Power, Responsibility, Ethics

Superuser or root

- Privileged account with unrestricted access to all files and commands
- Powerful and dangerous
  - Unlimited powers to the capabilities of the system
    - Can be dangerous in wrong or inexperienced hands
    - *Never do your routine work as root*
  - Have to make decisions such as when to kill a process that is using too much CPU time
  - Responsible to make backups
  - Monitor system logs
  - Plan and make changes reversible; test changes in a nonproduction environment before release

Ownership of files and processes

- User owner (uid) and group owner (gid), with two ownerships decoupled
- Only owner can modify the permissions on the files
- Effective uid and gid for processes
  - Real uid/gid used for accounting
  - Effective uid/gid used for access privileges
    - Identity of the process is *promoted* for a command using the suid and sgid bits
- User owner of a process can send it signals and reduce its scheduling priority (by increasing the nice value)

Superuser

- Defined by uid 0 on the account in the file `/etc/passwd`
- Additional usernames with uid 0 will act as superusers as well
  - Not recommended from system security viewpoint
- Operations restricted to superuser only
  - Changing root directory of a process with `chroot(1M)`
  - Creating device files
  - Setting system clock
  - Raising resource usage limits and process priorities
  - Setting system's hostname
  - Configuring network interfaces
  - Shutting down the system
- Any process being run as root can change its ownership
  - Example: login process

  - Cannot recover to its former state after change

Choosing a root password

- Follow good password practices
- *Shocking nonsense*
- Change the password
  - Every three months or so
  - Any time someone with the knowledge of root password leaves site
  - Whenever security is compromised

Becoming root

- Logging in directly
  - Bad idea
  - No record of operations performed as root
  - No record of who did the work
  - Most systems disable root logins on terminals and across the network
    - Only allow direct root login on system console
    - File `/etc/default/login` on Solaris
    - File `/etc/securetty` on RH Linux and HP-UX
- Using the command `su(1M)`
  - Better than direct login
  - Records who became root with the timestamp (`/var/adm/sulog`)
  - No record of work performed as root
  - Must belong to a group `wheel` to use `su(1M)`
  - `su` with and without –
    - Without –, the shell created by `su` inherits the environment from the current shell
    - With –, the new shell simulates the actual root login
  - Convenient to execute a single command with the `-c` option
    - Command should be enclosed in quotes if it contains spaces
  - Some systems disable `su` by changing permissions on the command
- Using `sudo(8)`
  - Limited form of `su(1M)`
  - Allows a set of people (in file `/etc/sudoers`) to perform a limited set of commands as root
    - User can check the command availability by issuing `sudo -l`
  - Records the user who issued the command as well as the command
    - Possible through `syslog(3)`, or to a file, or both
  - User has to supply his/her own password

  - Repeated commands as `sudo(8)` do not require the password for up to a certain time period (configurable with a default 5 minutes)
    - The time period can be extended by issuing the command `sudo -v`
    - The time period can be terminated by `sudo -K`
  - Possible to configure `sudo` for multiple machine with a single `sudoers` file
  - Make sure that you specify the commands allowed to be executed by users with absolute path
    - Do not allow any commands that will provide shell escape to users
  - `sudoers` file should be modified by the command `visudo`
  - Advantages of `sudo`
    - Improved accountability due to command logging
    - People do not always have to log in as root
    - Real root password is protected; known to only the real root
    - Privileges can be revoked without changing root password
    - Faster and convenient
    - List of users with root privileges is maintained
    - Less chance of root shell being left unattended
    - A single file can be used to control access to entire network
  - Disadvantages
    - A compromised `sudoer` account can compromise security
    - Command logging can be subverted by shell escapes from within a program
- *Never leave a session logged in as root unattended*

Other pseudo-users

- Identified with an asterisk in the password file to prevent logins
- `daemon`
  - Owns unprivileged system software
  - uid 1
  - Avoids security hazards with root ownership
- `bin`
  - Owns system commands
  - Most modern systems use root as owner
- `sys`
  - Owns kernel and memory images (`/dev/kmem`, `/dev/mem`, `/dev/drum`, `/dev/swap`)
- `nobody`
  - Generic NFS user
  - uid -1 or -2
  - NFS uses `nobody` to represent root users on other systems for file sharing
  - Prevents remote roots from using their rootly powers on local machine
  - Represents a generic and powerless user
  - Should not own any files because remote roots can exploit those

  - Some daemons (`fingerd`) run as `nobody`

Communicating with users

- Sending a message to a user
  - The `write(1)` command
  - The recipient may disable messages by using the command `mesg n`
- Sending a message to all users
  - Use the `wall(1M)` (write all) command
  - `rwall(1M)` should be used for local subnet
- Message of the day
  - Specified in the file `/etc/motd`
  - Used to make announcements at login time, such as new software and scheduled downtime
  - Keep it short to enable users to read it
  - Can be suppressed by using an empty file `.hushlogin` in user's home directory
- Pre-login message
  - Specified in the file `/etc/issue`
  - Displays before the login prompt
- Email
  - Used to indicate non-urgent messages
  - Can be used to draw attention of a specified group of users to an issue