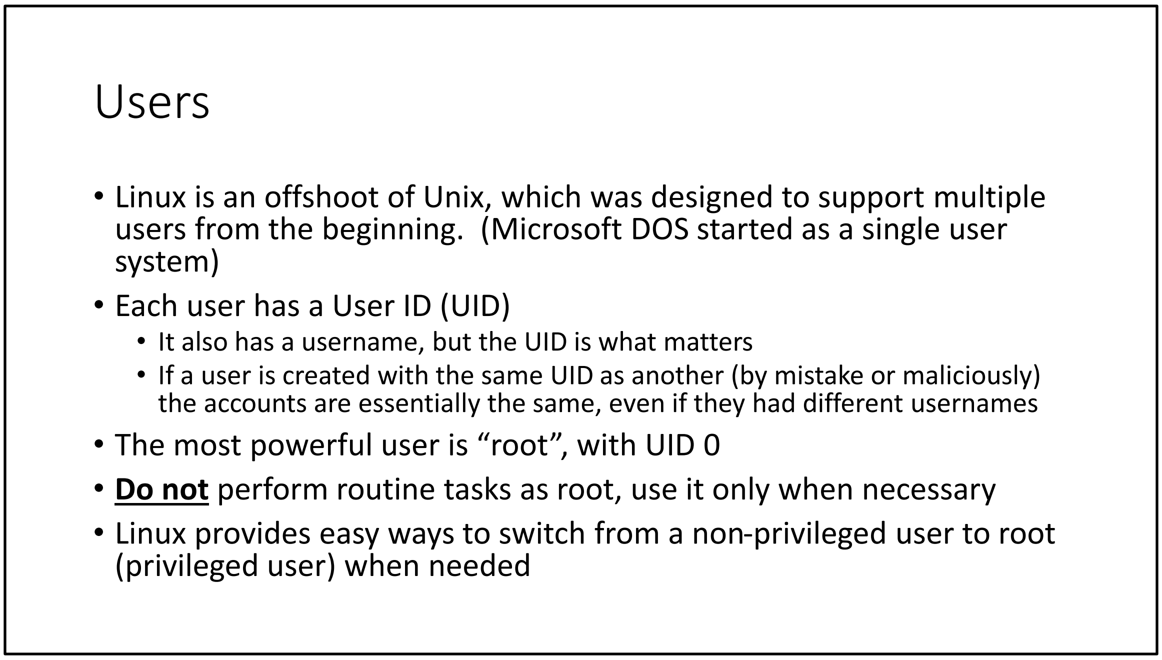
Linux Users and Groups

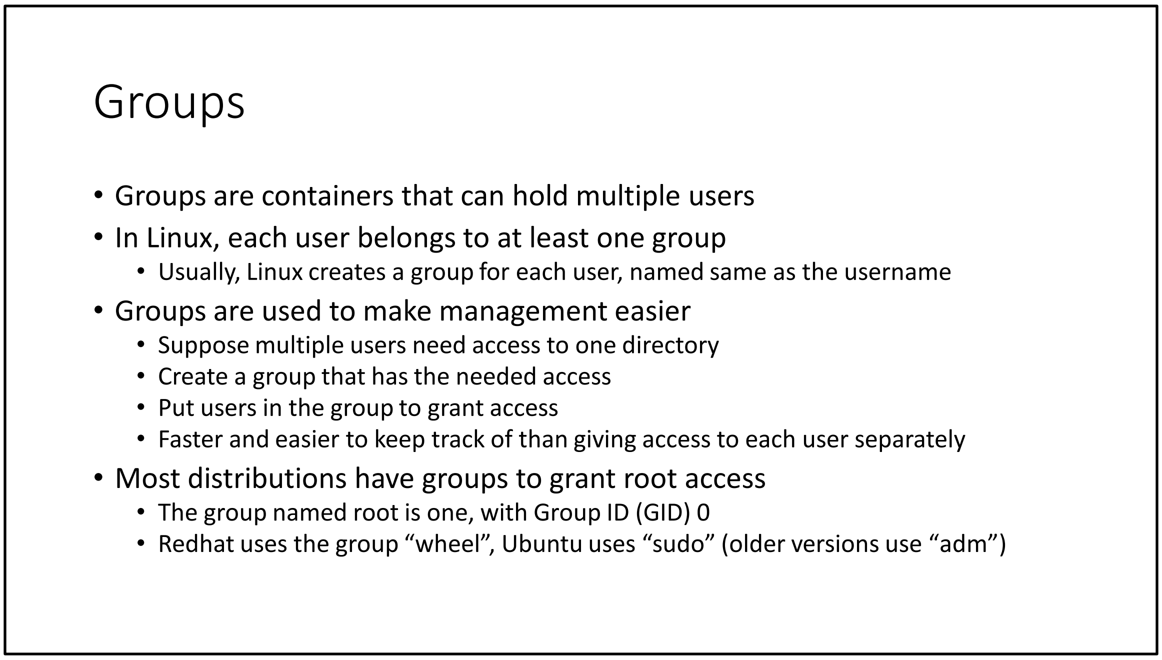


Users

- Linux is an offshoot of Unix, which was designed to support multiple users from the beginning. (Microsoft DOS started as a single user system)
- Each user has a User ID (UID)
  - It also has a username, but the UID is what matters
  - If a user is created with the same UID as another (by mistake or maliciously) the accounts are essentially the same, even if they had different usernames
- The most powerful user is “root”, with UID 0
- **<u>Do not</u>** perform routine tasks as root, use it only when necessary
- Linux provides easy ways to switch from a non-privileged user to root (privileged user) when needed

If you execute a program, say a web browser, as root an attacker who compromises that program will obtain root privileges. So, if you browse or check mail as root and happen upon a page or attachment that can compromise your browser/mail reader, the attacker owns your computer. If the same thing happens while you are browsing as your limited rights user, the attacker only gets those limited rights.

