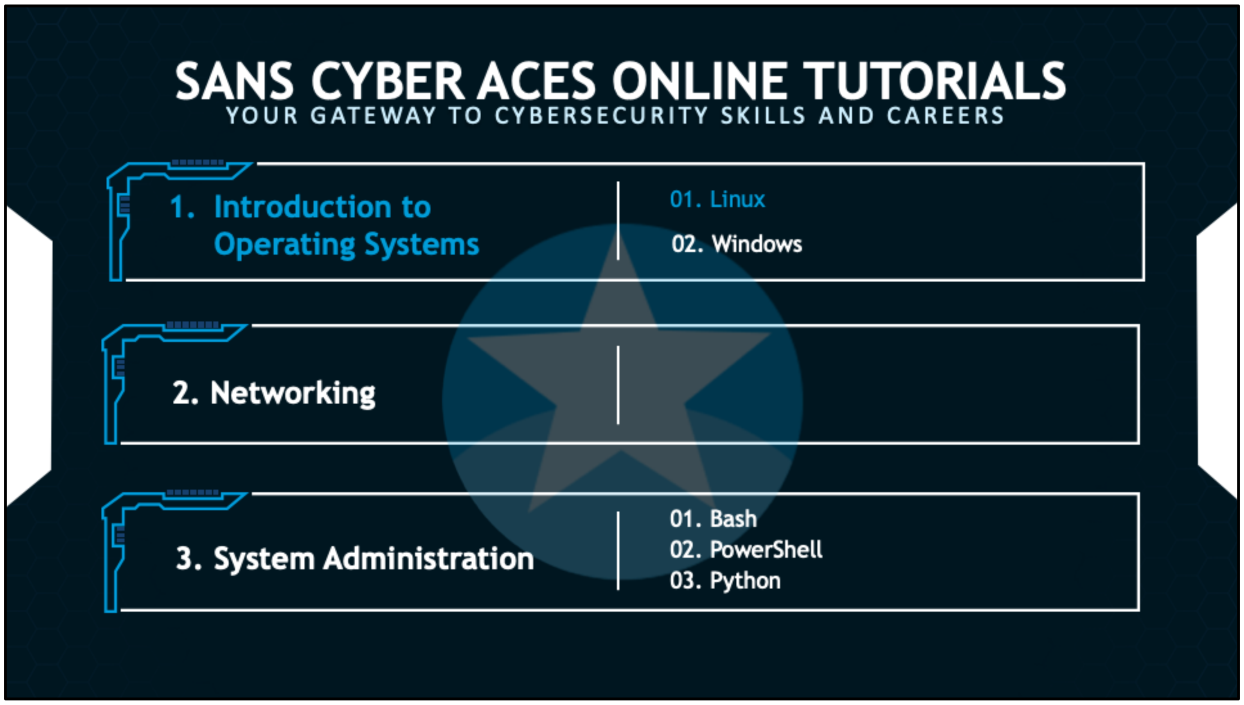
Session 4 – Users and Groups

Presented by Tim Medin

YOUR GATEWAY TO CYBERSECURITY SKILLS AND CAREERS

Welcome to Cyber Aces Online! Welcome back to Module 1, an introduction to operating systems. In this session we cover the concepts of users and groups in the Linux environment.

Content in this session has been developed by Tom Hessman, Tim Medin, Mark Baggett, Doug Burks, Michael Coppola, Russell Eubanks, Ed Skoudis, and Red Siege.



SANS CYBER ACES ONLINE TUTORIALS

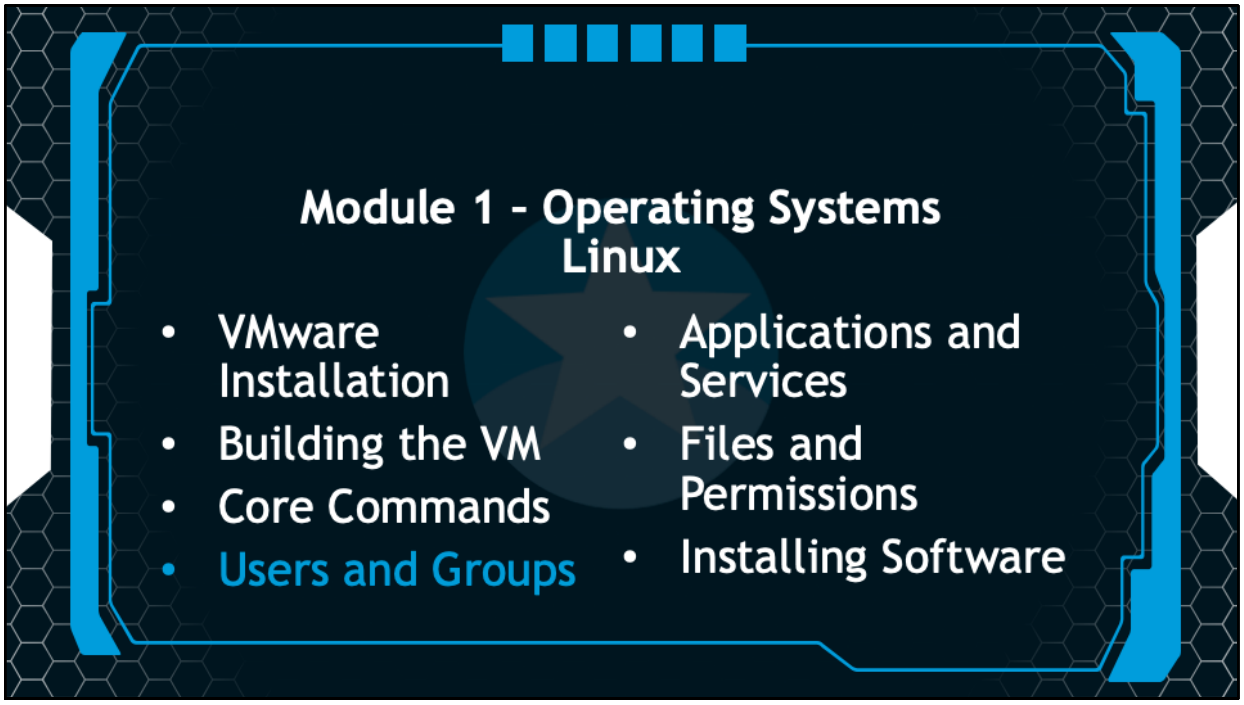
YOUR GATEWAY TO CYBERSECURITY SKILLS AND CAREERS

1. Introduction to Operating Systems
   - 01. Linux
   - 02. Windows
2. Networking
3. System Administration
   - 01. Bash
   - 02. PowerShell
   - 03. Python

This training material was originally developed to help students, teachers, and mentors prepare for the Cyber Aces Online Competition. This module focuses on the basics of what an operating systems is as well as the two predominant OS's, Windows and Linux. This session is part of Module 1, Introduction to Operating Systems. This module is split into two sections, Linux and Windows. In this session, we will continue our examination of Linux.

The three modules of Cyber Aces Online are Operating Systems, Networking, and System Administration.

