Information about files

* Example: directory listing with ls -l
* File type: first char
* File permissions: next nine chars
* Link counter: the number of links (ie directory entries pointing to) the file
* Username of the owner: usually the user that has created the file
* Group: depending on the version of Unix, a newly created file belongs to its creator’s group or to its directory’s group
* File size, modification, filename
* Owner and root can change permissions (chmod)
* Root can change file owner and group (chown)

Need for privileged programs

* Password dilemma
  + Permissions of /etc/shadow file
  + Only writable to the owner
  + How would normal users change their password?
* Privileged programs
  + Implementing fine-grained access control in operating systems makes OS over complicated
    - OS relies on extensions to enforce fine-grained access control
    - Privileged programs are such extensions
  + Types of **privileged programs**
    - **Daemons / services**
      * Computer program that runs in the background
      * Needs to run as root or other privileged users
    - **Set-UID programs**
      * Widely used in UNIX systems
      * Program marked with a special bit

Set-UID concept

* Allows user
  + To run a program with the program owner’s privilege
  + To run programs with temporary elevated privileges
* Every process has two user IDs
  + Real UID (RUID): identifies real owner of the process
  + Effective UID (EUID): identifies privilege that a process will be executed with
    - Access control is based on euid
* When a normal program is executed, RUID = EUID
  + They both equal to the ID of the user who runs the program
* When a set-UID is executed, RUID != EUID
  + RUID still equals the user’s ID, but EUID equals the program owner’s ID
  + If the program is owned by root, the program runs with the root privilege

Example: turn a program into set-UID

* **Slide 10-12 (lec 22)**
* A set-UID program is just like any other program
  + Except it has a special marking bit called set-UID bit

How is set-UID secure?

* Allows normal user to escalate privileges
  + This is different from directly giving the privilege (sudo)
  + Restricted behavior

Attacks via user inputs

* User inputs: explicit inputs
  + Buffer overflow -> overflowing a buffer to run malicious code
* Chsh – change shell
  + Set-UID program with the ability to change default shell programs
  + Shell programs are stored in /etc/passwd file
  + Issues
    - Failing to sanitize user inputs -> inputs may contain multiple lines
    - Attackers could create a new root account
      * Put 0s in the third and fourth fields (the user ID and group ID fields)

Attacks via system inputs

* Symbolic link to privileged file from an unprivileged file
* Influence programs
* Writing inside world writable folder

Attacks via environment variables

* Environment variables
  + A set of dynamic named values
  + Part of the operating environment in which a process runs
  + Affect the way that a running process will behave
  + Example: PATH variable
    - When a program is executed, the shell process will use the environment variable to find where the program is, if the full path is not provided
* Behavior can be influenced by inputs that are not visible inside a program
* Shell programs’ behavior is affected by many environment variables
  + The most common of which is the PATH variable
* When a shell program runs a command and the absolute path is not provided, it uses the PATH variable to locate the command
* **Example on slide 19-20**

Capability leaking

* In some cases, privileged programs downgrade themselves during execution
* Example: the su program
  + This is a privileged set-UID program
  + Allows one user to switch to another user
  + Program starts with EUID as root and RUID as user1
  + After password verification, both EIUD and RUID become user2’s (via privileged downgrading)
* Such programs may lead to capability leaking
  + programs may not clean up privileged capabilities before downgrading
* **attacks via capability leaking: slide 22-2**4
  + how do we fix the problem
    - destroy the file descriptor before downgrading the privilege (close the file)

invoking other programs

* applications may invoke external commands from inside a program
  + ex:
    - a privileged program may need to send an email to user
    - Invoke an external email program
* Attack:
  + Users are often asked to provide input data to the command
  + If the command is not invoked properly, user’s input data may be turned into command name
* Invoking programs: unsafe approach
  + Scenario: use cat to view files
  + The easiest way:
    - Invoke an external command is the system() function
    - This program uses the /bin/cat program
    - Should be a root-owned set-UID program (to view restricted files)
    - The program can view all files, but it can’t write to any file
    - **Slide 27-3**0
* Invoking programs: the safe approach
  + Use execve()
  + Why is it safe?
    - Code (command name) and data are sanitized separated
    - There is no way for the user data to become code

Set-UID & dynamic linking

* Dynamic linking
  + The linking is done during runtime
  + Before a program compiled with dynamic linking is run, its executable is loaded into memory first
  + This means that a part of the program’s code is undecided during the compilation time
* If the user can influence the missing code, they can (possibly) compromise the integrity of the program
* LD\_PRELOAD contains a list of shared libraries which will be searched first by the linker
* If not all functions are found:
  + The linker will search among several lists of folder including the one specified by LD\_LIBRARY\_PATH
* Both variables can be set by users
  + Users get an opportunity to control the outcome of the linking process
* If that program were a set-UID program, it may lead to security breaches
* **Example on slide 35-40**

Privileged programs: remarks

* **Principle of isolation**: don’t mix code and data
  + System() code execution
  + Buffer overflow attacks
  + Cross site scripting
  + SQL injection
* **Principle of least privilege**: a privileged program should be given the power which is required to perform its tasks
  + Disable the privileges (temporarily or permanently) when a privileged program doesn’t need those
  + In linux, seteuid() and setuid() can be used to disable / discard privileges