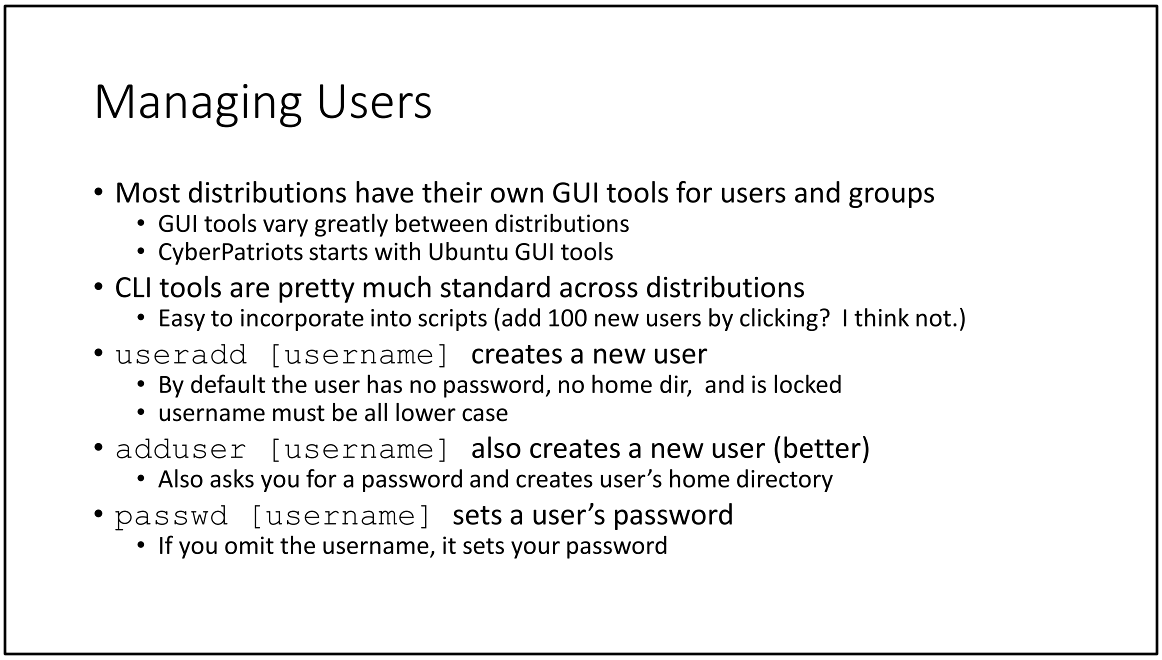
When an attacker owns a computer, they will often create a second account with UID 0 but an innocent looking name. Because the UID is 0, the account is the same as the regular root account.



Groups

- Groups are containers that can hold multiple users
- In Linux, each user belongs to at least one group
  - Usually, Linux creates a group for each user, named same as the username
- Groups are used to make management easier
  - Suppose multiple users need access to one directory
  - Create a group that has the needed access
  - Put users in the group to grant access
  - Faster and easier to keep track of than giving access to each user separately
- Most distributions have groups to grant root access
  - The group named root is one, with Group ID (GID) 0
  - Redhat uses the group “wheel”, Ubuntu uses “sudo” (older versions use “adm”)

It is much better to control access with groups than by assigning access to individual user accounts. Create groups and assign access to the groups. Then control user access by adding or removing the users from groups.



