System Security

Waltzing Access Control in Unix 1/2

Mohamed Sabt

Univ Rennes, CNRS, IRISA

2023 / 2024

Part I

Introduction

What is Access Control

- The ability to allow only authorized users, programs or processes.

- The granting or denying, according to a particular security model, of certain permissions to access a resource.

- An entire set of procedures performed by hardware and software to
  - Identify users requesting access,
  - Record access attempts,
  - Monitor access,
  - Grant or deny access based on pre-established rules.

Examples of Access Control

- Operating Systems:
  - One user cannot arbitrarily access another user's files; a normal user cannot kill another user's processes.

- Web Browsers:
  - For instance, a code from one website cannot access the cookies from another website.

- Firewalls:
  - Firewalls inspect every incoming (sometimes outgoing) packet. If a packet matches with certain conditions, it will be dropped by the firewalls.

- Memory Protection:
  - In modern processors, code in Ring 3 cannot access the data in another more privileged region (e.g. Ring 0).

Overview of System Access Control

- All operating systems have an access control component.
- The system must first authenticate a user seeking access.
- Then, the access control function determines if the specific requested access by this user is permitted.
  - The access control function consults a database to determine whether to grant access.
- An auditing function monitors and keeps a record of user accesses to system resources (logs).

What is a Security Model

- A model describes the system
  - A high level specification or an abstract machine description of what the system does.

- A security policy
  - Defines the security requirements for a given system.

- Verification techniques that can be used to show that a policy is satisfied.

- System Model + Security Policy = Security Model

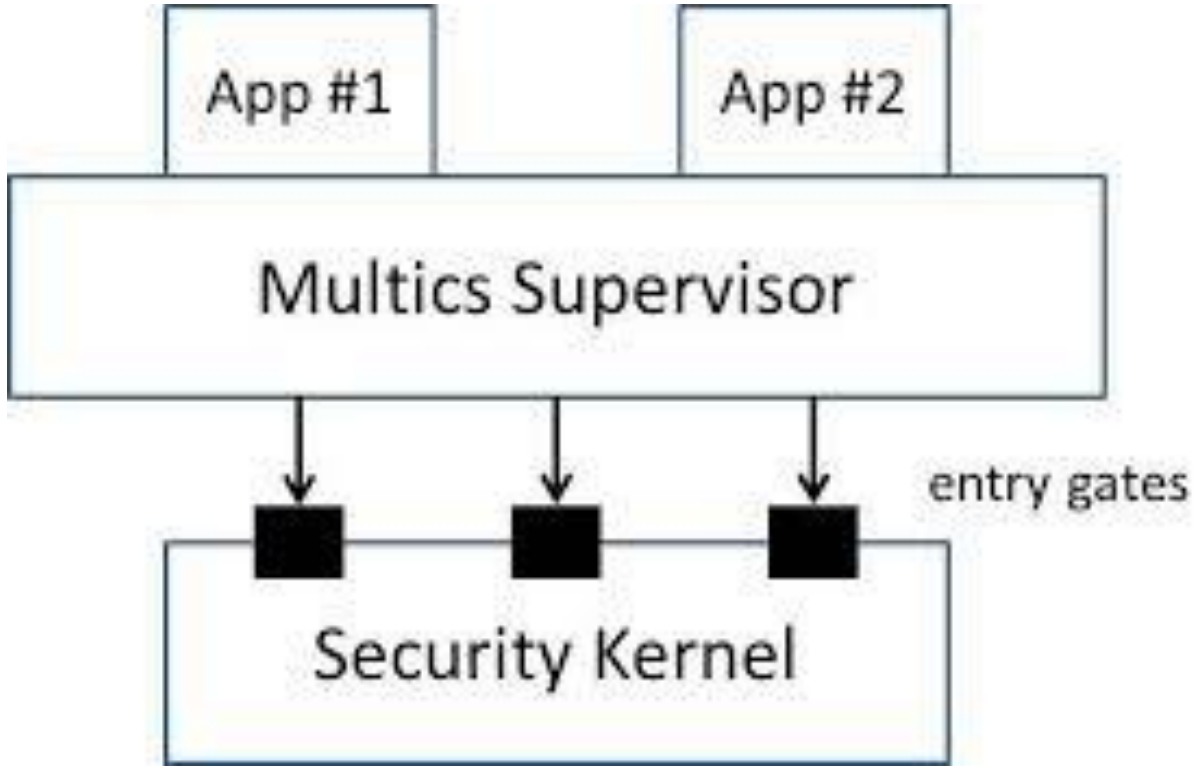
Policy Vs Mechanisms

- Access Control Policy Models:
  - pre-established rules:
    - All users should be treated equally.
    - Preferred users should be treated better.
  - Discretionary Access Control (DAC)
  - Mandatory Access Control (MAC)

- Mechanisms: tools to implement policies
  - Access control matrices
  - Access control list
  - Capability

Reference Monitor 1/2

- The Reference Monitor was introduced by James Anderson et al. in 1972 in the *Computer Security Technology Planning Study*.



- It provides an abstract model of the necessary properties that must be achieved by any system claiming to enforce access controls.

Reference Monitor 2/2

The three properties of Reference Monitor are:

- The access control mechanism is always invoked:
  - The access control cannot by bypassed.

- The access control mechanism is tamperproof
  - It is impossible for an attacker to break into the access mechanism.

- It must be small enough to be subject to analysis and tests
  - Its completeness can be assured.

Design Vocabulary

- Subject/Object:
  - A *subject* is any entity that requests access to an *object*.
  - The Subject/Object inherits from the same terms as used in grammar.
  - In System, all verbs are transitive: subject implies object. In other words, there is no subject without an object.

- Principle:
  - Principles are the keys we use in access control.
  - Principles are what subjects resolve to.
  - A subset of *subject* that is represented by a unique identifier.

