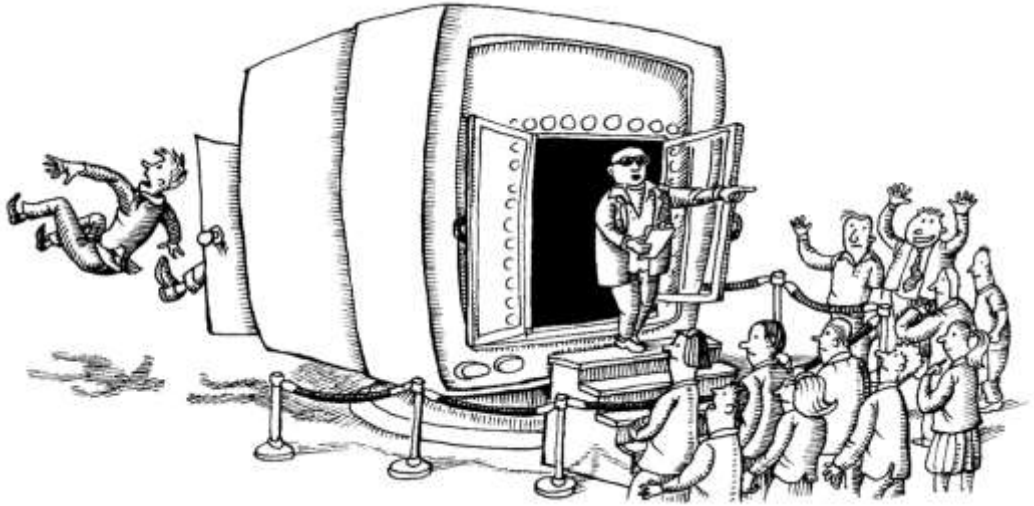
# CS183

Instructor: Ali Davanian
(Slides were adopted from Brian Crites and Alireza Abdoli)



# User Management

# What is a User?

- At its core, a user is an unsigned 32-bit integer value (the UID)
- The system provides an API abstraction for login services to retrieve user information from user backends
  - The functions `getpwuid()` and `getpwnam()` allow you to lookup info via UID or login name, respectively
- This allows for the creation of new login services that can interface with existing user backends and vice-versa
- The functions return a record the login name, UID, GID, password hash, home directory, and shell program (and a few other things)

# How Login Works

- When a user logs in, the following steps happen
  - The user provides their login name and password
  - The login service (windows server, login, getty, etc.) makes a call to `getpwnam()` supplying the login name
  - The `getpwnam()` returns the record for the given login name from a user backend (passwd file, LDAP, Windows AD, etc.)
  - The login service checks the hashed input password against the hash stored in the returned record
  - If they match, the user is put into the home directory and attached to the shell specified in the record

# The /etc/passwd File

- Text file which represents the original user backend and contains the following information
  - Login name
  - Encrypted password placeholder
  - UID number
  - Default GID number
  - Optional GECOS information
  - Home directory
  - Login shell

# User ID Rules

- Each user must have a unique UID, most user addition systems handle this automatically
- UIDs for users should be the same across the entire fleet.
  - For small fleets this coordination can be accomplished by allocating groups of UIDs to different teams to distribute as necessary
  - For large fleets specific user backends should be used to guarantee uniqueness and coordinate UIDs across the fleet
- Don’t recycle UIDs, even when the users associated with them are removed from the system. This prevents confusion when backups restore files that belonged to removed users

# Home Directory & Login Shell

- Home directory is the default directory the user will be redirected to when they log in and holds account specific customizations
  - Shell aliases, environmental variables, SSH keys, server fingerprints, etc.
- If a home directory is unavailable when logging in (for instance it is a NFS that isn’t currently connected) the login may dump the user into the base directory or disallow the login (system specific)
- Login shell is typically a command interpreter, but can technically be any program