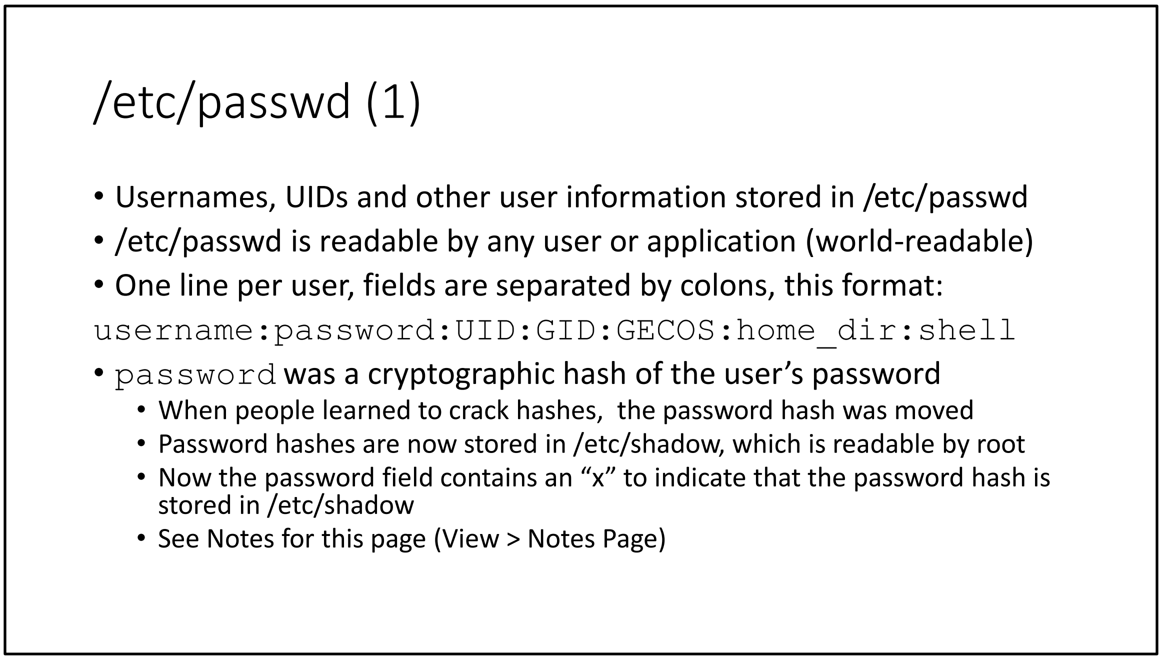
Managing Users

- Most distributions have their own GUI tools for users and groups
  - GUI tools vary greatly between distributions
  - CyberPatriots starts with Ubuntu GUI tools
- CLI tools are pretty much standard across distributions
  - Easy to incorporate into scripts (add 100 new users by clicking? I think not.)
- `useradd [username]` creates a new user
  - By default the user has no password, no home dir, and is locked
  - username must be all lower case
- `adduser [username]` also creates a new user (better)
  - Also asks you for a password and creates user's home directory
- `passwd [username]` sets a user's password
  - If you omit the username, it sets your password

Once you have added one user via the command line, you can see how the command works and what access it needs. Then you can write a script to read a text file of user names (and other required items) and create, modify, or remove large numbers of user accounts at once. Of course, you have to be careful…

Managing Groups

- `groupadd [groupname]` creates a new group
- `gpasswd -a [user] [group]` adds the user to the group
- `gpasswd -d [user] [group]` deletes the user from the group
- A user can belong to multiple groups but always has one primary group
  - primary group used during login and assigned to new files the user creates
- `usermod` also adds a user to group(s) but has a "gotcha"
  - It replaces existing supplementary groups unless you tell it to "append"
  - It can also be used to change a user's primary group
  - https://www.howtogeek.com/50787/add-a-user-to-a-group-or-second-group-on-linux/
  - https://www.cyberciti.biz/faq/howto-linux-add-user-to-group/



/etc/passwd (1)

- Usernames, UIDs and other user information stored in /etc/passwd
- /etc/passwd is readable by any user or application (world-readable)
- One line per user, fields are separated by colons, this format:

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

- `password` was a cryptographic hash of the user's password
  - When people learned to crack hashes, the password hash was moved
  - Password hashes are now stored in /etc/shadow, which is readable by root
  - Now the password field contains an "x" to indicate that the password hash is stored in /etc/shadow
  - See Notes for this page (View > Notes Page)