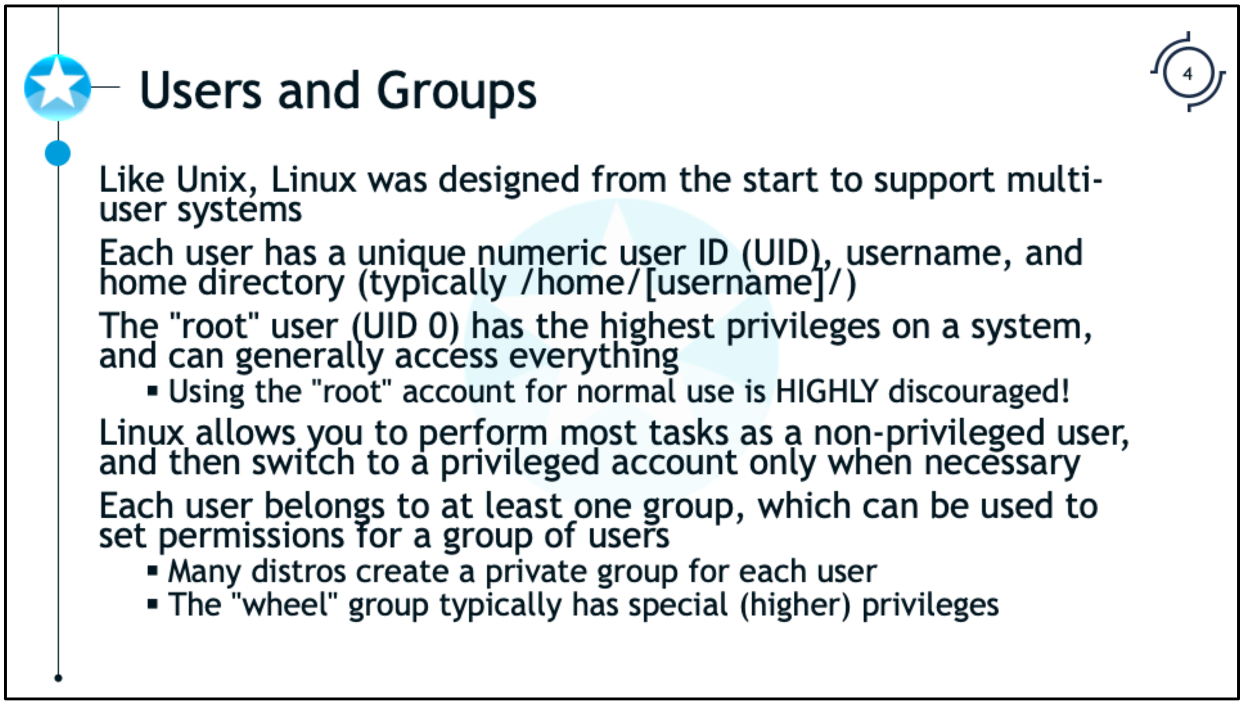
For more information about the Cyber Aces program, please visit the Cyber Aces website at https://CyberAces.org/.



Module 1 – Operating Systems Linux

- VMware Installation
- Building the VM
- Core Commands
- Users and Groups
- Applications and Services
- Files and Permissions
- Installing Software

In this session we will discuss users and groups in Linux.



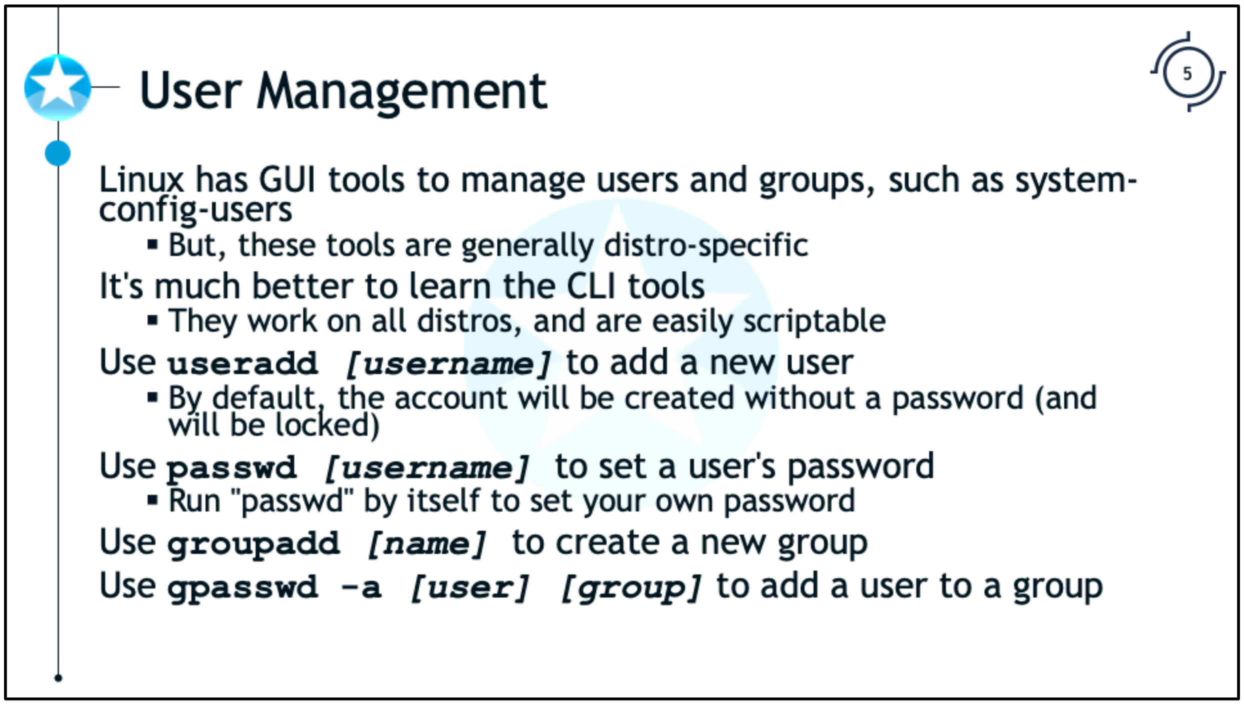
Users and Groups

- Like Unix, Linux was designed from the start to support multi-user systems
- Each user has a unique numeric user ID (UID), username, and home directory (typically /home/[username]/)
- The "root" user (UID 0) has the highest privileges on a system, and can generally access everything
  - Using the "root" account for normal use is HIGHLY discouraged!
- Linux allows you to perform most tasks as a non-privileged user, and then switch to a privileged account only when necessary
- Each user belongs to at least one group, which can be used to set permissions for a group of users
  - Many distros create a private group for each user
  - The "wheel" group typically has special (higher) privileges

Like Unix, Linux was designed from the start to support multi-user systems. Each user can independently use the system without affecting other users. Each user has a unique numeric user ID (UID), a username, and a home directory (typically /home/[username]) where their files and settings are stored. It's worth noting that it is possible for users to share a UID (most commonly to have multiple root-level users), but it is not good practice.

The "root" user (UID 0) has the highest privileges on a system, and can generally access and manipulate everything. Therefore, using the "root" account for anything other than administrative tasks that require it is strongly discouraged. For example, using root to browse the web is a bad idea because if someone exploits a vulnerability in your web browser, they now have full root privileges on your system. It is also very easy to make a mistake when using the command line, so staying out of the root account helps you avoid letting your typos have a system-wide impact. Linux and Linux-based software is generally designed such that you can perform most tasks without needing root access, and it's easy to obtain root privileges when you need them.

Linux also supports groups, which allow permissions or privileges to be assigned to sets of users. Many Linux distributions create a "private" group for each user, which is simply a group with the same name as the user's username, where the user is the only member. Users in the "wheel" group typically have special (i.e., higher) privileges on the system, such as the ability to become root, or the ability to bypass certain security restrictions.



User Management

5

- Linux has GUI tools to manage users and groups, such as system-config-users
  - But, these tools are generally distro-specific
- It's much better to learn the CLI tools
  - They work on all distros, and are easily scriptable
- Use **`useradd`** ***`[username]`*** to add a new user
  - By default, the account will be created without a password (and will be locked)
- Use **`passwd`** ***`[username]`*** to set a user's password
  - Run "passwd" by itself to set your own password
- Use **`groupadd`** ***`[name]`*** to create a new group
- Use **`gpasswd -a`** ***`[user] [group]`*** to add a user to a group

Linux provides GUI tools to manage user accounts, but these tools are generally unique to each individual distribution. For example, Red Hat comes with system-config-users, Suse comes with Yast, and Debian Linux and other distros come with the users-admin GUI tool set. Instead of learning how to use different tools for different distributions, you should instead learn how to use the command line to perform these same actions on any distribution. By utilizing the command line, you are also able to easily automate user management tasks through shell scripting.

