Example - login run as root

Unix users typically stored in files in `/etc` files `passwd`, `group`, and often `shadow` or `master.passwd`

For each user, files contain

- Textual username (e.g., "`dm`", or "`root`")
- Numeric user ID, and group ID(s)
- One-way hash of user's password: {`salt`; H`(salt; passwd)`}
- Other information, such as user's full name, login shell, etc.

For instance `/usr/bin/login` runs as root

- Reads username & password from terminal
- Looks up username in `/etc/passwd`, etc.
- Computes H`(salt; typed password)` & checks that it matches
- If matches, sets group ID & user ID corresponding to username
- Execute user's shell with `execve` system call

Setuid

Some legitimate actions require more privileges than UID
e.g. how users change their passwords stored in root-owned `/etc/passwd` and `/etc/shadow` files?

➡ Solution - `setuid` and `setgid` programs

- Run with privileges of file's owner or group
- Each process has real and effective UID/GID
- Real is user who launched `setuid` program
- Effective is owner/group of file, used in access checks

Shown as "`s`" in file listings

```
-rws--x--x 1 root root 52528 Oct 29 08:54 /bin/passwd
```

- Obviously need to own file to set the `setuid` bit
- Need to own file and be in group to set `setgid` bit