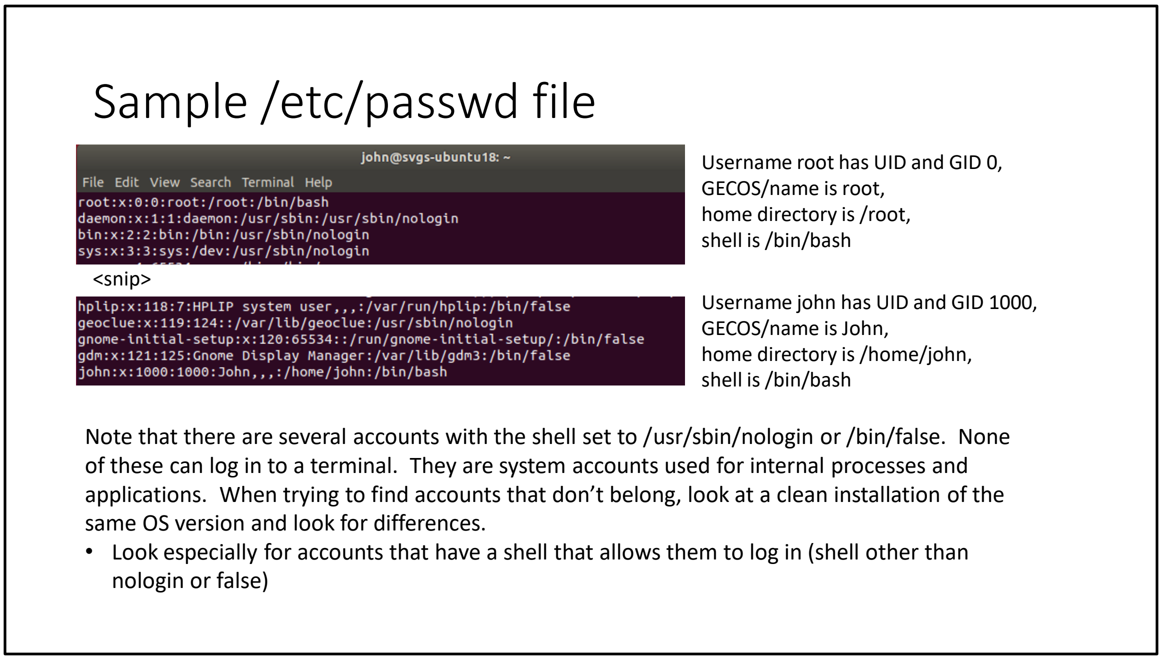
A cryptographic hash function takes a string as an input and generates a new fixed length string called a hash. The function is designed so that it is infeasible (would take a very long time) to compute the original string from the hash. It is a one way function.

When you try to log in, the OS takes the hash of your password and compares it to the hash it has stored. If they match, you are allowed in. This way the OS does not store the password in plain text.

It is nearly impossible to compute the password from the hash. However, people tend to chose passwords that are just common words. A dictionary attack takes a dictionary of common passwords, and computes the hashes for all of them. When it finds a hash that matches the hash for the user, it has found the password. This is called hash cracking.

/etc/passwd (2)

- `UID` is the user's unique ID number
- `GID` is the ID number of the user's primary group
- `GECOS` is a comment field, often the user's full name
  - This dates back to Unix at Bell Labs (1960's), which used print servers that ran the General Electric Comprehensive Operating Supervisor (GECOS). The GECOS field held the user's ID on the print server.
- `home_dir` is the path to the user's home directory (usually /home/username)
- `shell` is the path to the user's login shell, the program run when the user opens a terminal, often /bin/bash or /bin/sh



Sample /etc/passwd file

```
john@svgs-ubuntu18: ~
File  Edit  View  Search  Terminal  Help
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
```

Username root has UID and GID 0,
GECOS/name is root,
home directory is /root,
shell is /bin/bash

<snip>

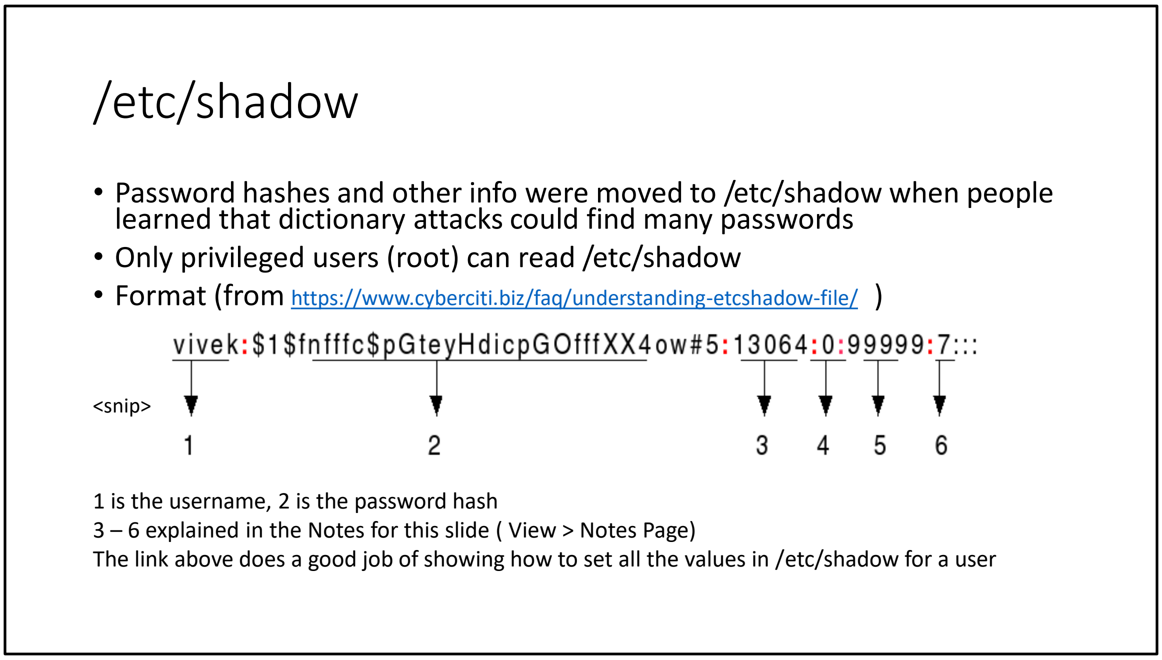
```
hplip:x:118:7:HPLIP system user,,,:/var/run/hplip:/bin/false
geoclue:x:119:124::/var/lib/geoclue:/usr/sbin/nologin
gnome-initial-setup:x:120:65534::/run/gnome-initial-setup/:/bin/false
gdm:x:121:125:Gnome Display Manager:/var/lib/gdm3:/bin/false
john:x:1000:1000:John,,,:/home/john:/bin/bash
```