To add a new user to a system, use the "useradd" command, passing the username as the parameter. For example, to create a new user account called "testuser", run:

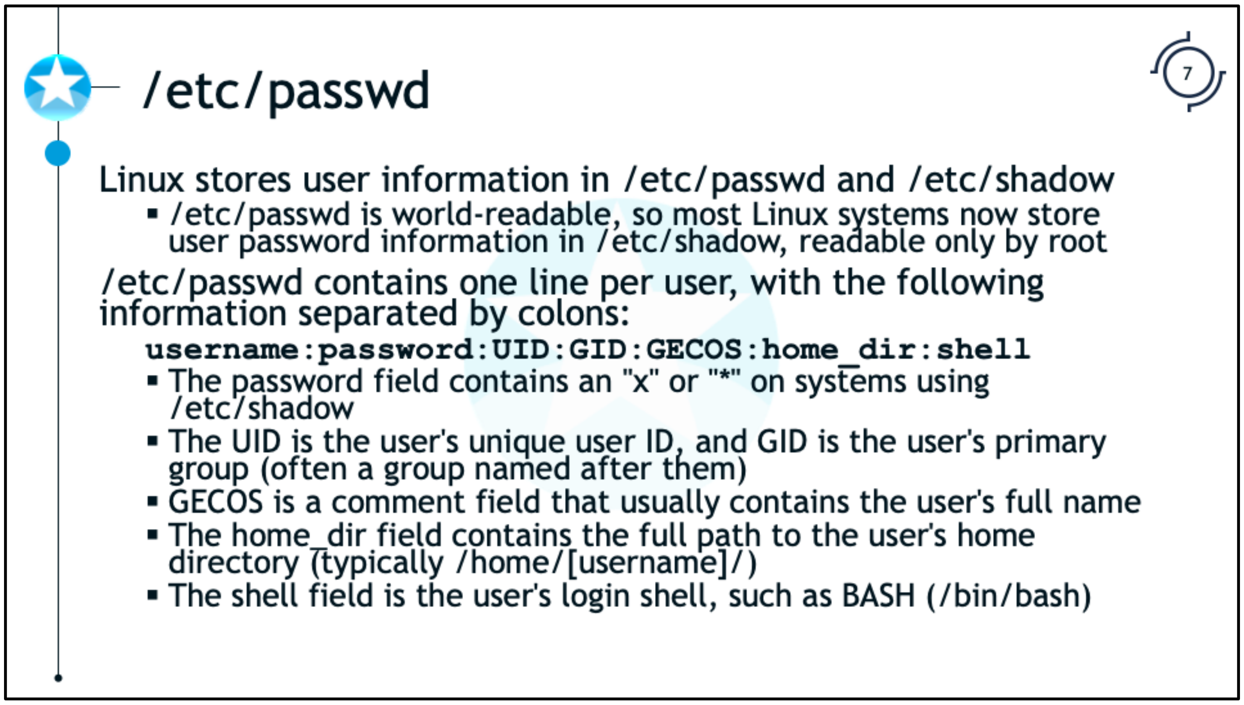
```
# useradd testuser
```

By default, new user accounts are created without a password (null, not a blank password) and are locked to prevent someone logging in. To set testuser's password, use the "passwd" command:

```
# passwd testuser
Changing password for user testuser.
New password: [enter password here]
Retype new password: [enter password here]
passwd: all authentication tokens updated successfully.
```

To create a new group, use the "groupadd" command:

```
# groupadd testgroup
```



/etc/passwd

Linux stores user information in /etc/passwd and /etc/shadow

- /etc/passwd is world-readable, so most Linux systems now store user password information in /etc/shadow, readable only by root

/etc/passwd contains one line per user, with the following information separated by colons:

```
username:password:UID:GID:GECOS:home_dir:shell
```

- The password field contains an "x" or "*" on systems using /etc/shadow
- The UID is the user's unique user ID, and GID is the user's primary group (often a group named after them)
- GECOS is a comment field that usually contains the user's full name
- The home_dir field contains the full path to the user's home directory (typically /home/[username]/)
- The shell field is the user's login shell, such as BASH (/bin/bash)

Linux passwords are managed through a joint effort of the /etc/passwd and /etc/shadow files, which provide the operating system a means of storing account information and credentials. Shadow passwords are critical to system security and have thus been implemented in every modern mainstream Linux distribution. It is important for system administrators and attackers alike to understand their purpose and significance. The /etc/passwd file has to be readable by all users on the system because various utilities need to look up user information, so shadow passwords were implemented to keep the passwords private in the protected /etc/shadow file (that only root can read).

The file /etc/passwd contains one line per user, with 7 fields of information separated by colons (:). The fields are:

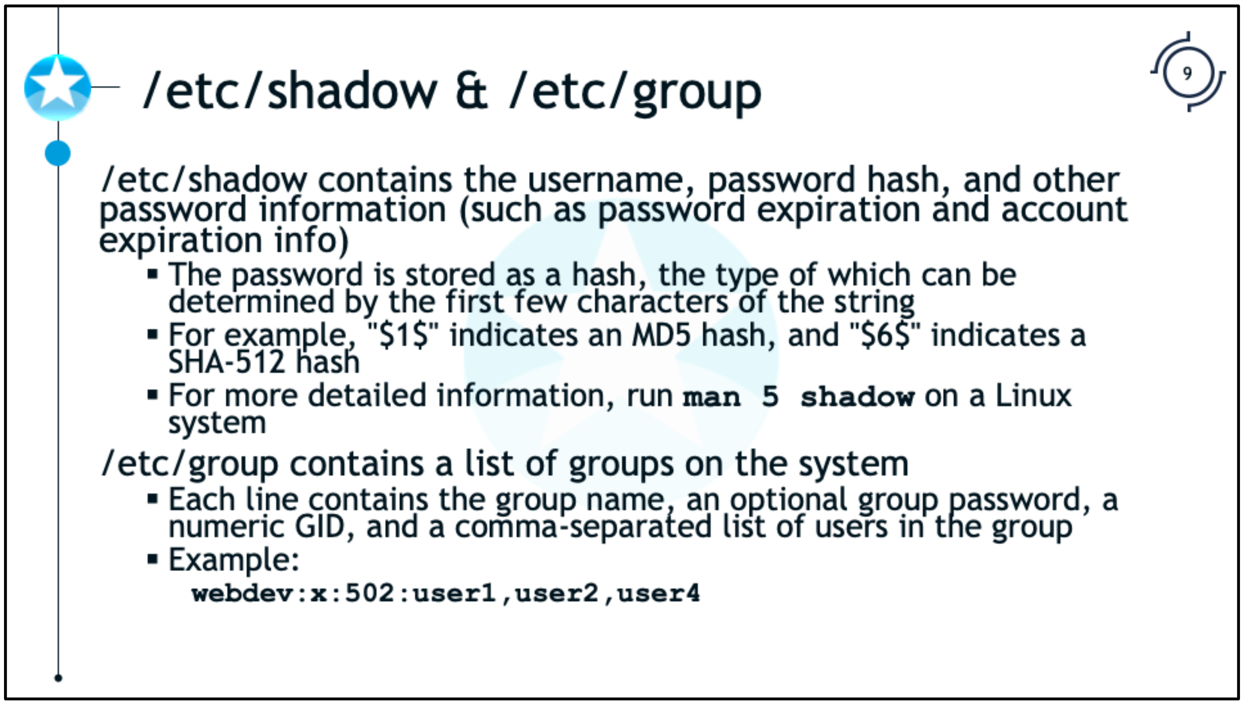
username: The account's unique username. It can only contain lower case letters, numbers, underscores, and hyphens. It can start with either a lowercase letter or an underscore, and can optionally end with a dollar sign ($).

password: Contains either an "x" if the system is using shadow passwords, or the hash of the user's password if it isn't. If it starts with an exclamation point, the account is disabled.

UID: The user's unique user ID, such as "501". The root account is UID 0, and any other account with UID 0 automatically has root privileges.

GID: The GID of the user's primary group. This is the primary group the user will be a member of for the purposes of file permissions, particularly on newly created files.

GECOS: This is used as a comment field, and usually contains the user's full name. It can also contain other information, such as phone numbers. This field can be edited using the "chfn" command.



/etc/shadow & /etc/group

/etc/shadow contains the username, password hash, and other password information (such as password expiration and account expiration info)

- The password is stored as a hash, the type of which can be determined by the first few characters of the string
- For example, "1" indicates an MD5 hash, and "6" indicates a SHA-512 hash
- For more detailed information, run **`man 5 shadow`** on a Linux system

/etc/group contains a list of groups on the system

- Each line contains the group name, an optional group password, a numeric GID, and a comma-separated list of users in the group
- Example:

```
webdev:x:502:user1,user2,user4
```

On systems with shadow passwords enabled (the vast majority of Linux systems today), the /etc/shadow file contains additional information about each user account, most importantly the password hash. Here is a full breakdown of the colon-separated fields:

Username: The username of the account.

Password: The password hash, which can be in a number of formats supported by crypt(). The hash type is determined by the number found after the first dollar sign. For example, "1" indicates salted MD5, while 6" indicates salted SHA-512.

Date of last password change: The date the user last changed his or her password, expressed as the number of days since January 1, 1970.

Minimum password age: The minimum number of days the user has to wait before changing his or her password after a password change.