Design Principles 1/2

- Complete Mediation:
  - Every access to every object must be checked.
- Economy of mechanism:
  - keep the designs as simple and small as possible.
- Fail-safe defaults:
  - Access decisions are based on permission rather than exclusion.
- Open Design:
  - The design should not be secret.

Design Principles 2/2

- Separation of Privilege:
  - Breaking up a single privilege into multiple independent components.

- Least privilege:
  - Every program and every user of the system should operate using the least set of privileges necessary to complete the job.

- Psychological acceptability:
  - Therefore, users happily apply the protection mechanisms correctly.

Part II

The UNIX DAC
(Discretionary Access Control)

DAC

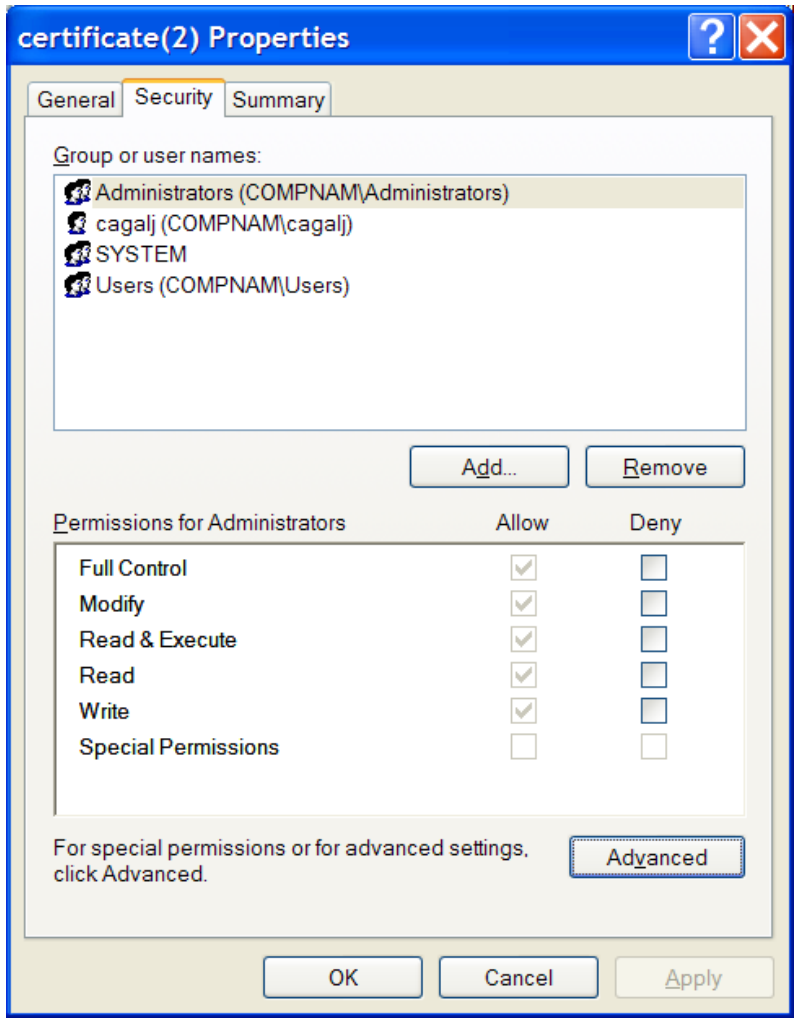
- Discretionary / Discrétionnaire:
  - Cambridge: decided by officials and not fixed by rules.
  - Larousse: Se dit du pouvoir de l'Administration lorsque cette dernière est libre de porter une appréciation sur l'utilité et l'opportunité d'une décision à prendre.

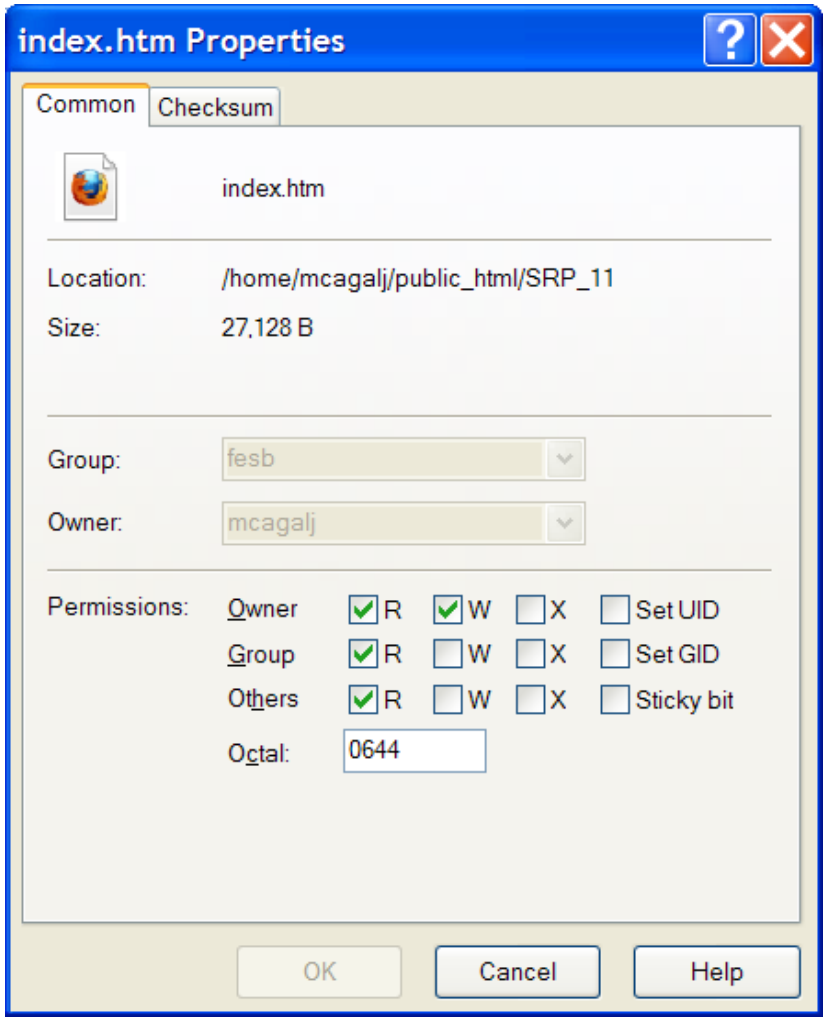
- In Computer Systems:
  - An individual user can set an access control mechanism to allow or deny access to an object.