Username john has UID and GID 1000,
GECOS/name is John,
home directory is /home/john,
shell is /bin/bash

Note that there are several accounts with the shell set to /usr/sbin/nologin or /bin/false. None of these can log in to a terminal. They are system accounts used for internal processes and applications. When trying to find accounts that don't belong, look at a clean installation of the same OS version and look for differences.

- Look especially for accounts that have a shell that allows them to log in (shell other than nologin or false)

You can change user accounts by modifying the /etc/passwd file directly if you have root access. In general it is better to do that work with command line tools as there is a smaller chance of error.



/etc/shadow

- Password hashes and other info were moved to /etc/shadow when people learned that dictionary attacks could find many passwords
- Only privileged users (root) can read /etc/shadow
- Format (from https://www.cyberciti.biz/faq/understanding-etcshadow-file/)

```
vivek:$1$fnfffc$pGteyHdicpGOfffXX4ow#5:13064:0:99999:7:::
```

<snip>

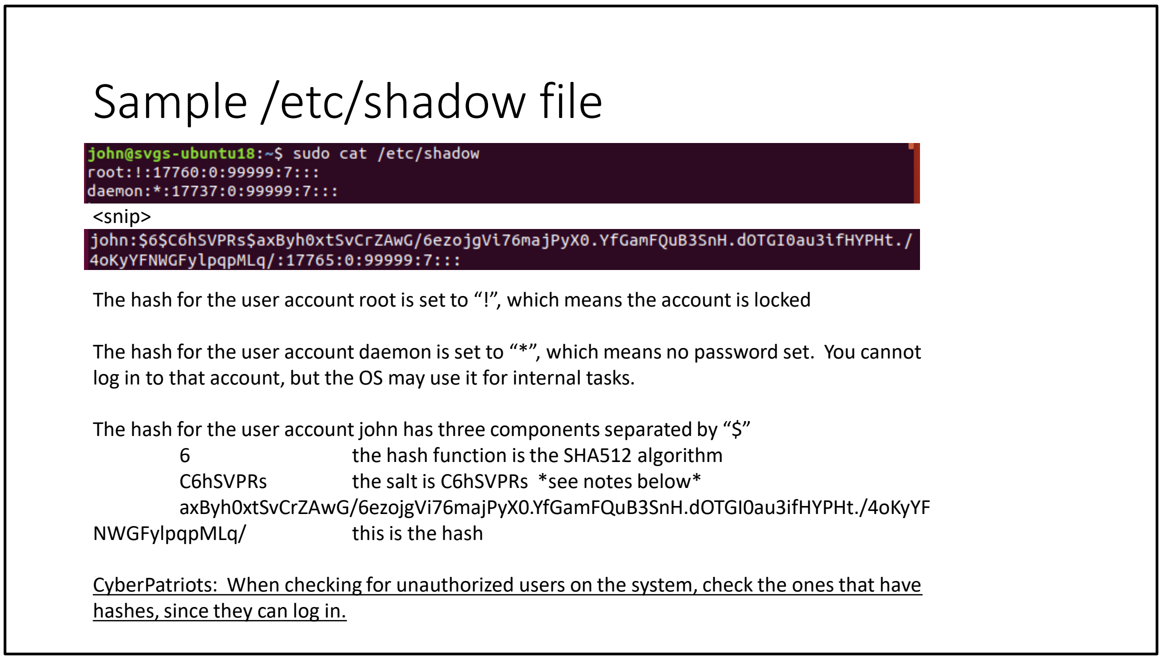
1 is the username, 2 is the password hash
3 – 6 explained in the Notes for this slide (View > Notes Page)
The link above does a good job of showing how to set all the values in /etc/shadow for a user