# The /etc/group File

- Lists all the systems groups and group members with the following fields:
  - Group name
  - Encrypted password or placeholder
  - GID number
  - List of members, separated by commas (no spaces)
- Group passwords can be set to allow users to join a group themselves (!)
- To increase security, users are typically given their own individual group and then optionally put into other groups for file sharing (!)
- You can also use groups for sharing permissions such as adding users to an admin group and giving that group root permissions in the sudoers file (!)
- GIDs should also be consistent across the fleet, with increased difficulty do to different OS using different GIDs for the same service groups

# Password Evolution

- Originally systems encrypted user passwords with DES and held the hash in the passwd file directly
- As computing power increased, those passwords became trivial to crack
- Systems then moved to MD5 cryptography and hidden password hashes
- Weakness have been discovered in the MD5 cryptography scheme, so systems have switched to salted SHA-512 cryptography as the default password hashing scheme

All UNIX versions provide a method for changing the encryption scheme and you can additionally use PAM to add additional authentication methods

# The /etc/shadow File

- File only viewable by root which which serves as an extension to the /etc/passwd file containing password centric information such as
  - **Login in**
  - **Encrypted password**
  - Date of last password change
  - Minimum number of days between password changes
  - Maximum number of days between password changes
  - Number of days in advance to warn users about password expiration
  - Days after password expiration that account is disabled
  - Account expiration date
  - Field reserved for future use which is currently always empty
- The `pwconv` utility syncs the contents of the shadow and passwd files

# Question

What steps do you need to do to add a new user?

# Manually Adding a User

- Creating a user requires performing the following steps:
  - **Have the new user sign your user agreement and policy statement**
  - **Edit the passwd and shadow files to define the user's account**
  - Add the user to the /etc/group file
  - **Set an initial password**
  - **Create, chown, and chmod the user's home directory**
  - Configure roles and permissions
  - Copy default startup files to the user's home directory
  - **Verify that the account is set up correctly**
  - **Document the user's contract information and account status**

# Scripts for Adding Users

- Use the commands useradd, usermod, and userdel to perform these tasks for you in an automated (and configurable) fashion
- There are two different configuration files
  - `/etc/login.defs` which is a file that is modified directly
    - Password aging, choice of encryption algorithms, location of mail spool files, preferred ranges of UIDs and GIDs
  - `/etc/default/useradd` which is modified using the useradd function
    - Location of home directories, default shell for new users
- There are additional useradd flags for modifying its function over the configs including setting groups, GECOS information, etc.

# Manually Removing a User

- Removing a user requires performing the following steps:
  - Remove the user from any local user databases (or phone lists)
  - Remove the user from mail aliases database (or add forwarding address)
  - Remove the user's crontab file and any pending `at` jobs or print jobs
  - Kill any of the user's processes that are still running
  - Remove the user from the `passwd`, `shadow`, and `group` files
  - Remove the user's home directory
  - Remove the user's mail spool (only for local mail servers)
  - Delete or transfer ownership of any mailing lists run by the delete user
- Remember that files can have shared permissions, so its best to move or backup user files before removing them

# Scripts for Removing and Locking Users

- Use the `userdel` command to remove users from the system, although different Linux variants offer additional and more powerful versions
- You may need to lock a users account, perhaps because you suspect it of being compromised, which you can do by appending a `*` or `!` before their password in the `shadow` file or by using `usermod -L` (to lock) and `usermod -U` (to unlock) which accomplishes the same thing
- This only causes the users login to fail, it may be more informative to replace the users shell with a program which prints why their account has been locked, however this will not disallow processes that ignore the shell

# Centralized Account Management

- Lightweight Directory Access Protocol (LDAP) is a database-like repository that stores user management and related data and servers as a central location for user information and coordination
- Single sign-on (SSO) allows the user to authenticate once and multiple applications to utilize those credentials for validation (UCR’s CAS is an example of this system)
- Identity and Access Management (IAM) represents systems for identifying, authenticating, and giving permissions to users and is generally implemented as a commercial system

Questions?

# Additional Resources

getpwnam man page