- It relies on the object owner to control access.
- DAC is widely implemented in most operating systems.

Examples: Unix and Windows

Windows





Unix (WinSCP view)

Principles

- The Principles are **users** and **groups**.
- Users have username and **user identifier (UID)**.
- Groups have group name, **group identifier (GID)**.

- UID and GID are usually 16-bit numbers:
  - 0: root.
  - 1 – 99: predefined accounts.
  - 100 – 999: reserved for administrative and system accounts/groups.

- Both names and identifiers are permanent:
  - UID values often differ from system to system.

Superuser

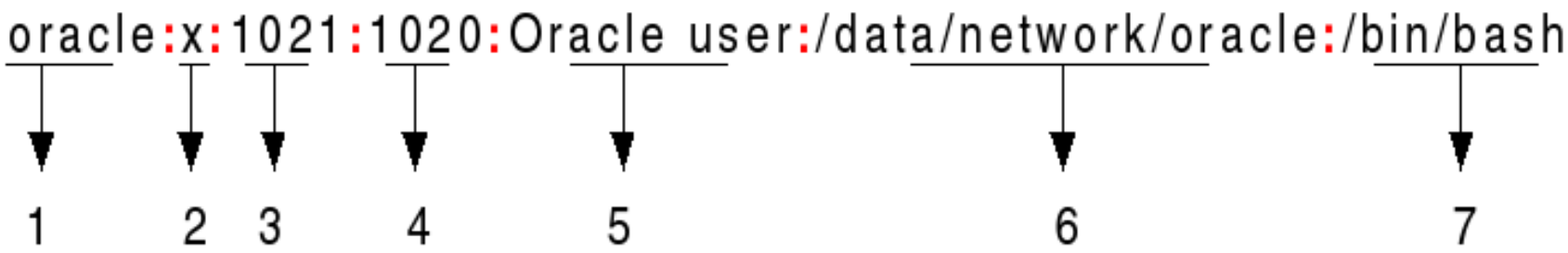
- The superuser is a special privileged principal with **UID zero** and usually the user name **root**.

- There are few restrictions on the superuser:
  - All security checks are not turned off for the superuser.
  - The superuser can become any other user.

- Examples:
  - Execute non-executable script.
  - The superuser cannot get passwords in plaintext, but can reset them.

User Accounts 1/2

- User accounts are stored in ***/etc/passwd***.
- User account format:
  - Username:password:UID:GID:name:homedir:shell



User Accounts 2/2

1. **Username**: It is used when user logs in. It should be between 1 and 32 characters in length.
2. **Password**: An x character indicates that the password is stored in ***/etc/shadow*** file.
3. **User ID (UID)**: Each user must be assigned a user ID (UID).
4. **Group ID (GID)**: The primary group ID (stored in /etc/group file)
5. **User ID Info**: The comment field. It allow you to add extra information about the users such as user's full name, phone number etc.
6. **Home directory**: The absolute path to the directory the user will be in when they log in. If this directory does not exist, then users directory becomes /.
7. **Command/shell**: The absolute path of a command or shell (/bin/bash). Typically, this is a shell. Please note that it does not have to be a shell.

The Shadow File

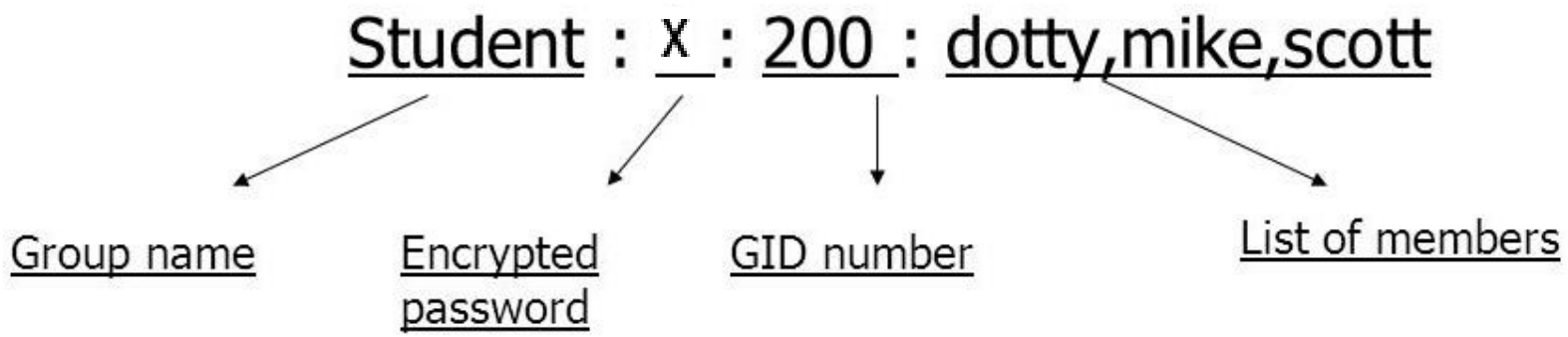
- Most modern Linux distributions use the **/etc/shadow** file to store encrypted password data.
- The **/etc/shadow** file is typically not readable by ordinary users.
  - The file is only accessible by root.