Username : It is your login name.
Password : It is your encrypted password. The password should be minimum 8-12 characters long including special characters, digits, lower case alphabetic and more. Usually password format is set to $id$salt$hashed, The $id is the algorithm used On GNU/Linux as follows:

  $1$ is MD5
  $2a$ is Blowfish
  $2y$ is Blowfish
  $5$ is SHA-256
  $6$ is SHA-512

Last password change (lastchanged) : Days since Jan 1, 1970 that password was last changed
Minimum : The minimum number of days required between password changes i.e. the number of days left before the user is allowed to change his/her password
Maximum : The maximum number of days the password is valid (after that user is forced to change his/her password)
Warn : The number of days before password is to expire that user is warned that his/her password must be changed
Inactive : The number of days after password expires that account is disabled
Expire : days since Jan 1, 1970 that account is disabled i.e. an absolute date specifying

when the login may no longer be used.



Sample /etc/shadow file

```
john@svgs-ubuntu18:~$ sudo cat /etc/shadow
root:!:17760:0:99999:7:::
daemon:*:17737:0:99999:7:::
```

<snip>

```
john:$6$C6hSVPRs$axByh0xtSvCrZAwG/6ezojgVi76majPyX0.YfGamFQuB3SnH.dOTGI0au3ifHYPHt./
4oKyYFNWGFylpqpMLq/:17765:0:99999:7:::
```

The hash for the user account root is set to "!", which means the account is locked

The hash for the user account daemon is set to "*", which means no password set. You cannot log in to that account, but the OS may use it for internal tasks.

The hash for the user account john has three components separated by "$"

6	the hash function is the SHA512 algorithm
C6hSVPRs	the salt is C6hSVPRs *see notes below*
axByh0xtSvCrZAwG/6ezojgVi76majPyX0.YfGamFQuB3SnH.dOTGI0au3ifHYPHt./4oKyYFNWGFylpqpMLq/	this is the hash

CyberPatriots: When checking for unauthorized users on the system, check the ones that have hashes, since they can log in.

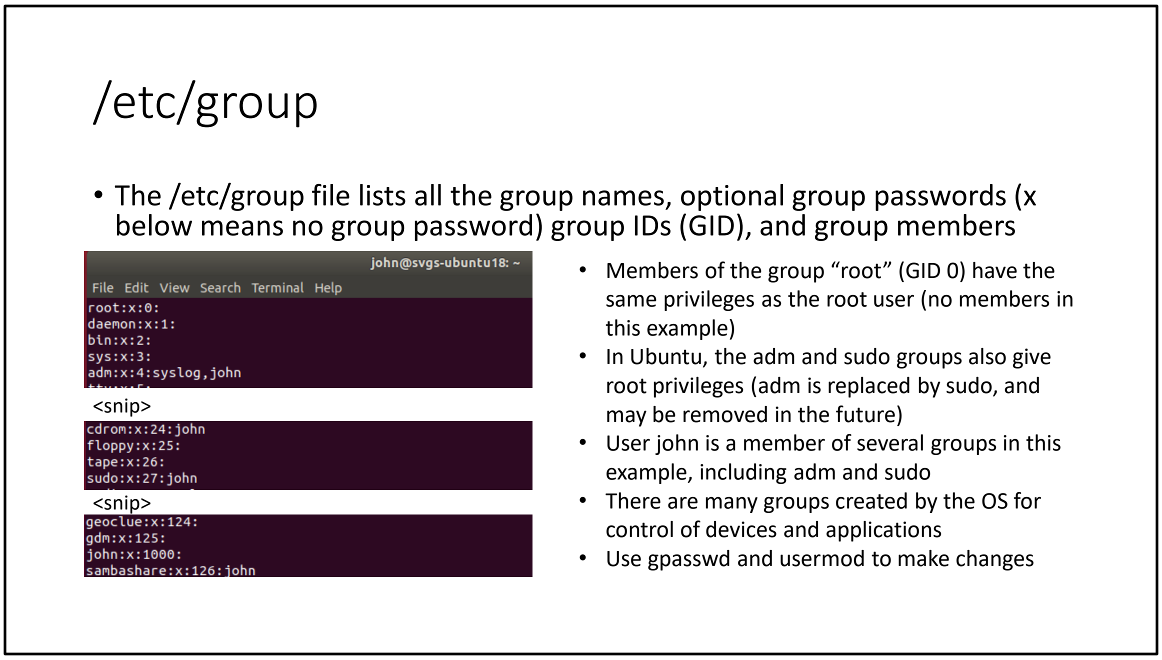
Salted Hashes

Hash functions are deterministic; if the input string is the same, the output hash will be the same every time. If the password is the same (it is "password" in the example above) the hash will always be the same. In that case an attacker can take a dictionary of common passwords and pre-compute the hashes ahead of time. Cracking hashes would be much easier.