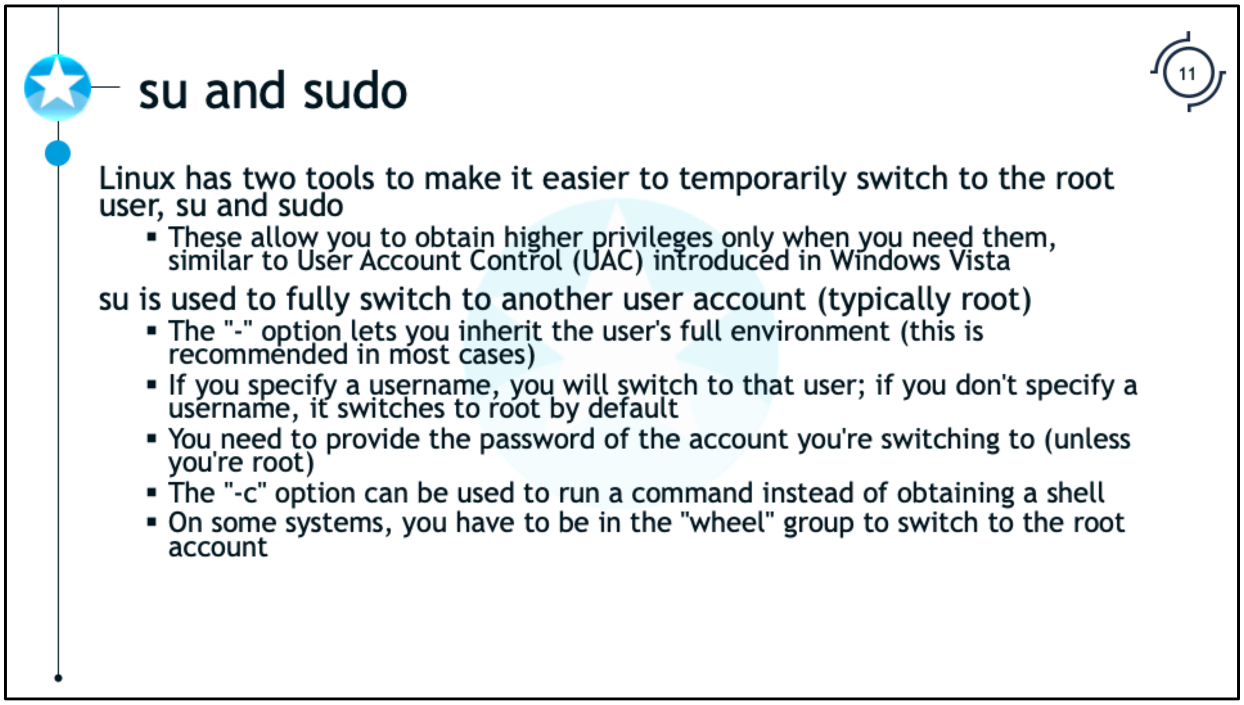
Maximum password age: The maximum number of days the user can keep a password before being forced to change it.

Password warning period: The number of days before a password expires that the user should be warned.

Password inactivity period: The number of days after a password has expired that a password should still be accepted as valid.

Account expiration date: The date that the account will expire and not allow ANY type of login, expressed as the number of days since January 1, 1970.

Reserved: This field is currently unused and reserved for future use.



su and sudo

Linux has two tools to make it easier to temporarily switch to the root user, su and sudo

- These allow you to obtain higher privileges only when you need them, similar to User Account Control (UAC) introduced in Windows Vista

su is used to fully switch to another user account (typically root)

- The "-" option lets you inherit the user's full environment (this is recommended in most cases)
- If you specify a username, you will switch to that user; if you don't specify a username, it switches to root by default
- You need to provide the password of the account you're switching to (unless you're root)
- The "-c" option can be used to run a command instead of obtaining a shell
- On some systems, you have to be in the "wheel" group to switch to the root account

Linux also offers tools that allow administrative tasks to be performed from user accounts without granting the account unnecessary permissions. Similar to User Account Control (UAC) introduced in Windows Vista, these two tools are named "su" and "sudo" and have been used by system administrators for many years.

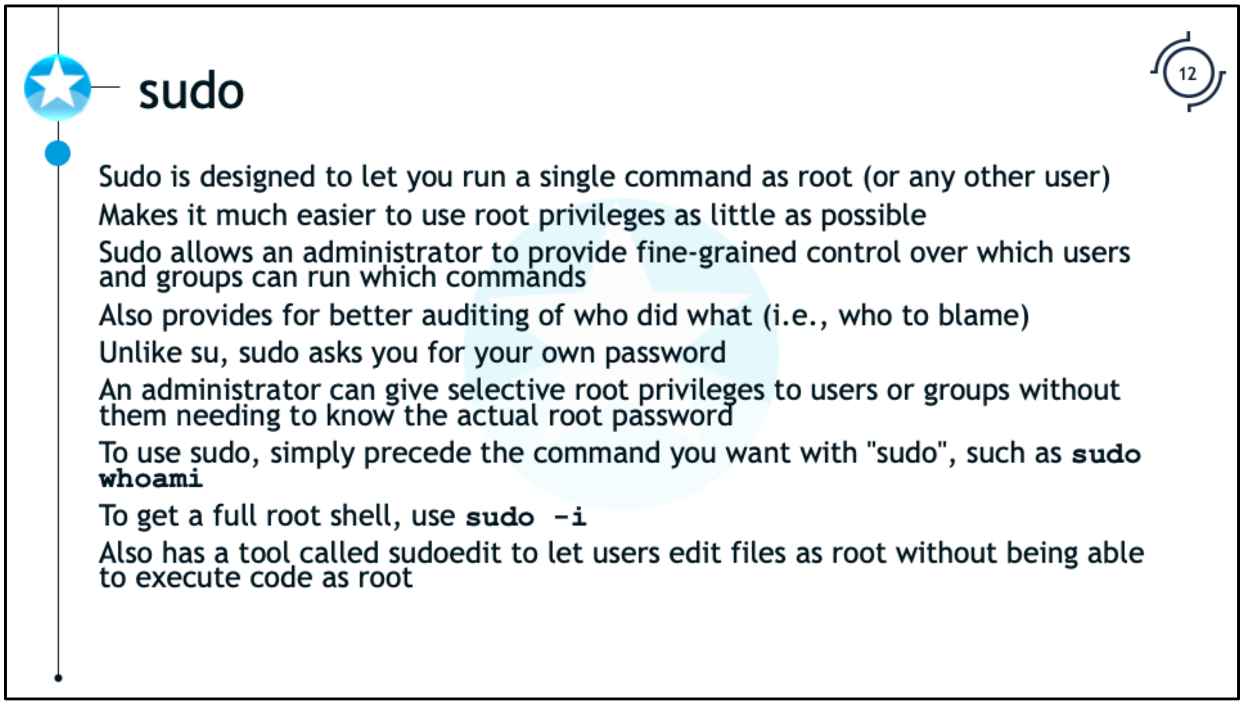
The "su" (substitute user) command is used to fully switch to another user account, typically root. When using su, you are prompted for the password of the account you are switching to (unless you are root, in which case you can switch to any account with a valid login shell). The "-" option can be used to run the new shell as a login shell, inheriting the full environment (this is recommended in most cases). To switch to root, you can just run **su** or **su -**.

```
jdoe@localhost$ su -
Password: <enter root password>
root@localhost$
```

To switch to a specific user, just specify the username.

```
jdoe@localhost$ su - testuser
Password: <enter testuser's password>
testuser@localhost$ exit
jdoe@localhost$
```

Some Linux systems require you to be in the "wheel" group in order to switch to the root account using su, even if you know the correct password.



sudo

- Sudo is designed to let you run a single command as root (or any other user)
- Makes it much easier to use root privileges as little as possible
- Sudo allows an administrator to provide fine-grained control over which users and groups can run which commands
- Also provides for better auditing of who did what (i.e., who to blame)
- Unlike su, sudo asks you for your own password
- An administrator can give selective root privileges to users or groups without them needing to know the actual root password
- To use sudo, simply precede the command you want with "sudo", such as **`sudo whoami`**
- To get a full root shell, use **`sudo -i`**
- Also has a tool called sudoedit to let users edit files as root without being able to execute code as root

The "sudo" command is designed to let a user run a single command as root (or any other user), rather than having to completely switch to a full shell as the other user. This makes it much easier to avoid using the root account except when absolutely necessary. Sudo allows an administrator to specify specific users and groups that have permission to run commands as root, and also allows the administrator to specify specific commands that each user or group can run.