- Passwords are stored using a **hash** (a one-way type of scrambling).
- Here is how an entry in the **/etc/shadow** file looks like:

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

Groups

- Users belong to one or more groups at the same time.
- The file ***/etc/group*** contains a list of all groups; entry format:
  - groupname:password:GID:list of users



- Every user belongs to a primary group; the group ID (GID) of the primary group is stored in /etc/passwd.
- Use the **`groups`** command to see your groups

Group Passwords

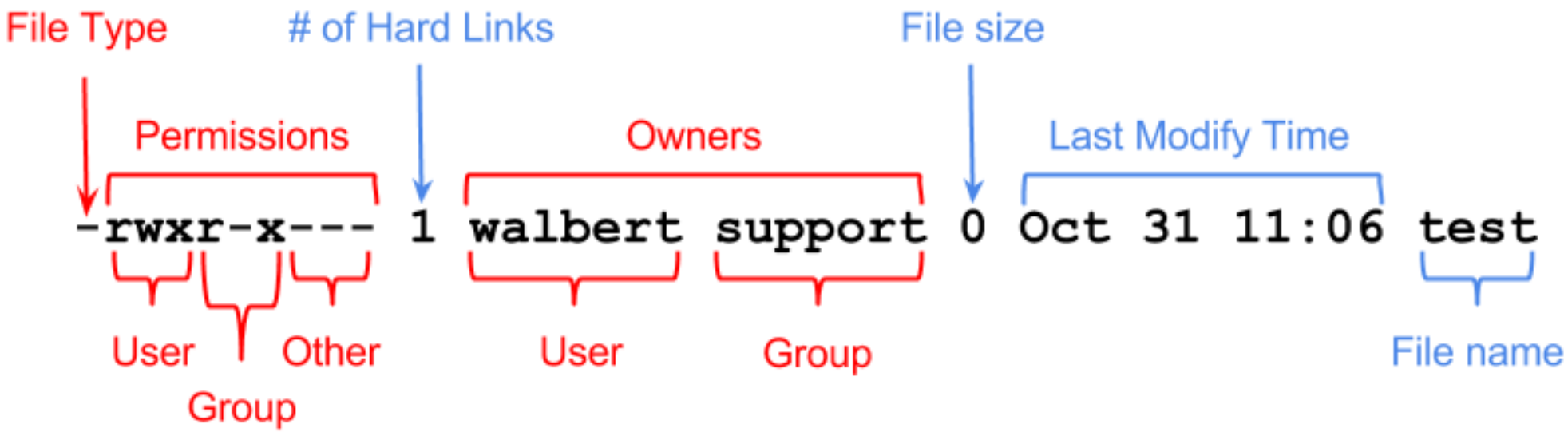
- Every group can have administrators, members and a password.
  - This can be managed with ***gpasswd*** (group passwords are in /etc/gshadow)

- Group passwords are an inherent security problem, since more than one person is permitted to know the password.
  - However, they can be useful in some contexts.

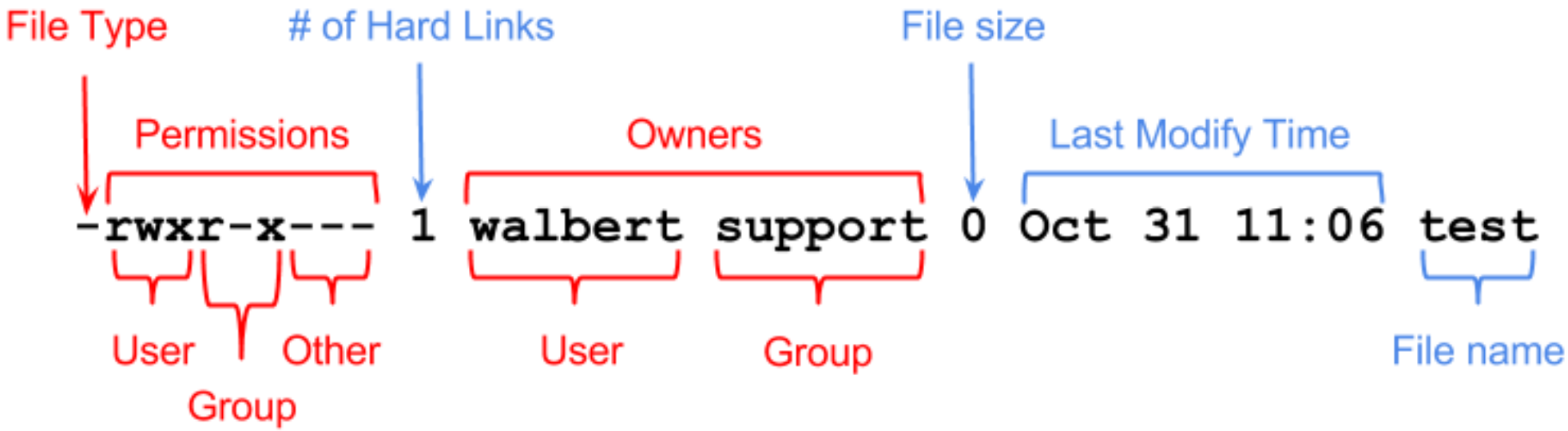
- The ***newgrp*** command
  - It is used to change the current real group ID (GID) to the specified group.
  - If the user is not root, they will be prompted for a group password if the user is not listed as a group member.
  - If there is NO group password set, and the user is not listed as a member of the group, the user will be denied access after asking them for a password.