The salt is chosen randomly, but is not secret. It is added to the password as input to the hashing function. Since the salt is random, the thousands of users that have chosen "password" as their password will have different hashes. An attacker will not be able to use a table of precomputed hashes.

An attacker can still crack the hashes that have salts; they just have to crack every hash separately since the salt is different each time.

For reasons unknown, Windows hashes do not use salts. Therefore precomputed hashes, known as "rainbow tables" work well



/etc/group

- The /etc/group file lists all the group names, optional group passwords (x below means no group password) group IDs (GID), and group members

john@svgs-ubuntu18: ~
File Edit View Search Terminal Help

```
root:x:0:
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:syslog,john
```

<snip>

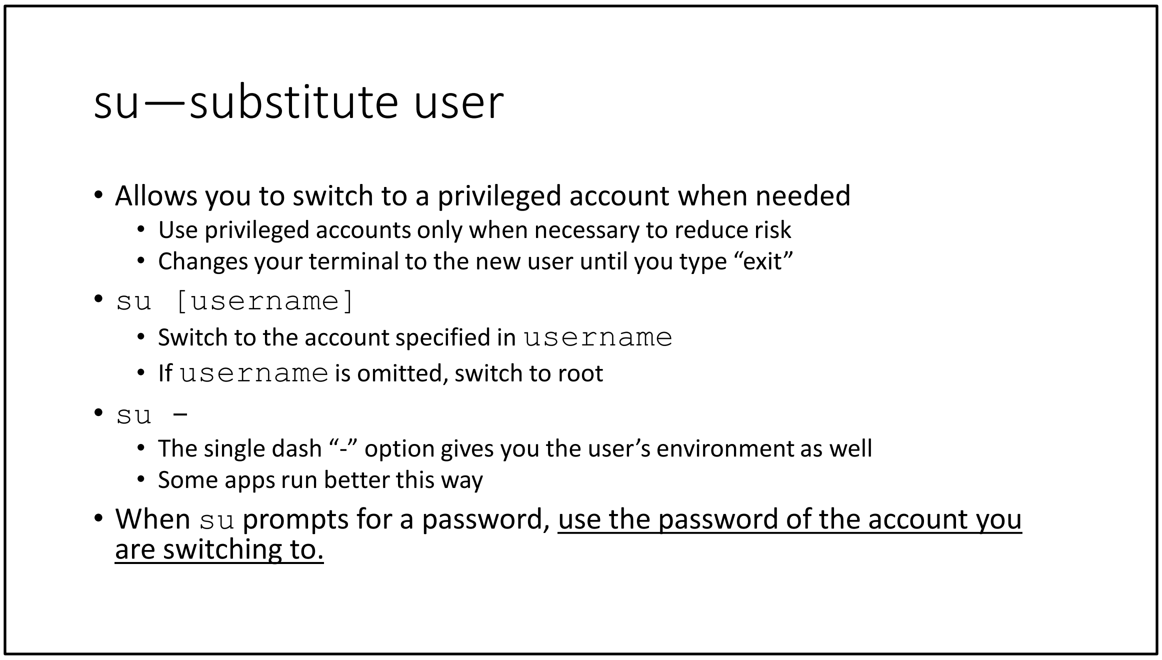
```
cdrom:x:24:john
floppy:x:25:
tape:x:26:
sudo:x:27:john
```

<snip>

```
geoclue:x:124:
gdm:x:125:
john:x:1000:
sambashare:x:126:john
```

- Members of the group “root” (GID 0) have the same privileges as the root user (no members in this example)
- In Ubuntu, the adm and sudo groups also give root privileges (adm is replaced by sudo, and may be removed in the future)
- User john is a member of several groups in this example, including adm and sudo
- There are many groups created by the OS for control of devices and applications
- Use gpasswd and usermod to make changes