Unlike su, sudo asks you for *your own password*. This allows administrators to grant root privileges to users without them needing to know the actual root password, which also allows for privileges to be easily audited and revoked. To use sudo, simply put the command "sudo" before the command that you want to run as root. For example, to run the "whoami" command as root, run:

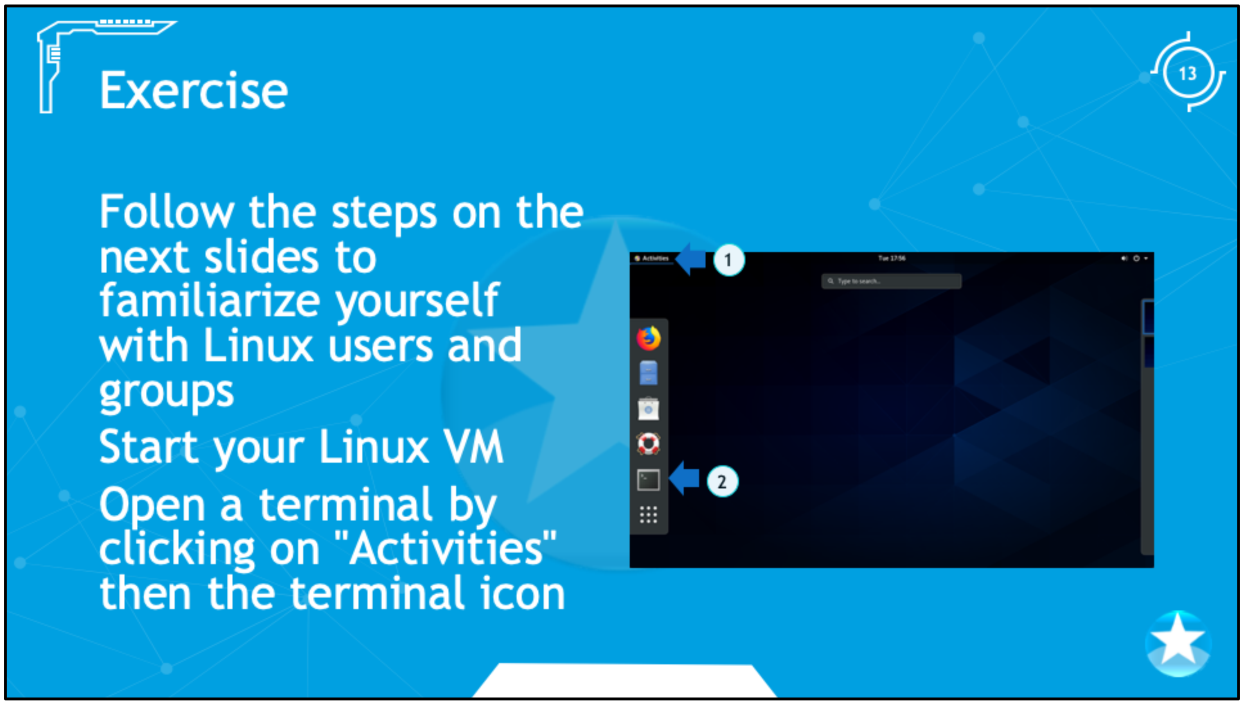
```
jdoe@localhost$ sudo whoami
[sudo] password for jdoe: <enter jdoe's password>
root
```

If you still want to get a full root shell isntead of just running a single command, you can use the -i option.

```
jdoe@localhost$ sudo -i
[sudo] password for jdoe: <enter jdoe's password>
root@localhost$
```

Sudo also has a tool called "sudoedit" that allows you to edit files with root privileges without having full code execution.

```
jdoe@localhost$ sudoedit /etc/shadow
```



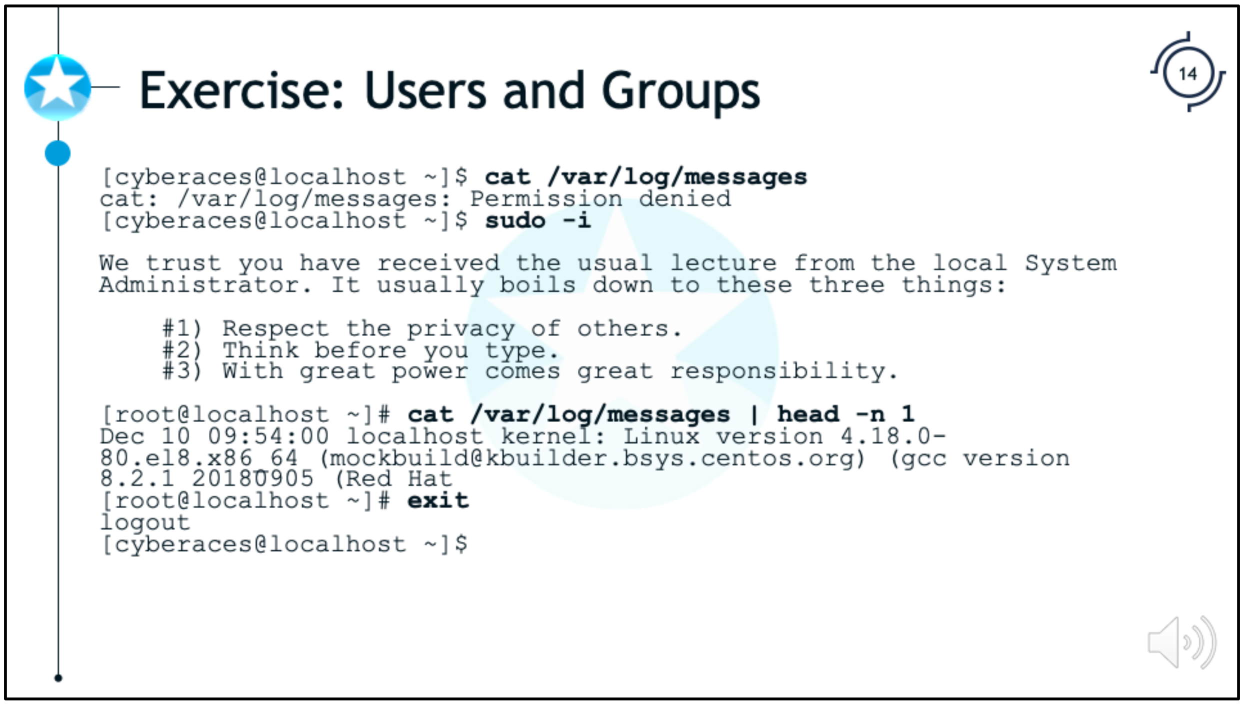
Exercise

Follow the steps on the next slides to familiarize yourself with Linux users and groups

Start your Linux VM

Open a terminal by clicking on "Activities" then the terminal icon

At this point we will start using our Linux environment to familiarize ourselves with the command line and Linux's users and groups. To open the command line interface, or terminal, click on Applications menu at the top, then System Tools, and finally Terminal. At this point you should see a new window open in the VM titled "cyberaces@localhost:~".



Exercise: Users and Groups

```
[cyberaces@localhost ~]$ cat /var/log/messages
cat: /var/log/messages: Permission denied
[cyberaces@localhost ~]$ sudo -i

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[root@localhost ~]# cat /var/log/messages | head -n 1
Dec 10 09:54:00 localhost kernel: Linux version 4.18.0-
80.el8.x86_64 (mockbuild@kbuilder.bsys.centos.org) (gcc version
8.2.1 20180905 (Red Hat
[root@localhost ~]# exit
logout
[cyberaces@localhost ~]$
```

Let's practice using our newfound knowledge of users, su, and sudo and walkthrough these steps:

1. Notice that your prompt says "cyberaces@localhost". This means you're logged into an account called "cyberaces". The user account "cyberaces" has limited privileges. Let's demonstrate that by trying to read a protected file with the following command:

   ```
   $ cat /var/log/messages
   ```

   You should receive a "Permission Denied" error, because the user "cyberaces" doesn't have permissions to read the /var/log/messages file.