Information about Files 1/2

- File type:
  - - file
  - d directory
  - b block device file
  - s socket
  - i symbolic link
  - p FIFO pipe
- File permissions: nine characters



Information about Files 2/2



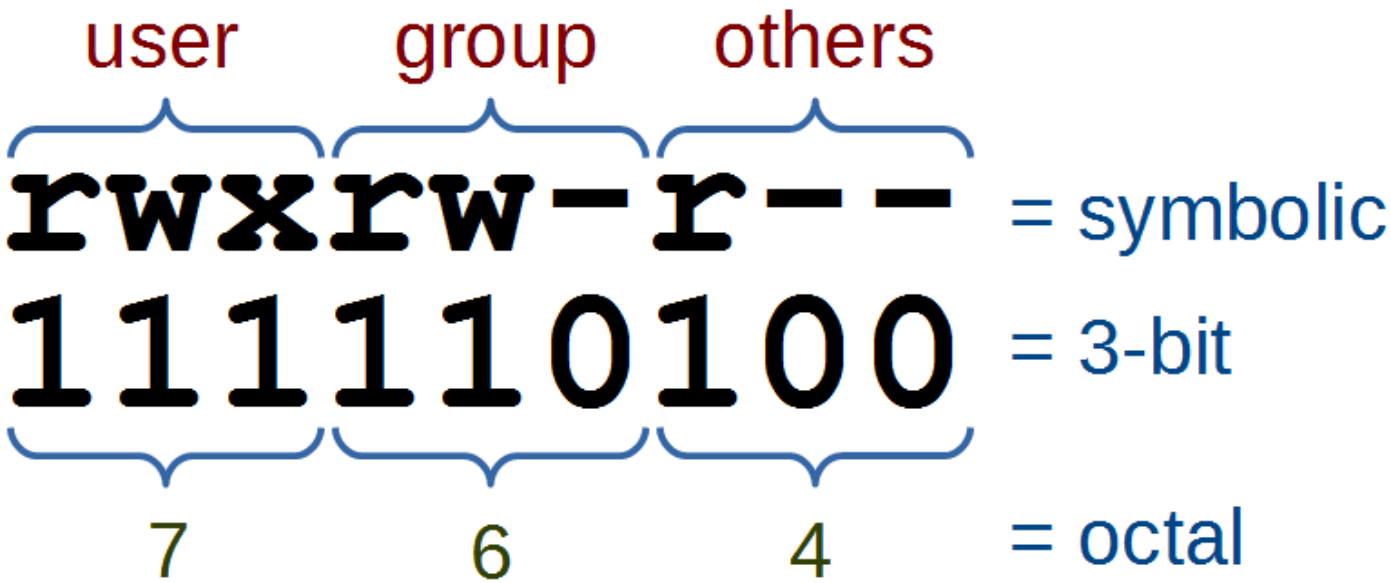
- Owners and root can change permissions.
- Only root can change the file owner and group.
- Owners can change the file group to one of its own group.

Permissions bits

- Permission bits are grouped in three triplets that define **read**, **write**, and **execute** for **owner**, **group** and **other**.
- Users:
  - Owner
  - Group
  - Other
- Operations:
  - Read: r
  - Write: w
  - Execute: x
- Permissions semantics are different between files and directories.

Octal Representation

- File permissions can also be specified as octal numbers.



Permissions Semantics

- Read:
  - File: to view the contents of the file.
  - Directory: to find which files are in the directory, e.g. for executing ls.

- Write:
  - File: to modify and delete the contents of the file.
  - Directory: to add and delete files.

- Execution:
  - File: to the run the file as a program.
  - Directory: to open files and make it the current directory (cd).

Access Control Decision

- Access control uses the effective UID/GID:
  - If the subject's UID owns the file, the permission bits for owner decide whether access is granted
  - If the subject's UID does not own the file but ones of its groups' GIDs does, the permission bits for group decide whether access is granted
  - If the subject's UID and no GID do not own the file, the permission bits for other (also called world) decide whether access is granted

- Note that the owner can always change the permissions.

Unix Access Control -- Discussions

- Unix permissions have been standardized by IEEE as part of the POSIX standards.
  - Fairly universal across Unix systems.
- Advantages:
  - Relatively simple and widely understood.
  - Relatively easy to check the protection state.
- Limitations:
  - Files have only one owner and one group.
  - Superuser needed for maintaining groups.
  - Complex policies, e.g. access several groups, are impractical to implement.
  - All access rights (e.g. shutdown) must be mapped to file access.

Umask

- Consider default behavior of file and directory creation
  - 666 & 777 respectively
  - To change this default behavior – use **umask**

- Defines the permissions to be masked while object is created
- Examples: umask 002
  - File creation: (666 – 002 = 664) = rw-rw-r--
  - Directory creation: (777 – 002 = 775) = rwxrwxr-x

Special Permissions

- Setgid: stands for SET Group ID.
  - Numerical value: 2000
  - Represented by an s in place of x, or S in place of - in the group permission.
  - Assigned to directories, makes all created files and directories belong to the group owning the setgid directory, instead of the GID.

- Sticky bit: meaningful when assigned to directories.
  - Numerical value: 1000
  - Represented by an t in place of x, or T in place of - in the other permission.
  - Prevents users from deleting each other’s file in a shared writeable directory.
  - Example: /tmp where any user can store files, but only owner of file has rights to modify or delete the file.

Extended ACL for Finer-Grain Control

- Extended ACLs provide:
  - Beyond simple user/group/other ownership.
  - Contain any number of named user & groups.
  - Contain mask entry.

- Utility/Library Functions
  - **getfacl**: check the current state of ACL on file/directory.
  - **setfacl**: modify/add ACL to additional user or group.
  - **chacl**: changes the ACL of file or directory.