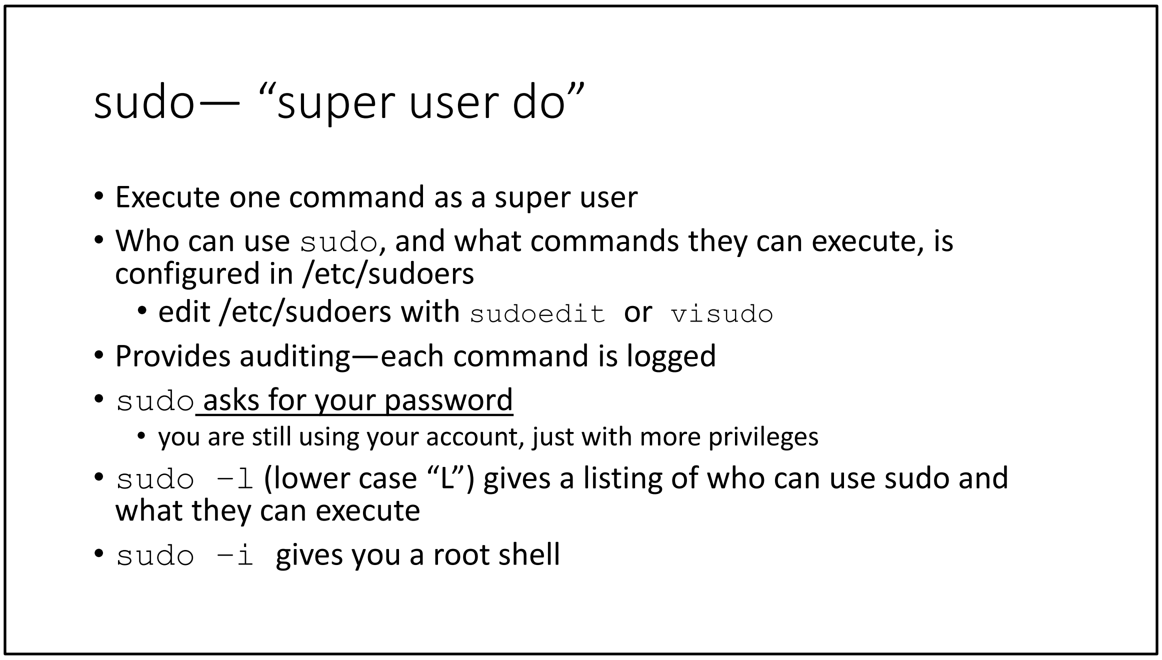
You can modify groups and group memberships by editing the etc/group file directly. As before, it is usually better to use the command line tools to reduce the chance of error.



su—substitute user

- Allows you to switch to a privileged account when needed
  - Use privileged accounts only when necessary to reduce risk
  - Changes your terminal to the new user until you type “exit”
- `su [username]`
  - Switch to the account specified in `username`
  - If `username` is omitted, switch to root
- `su -`
  - The single dash “-” option gives you the user’s environment as well
  - Some apps run better this way
- When `su` prompts for a password, <u>use the password of the account you are switching to.</u>

Always log in with a regular account and use su or sudo when you need elevated privileges. sudo (or su –i) is useful when you just need to run a command or two, sudo when you need to run many commands as root.



sudo— "super user do"

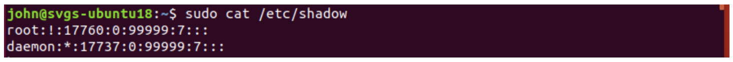
- Execute one command as a super user
- Who can use `sudo`, and what commands they can execute, is configured in /etc/sudoers
  - edit /etc/sudoers with `sudoedit` or `visudo`
- Provides auditing—each command is logged
- `sudo` asks for your password
  - you are still using your account, just with more privileges
- `sudo –l` (lower case "L") gives a listing of who can use sudo and what they can execute
- `sudo –i` gives you a root shell

Sometimes administrators will mistakenly give excessive sudo rights to an account. That happens a lot in Capture The Flag (CTF) challenges. It is always a good idea to run `sudo –l` when you are on a new machine.

Editing the /etc/sudoers file manually is not recommended. If you make a mistake, you can render the system unuseable. Instead, use sudoedit or visudo. If you are just trying to give a user root privileges via sudo, look at the line in the /etc/sudoers file that gives rights to root and copy it. In Ubuntu it's easier to just add the user to the sudo group.

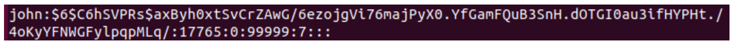
Ubuntu Uniqueness

- Ubuntu is used by brand new Linux users as well as power users
- Ubuntu encourages use of sudo for all actions requiring elevated privileges, so it locks the root account by default
- `sudo -i` will give you a full root shell for multiple commands
- `sudo su -` works the same as `su -` does in other distributions



```
john@svgs-ubuntu18:~$ sudo cat /etc/shadow
root:!:17760:0:99999:7:::
daemon:*:17737:0:99999:7:::
```

<snip>



```
john:$6$C6hSVPRs$axByh0xtSvCrZAwG/6ezojgVi76majPyX0.YfGamFQuB3SnH.dOTGI0au3ifHYPHt./
4oKyYFNWGFylpqpMLq/:17765:0:99999:7:::
```

The root user has an "!" where the hash should be, so it has no password and cannot log in. To make Ubuntu work like other distributions, set the root password with `sudo passwd root`. To return to normal, lock the account with `sudo passwd -l root`.