2. Let's get root level access using the "sudo -i" command:

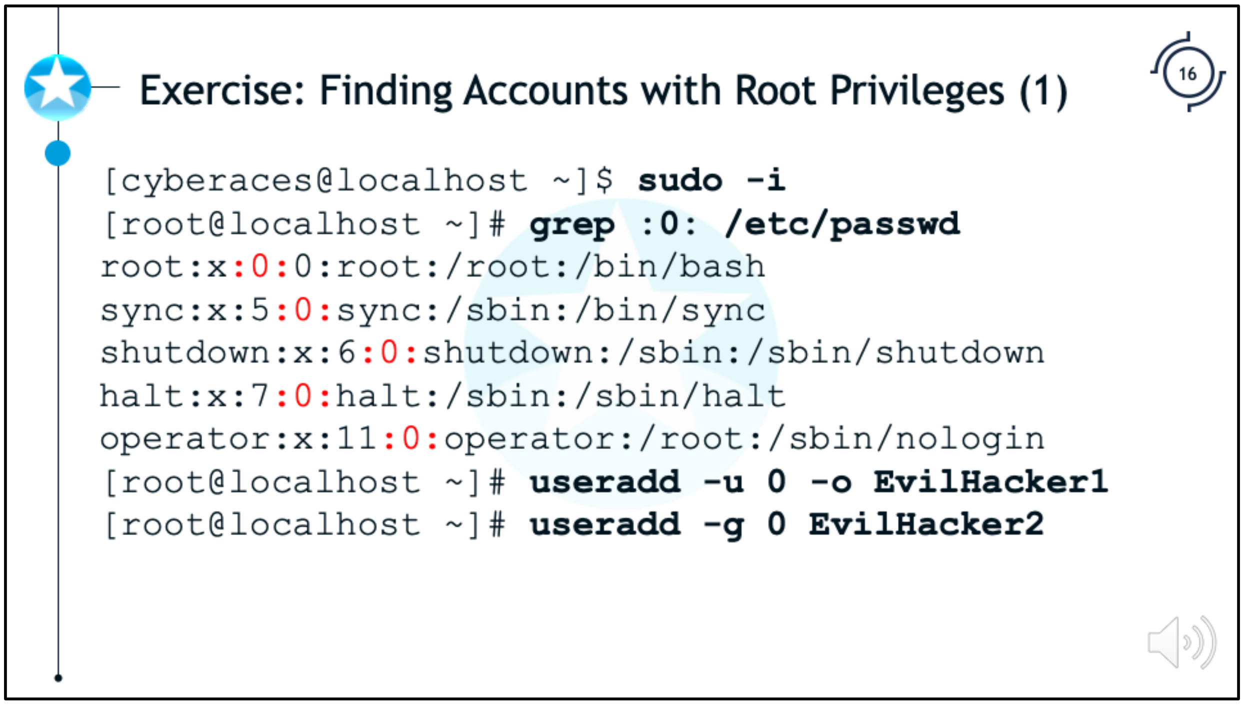
   ```
   $ sudo -i
   ```

   Normally, you would be prompted for your password here, but remember we created this account without a password. Alternatively, we could have logged in to the root account (using the password for the "root" account) with the "su" command. Notice that your prompt now says "root@localhost" and the dollar sign ($) has changed to a pound sign (#). This means that you're now logged into the "root" account and you have full privileges in this VM.

3. Now, let's try reading the protected file again:

   ```
   # cat /var/log/messages | head -n 1
   ```

   The command should now succeed and you should see the first 2 lines (because of head -2) of the /var/log/messages file. This is because the root user can read ANY file on the system. Your output will be different that that shown above.



Exercise: Finding Accounts with Root Privileges (1)

```
[cyberaces@localhost ~]$ sudo -i
[root@localhost ~]# grep :0: /etc/passwd
root:x:0:0:root:/root:/bin/bash
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
operator:x:11:0:operator:/root:/sbin/nologin
[root@localhost ~]# useradd -u 0 -o EvilHacker1
[root@localhost ~]# useradd -g 0 EvilHacker2
```

As you will recall, users with a UID of 0 have full access to the system. This access is often referred to as root level access. Let's walkthrough and exercise to create then find accounts with UID 0 or GID 0.

1. In your CentOS VM, open a terminal and type "su -" to become root.

   ```
   $ sudo -i
   ```

2. Type the command and note the output:

   ```
   # grep :0: /etc/passwd
   ```

   There should be 5 accounts that have a 0 in the UID or GID fields. The "grep" command will filter for lines matching ":0:" in the /etc/passwd file.

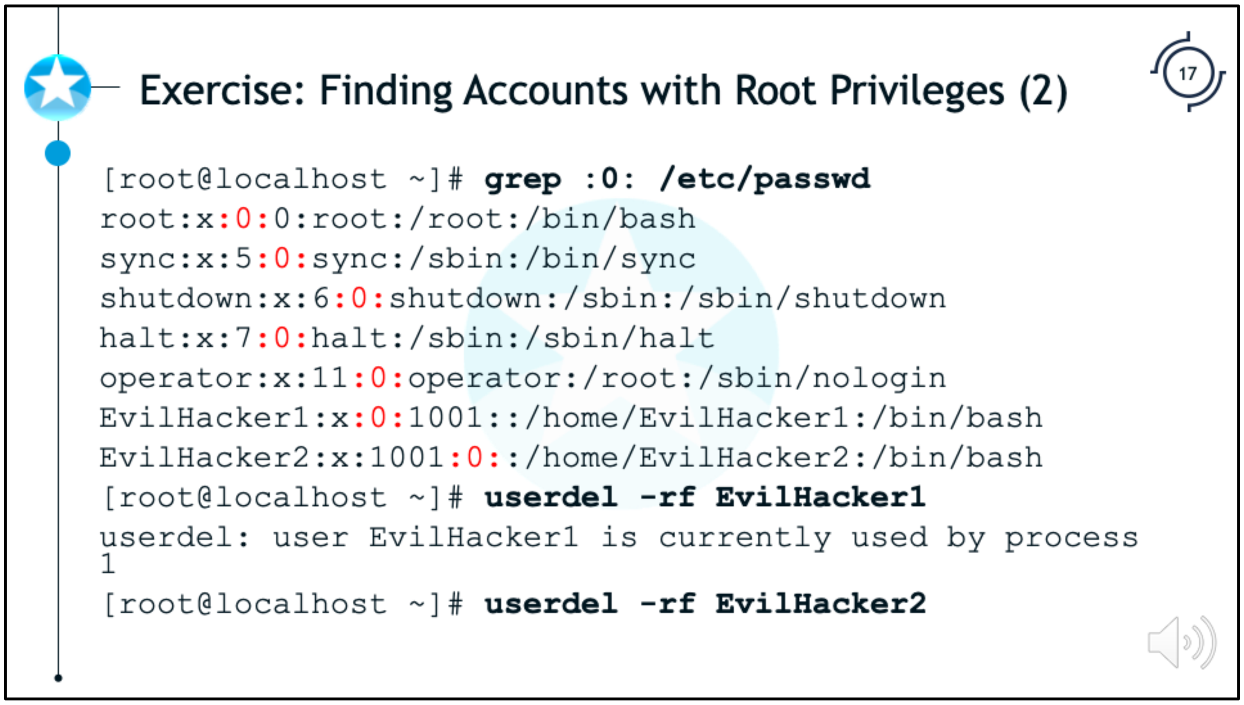
3. Add a new user with UID 0 with the following command:

   ```
   # useradd -u 0 -o EvilHacker1
   ```

   The -u option specifies the UID value, otherwise the OS will pick the next available number. The -o option will allow us to create an account with a non-unique UID (the existing root account has UID 0).

4. Add a new user with GID 0 with the following command:

   ```
   # useradd -g 0 EvilHacker2
   ```



Exercise: Finding Accounts with Root Privileges (2)

```
[root@localhost ~]# grep :0: /etc/passwd
root:x:0:0:root:/root:/bin/bash
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
operator:x:11:0:operator:/root:/sbin/nologin
EvilHacker1:x:0:1001::/home/EvilHacker1:/bin/bash
EvilHacker2:x:1001:0::/home/EvilHacker2:/bin/bash
[root@localhost ~]# userdel -rf EvilHacker1
userdel: user EvilHacker1 is currently used by process
1
[root@localhost ~]# userdel -rf EvilHacker2
```