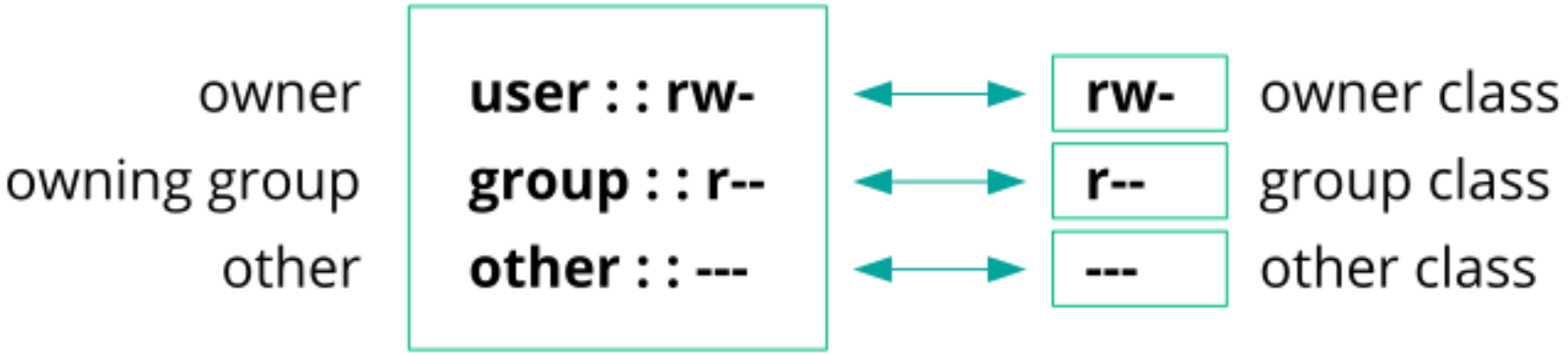
ACL Entry Types

Type	Text Form
Owner	user::rwx
Named user	user:name:rwx
Owning group	group::rwx
Named group	group:name:rwx
Mask	mask::rwx
Other	other::rwx

- The mask entry
  - Limits the permissions granted by named user, group and named group.
  - Permissions contained only in the mask or only in the actual entry are not effective.

Classes of ACLs

- Minimum ACL:
  - contains only the entries for the owner, the owning group and other, which correspond to the conventional permissions bits.
- Extended ACL
  - Contains several named user and group.



Example

user *heidi*, group *family* owns file with permissions:

```
user::rw-
user:skyler:rwx
group::rw-
group:child:r--
mask::rw-
other::r--
```

- *heidi* can read, write file (first line)
- *matt*, not in group *child*, can read file (last line)
- *skyler* can read, write file (second line masked by fifth line)
- *sage*, in group *family*, can read, write the file (third line masked by fifth line)
- *steven*, in group *child*, can read file (fourth line masked by fifth line)

Default ACL

- Directories can have a default ACL, which is a special kind of ACL defining the access permissions that objects in the directory inherit when they are created.

- Effects of a Default ACL
  - A file inherits the default ACL as its ACL.
  - A subdirectory inherits the default ACL of the parent directory both as its default ACL and as an ACL.

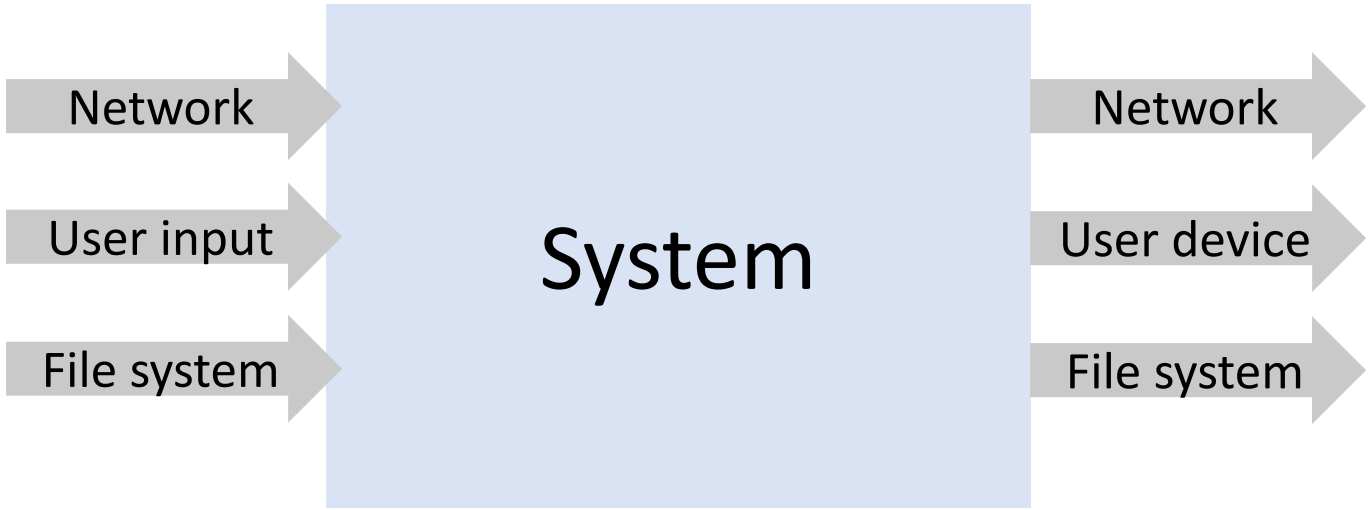
- The umask is disregarded in case default ACLs are used.

Principle of Least Privilege

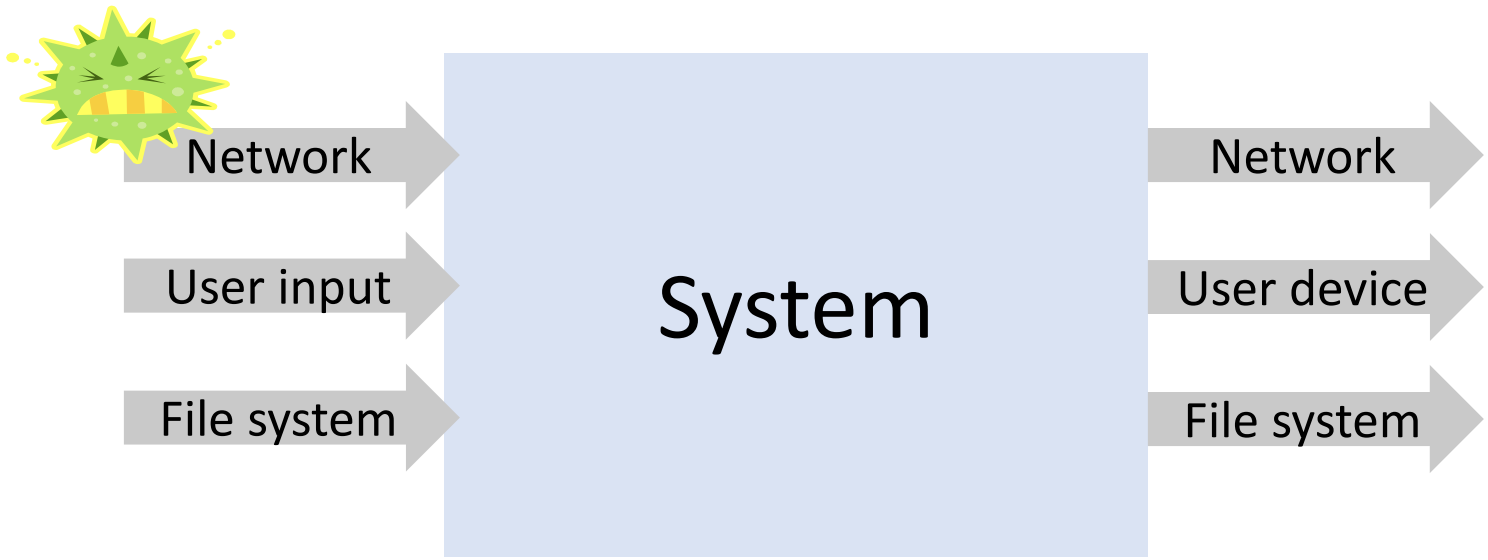
A subject should be given only those privileges that it needs in order to complete its task.

- **Privilege**: ability to access or modify a resource.
- Each system component or process should have the least authority necessary to perform its duties.
- This helps reduce the "attack surface" of the computer by eliminating unnecessary privileges that can result in security exploits.