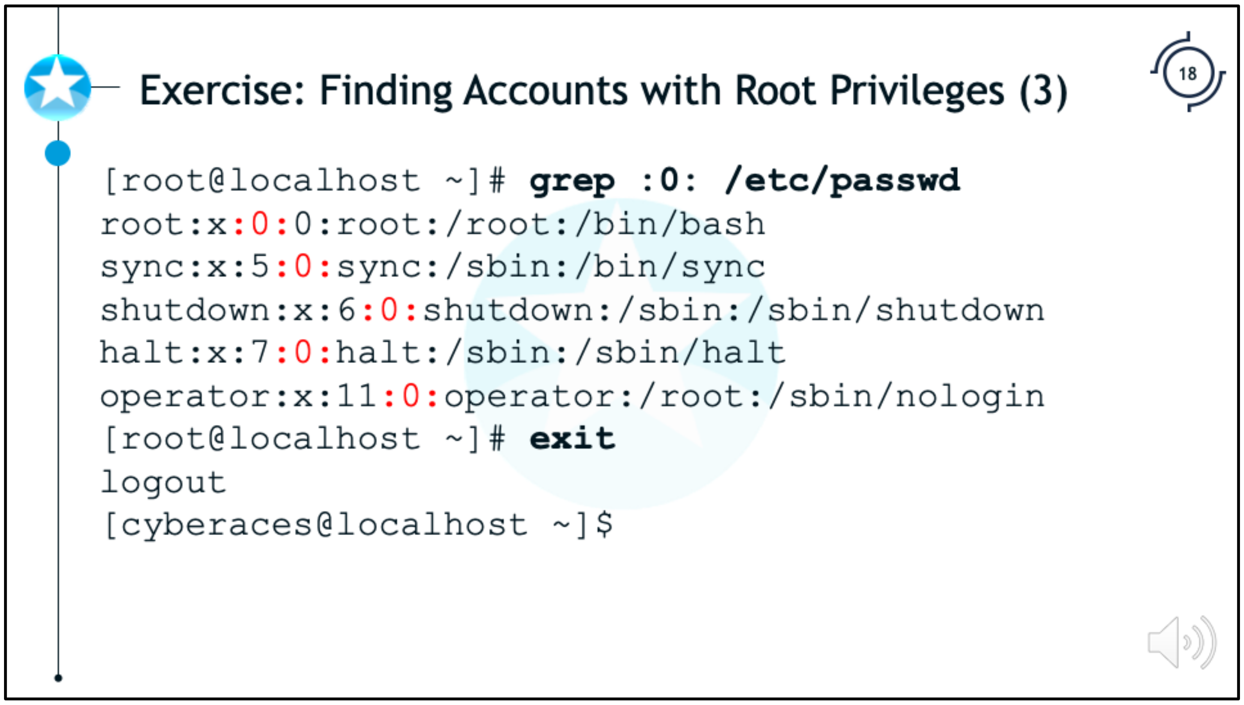
5. Run the grep command again. You could type the command all over again, or you could simply press the Up arrow three times to retrieve that command from your command history and then press Enter. There should now be 7 accounts shown. Notice that the EvilHacker1 account has a 0 in the UID field and EvilHacker2 has a 0 in the GID field.
6. Delete the EvilHacker account with the following command:

   ```
   # userdel -rf EvilHacker
   ```

   You will see a warning that "user EvilHacker1 is currently logged in" because there are processes running as the root user (also UID 0). You can ignore this error as EvilHacker1 has been deleted. If you want to confirm you could run the previous grep command again.
7. Delete the EvilHacker2 account by pressing the Up arrow, replacing the 1 with a 2, and pressing Enter:

   ```
   # userdel -rf EvilHacker2
   ```

   At this point both of the accounts have been deleted.



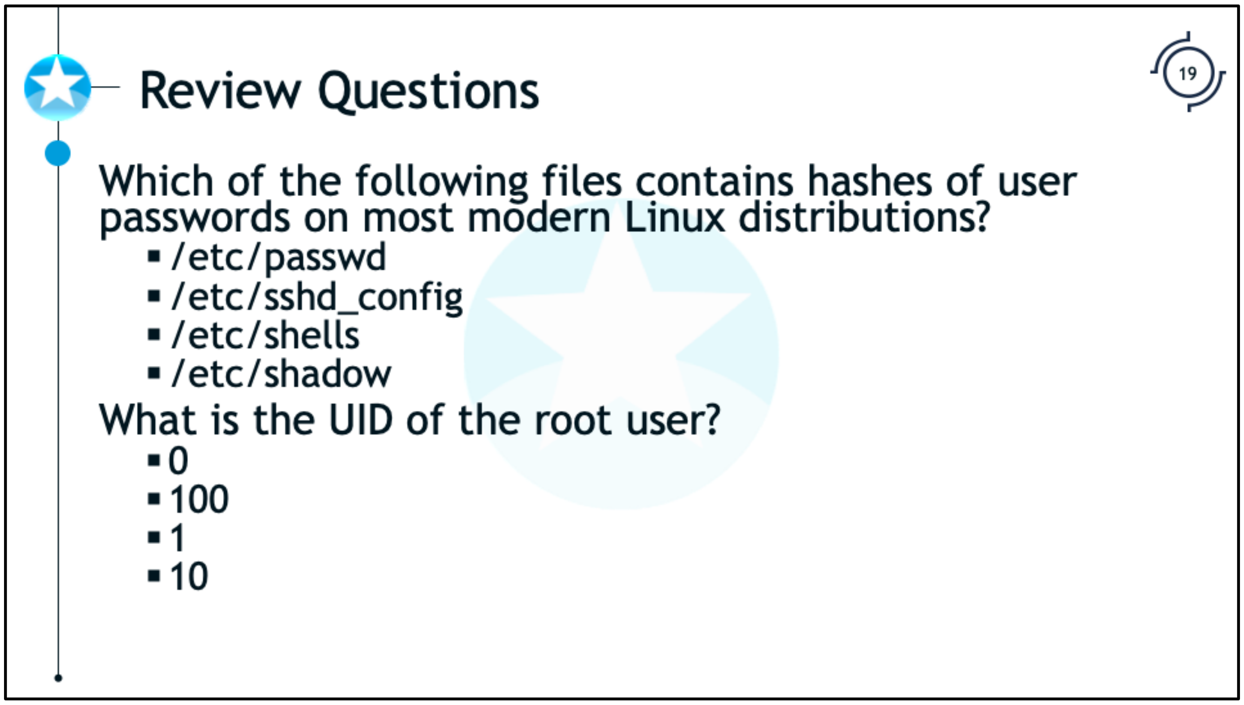
Exercise: Finding Accounts with Root Privileges (3)

```
[root@localhost ~]# grep :0: /etc/passwd
root:x:0:0:root:/root:/bin/bash
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
operator:x:11:0:operator:/root:/sbin/nologin
[root@localhost ~]# exit
logout
[cyberaces@localhost ~]$
```

8. Press the Up arrow three times to retrieve the grep command again and press Enter. You should now be back to 5 accounts with a 0 in the UID or GID fields. The two extra accounts, EvilHacker1 and EvilHacker2, have been deleted.

9. Type "exit" to leave root level access.

The grep command used here is useful to find accounts that have a UID or GID of 0. With the right access, an attacker could create a backdoor account that would give him full access to the system. This is a good way to see if any such accounts exist on the system. This command is referenced in the SANS Linux Intrusion Discovery Cheat Sheet (free login required), located at: https://redsiege.com/ca/lin-cheat



Review Questions

Which of the following files contains hashes of user passwords on most modern Linux distributions?
- /etc/passwd
- /etc/sshd_config
- /etc/shells
- /etc/shadow

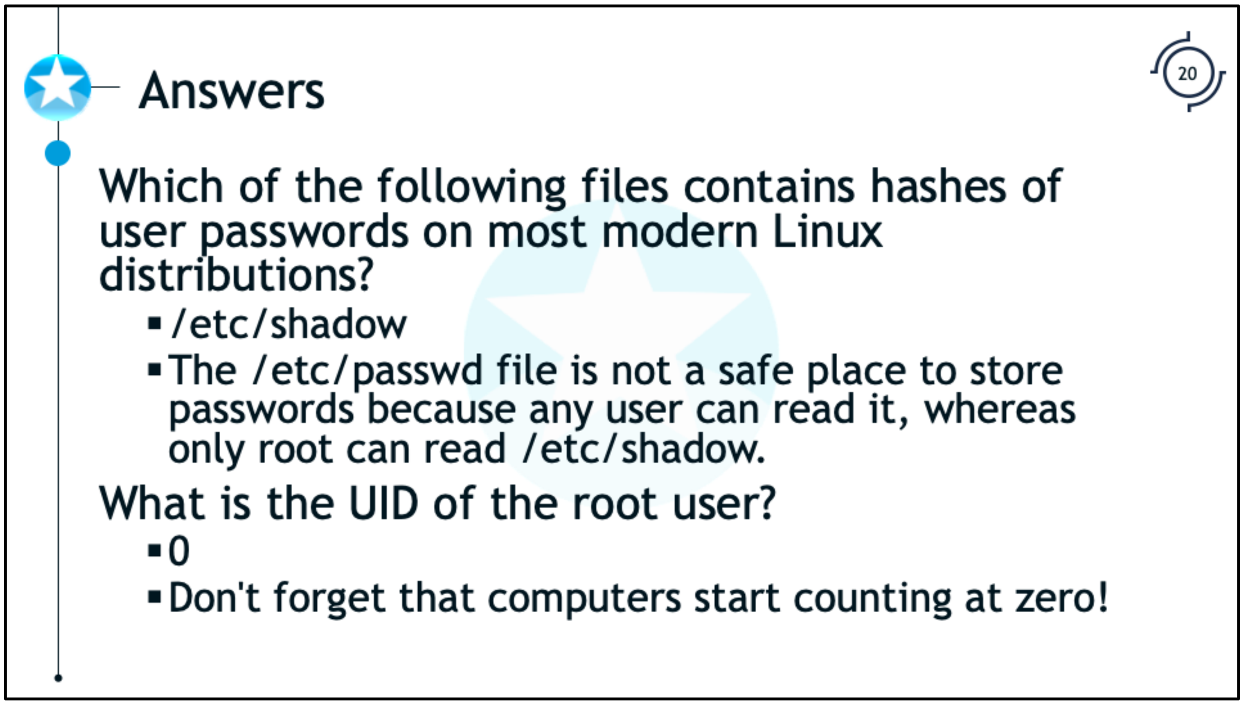
What is the UID of the root user?
- 0
- 100
- 1
- 10

Which of the following files contains hashes of user passwords on most modern Linux distributions?

/etc/passwd
/etc/sshd_config
/etc/shells
/etc/shadow

What is the UID of the root user?

0
100
1
10



Answers

Which of the following files contains hashes of user passwords on most modern Linux distributions?

- /etc/shadow
- The /etc/passwd file is not a safe place to store passwords because any user can read it, whereas only root can read /etc/shadow.

What is the UID of the root user?

- 0
- Don't forget that computers start counting at zero!

Which of the following files contains hashes of user passwords on most modern Linux distributions?

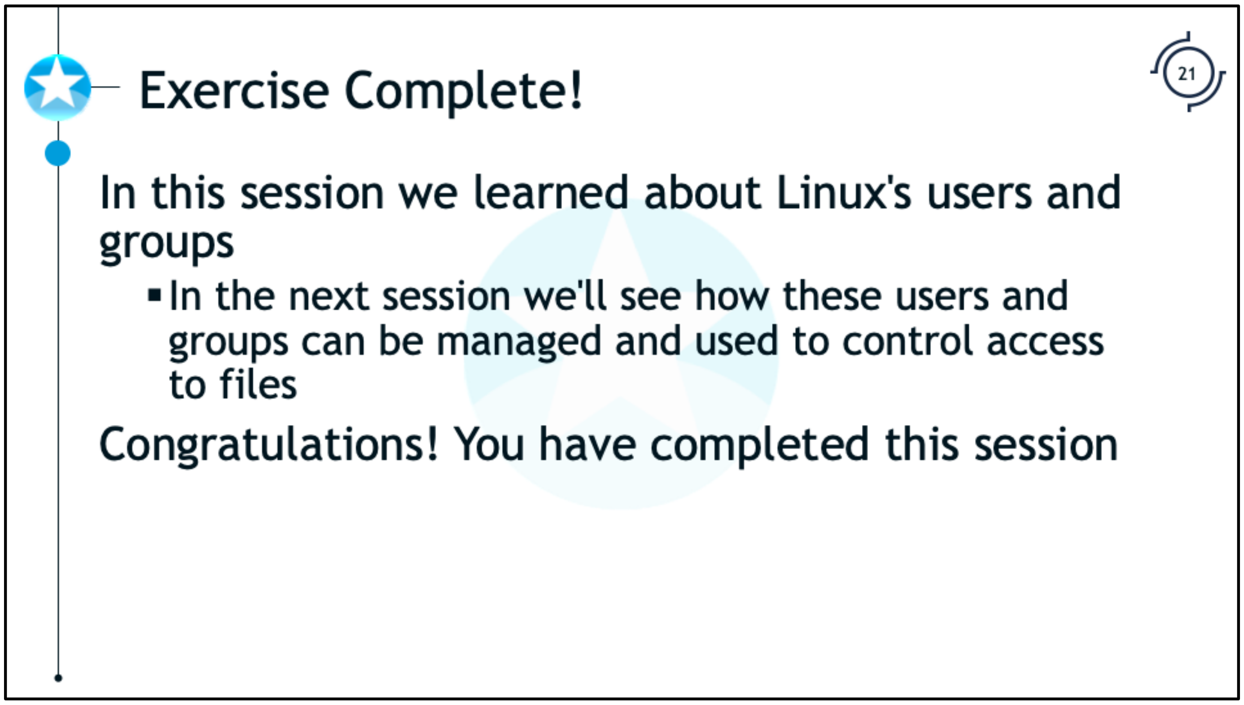
/etc/shadow

The /etc/passwd file is not a safe place to store passwords because any user can read it, whereas only root can read /etc/shadow.

What is the UID of the root user?

0

Don't forget that computers start counting at zero!



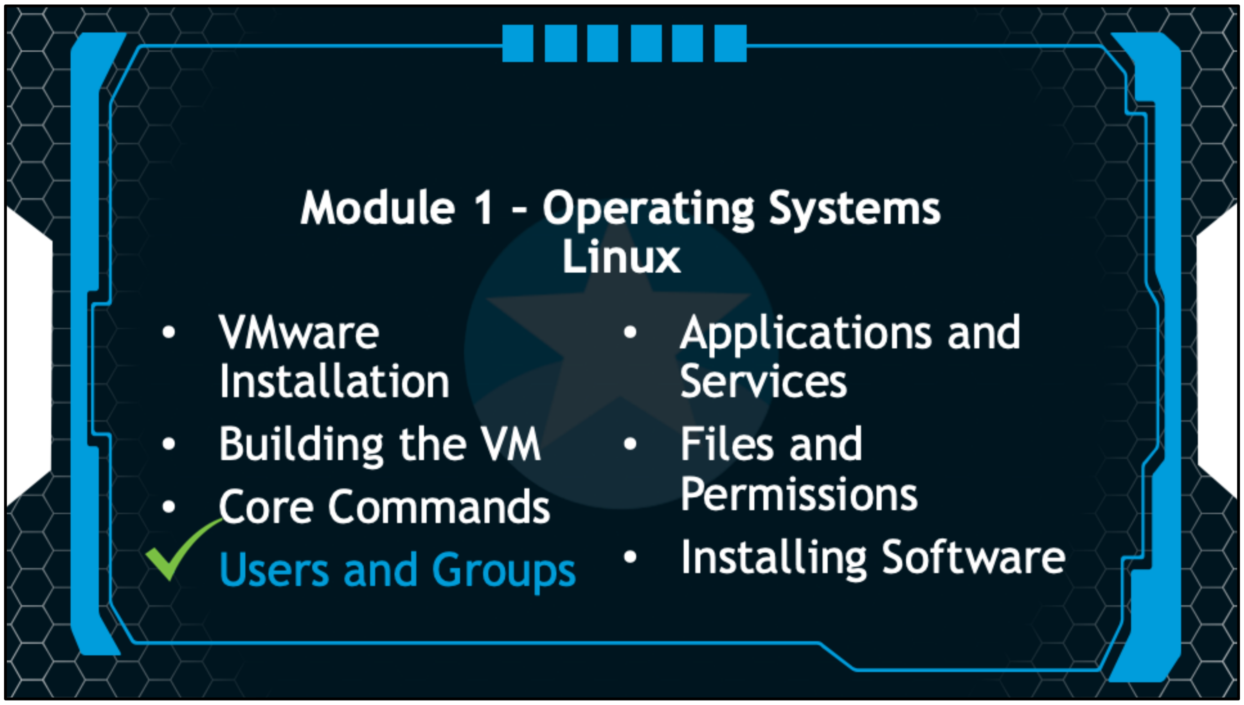
Exercise Complete!

In this session we learned about Linux's users and groups

- In the next session we'll see how these users and groups can be managed and used to control access to files

Congratulations! You have completed this session

Congratulations! You have completed the session on Linux's users and groups.



Module 1 - Operating Systems Linux

- VMware Installation
- Building the VM
- Core Commands
- ✓ Users and Groups
- Applications and Services
- Files and Permissions
- Installing Software

In the next session, we will discuss Linux services and applications.