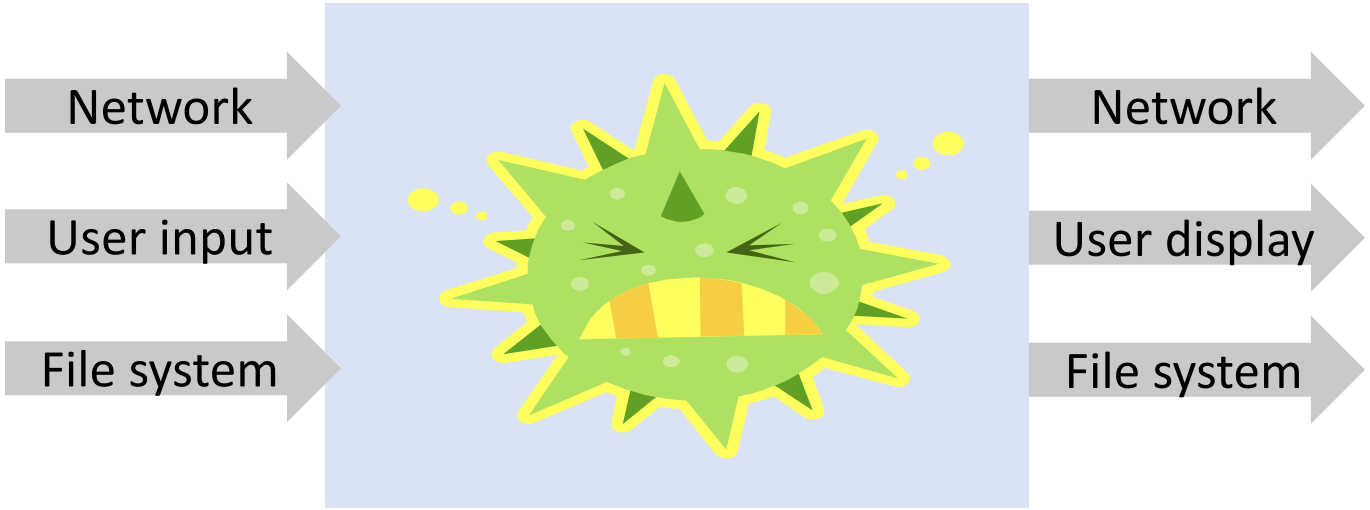
Monolithic Design



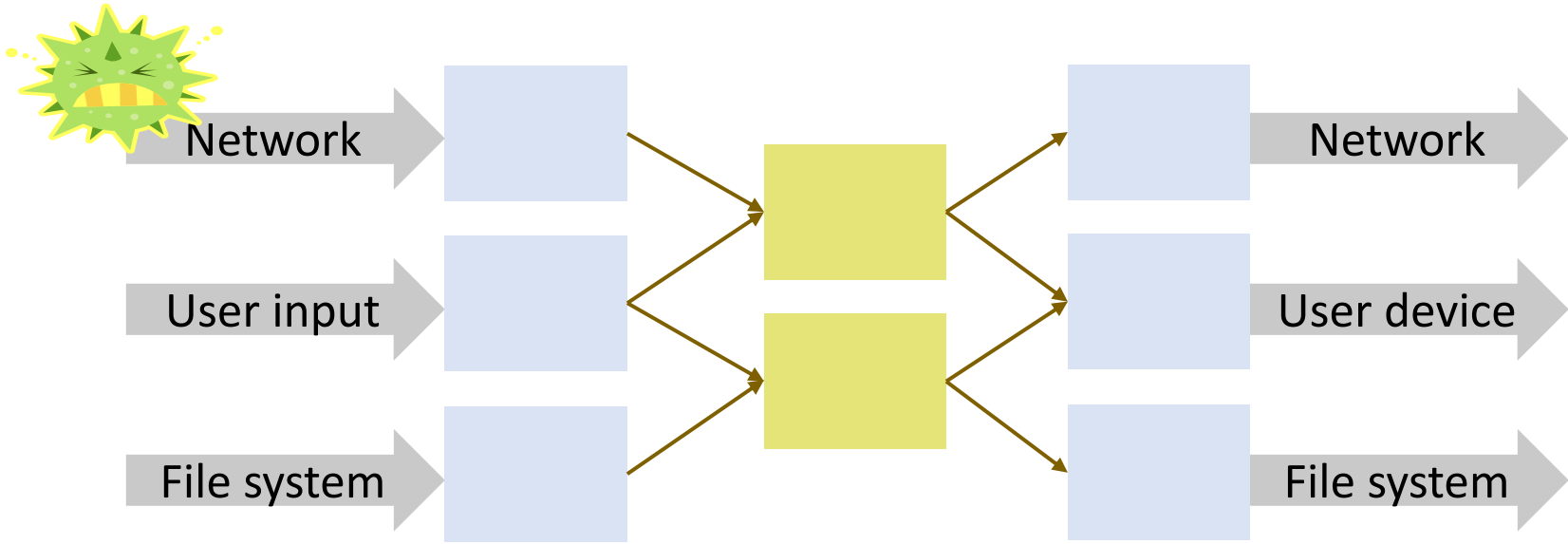
Monolithic Design



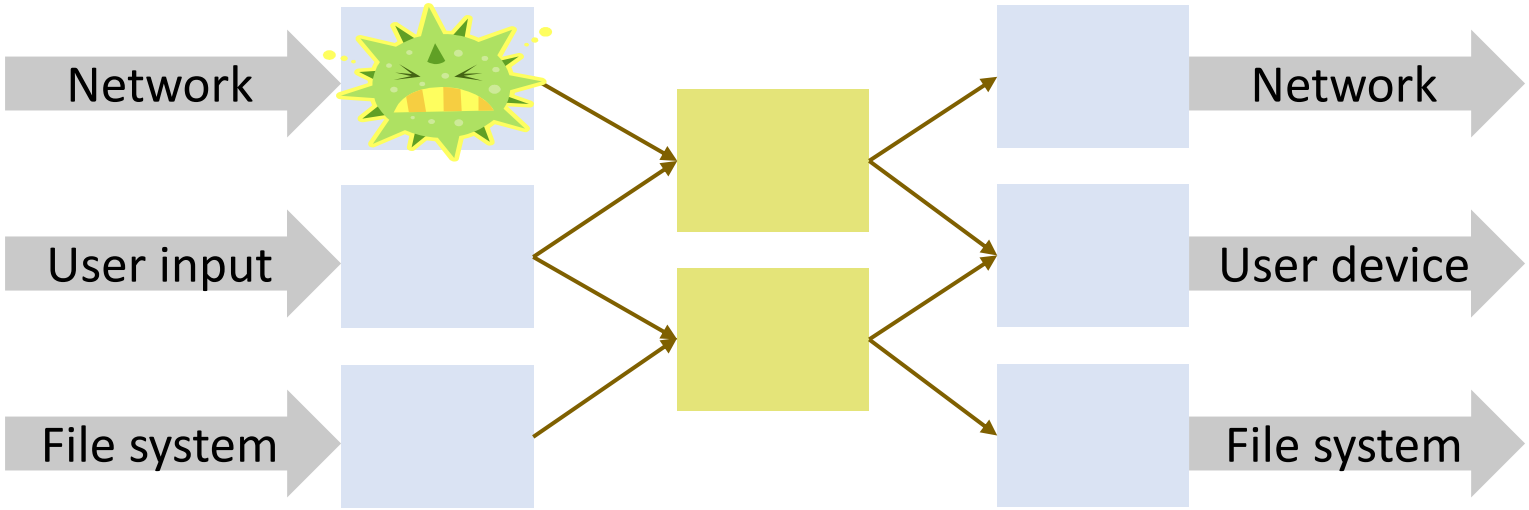
Monolithic Design



Component Design



Component Design



The Need of Superuser (The Unix Case)

- Users need to change their passwords.
  - Passwords are stored in the **/etc/shadow** file.
  - The permission of the file rw-r----- and its owned by root.

- Therefore, users need to be **root** to change their passwords.
  - Root, the superuser, has unlimited power.
  - Consequently, any user should have unlimited power to change their passwords.

The Need of Superuser (The Unix Case)

- Exercise: Noticing that you can change your password without root privilege in Linux, how does Unix solve this problem?

A Deputy to the Rescue

- A Deputy is a program that can be called by any user and performs a particular task requiring an escalated privilege.

- A user should be able to run programs on behalf of a different user.
  - Technical problem: when a user runs a program, the program gets its privilege by the UID/GID associated to the user.
  - Security impact: users must not be able to run arbitrary programs performing arbitrary tasks on behalf of any user, especially the root.

- Proposed Solution:
  - The root defines some programs runnable by other users.
  - These programs have the root privilege when they are called.

SUID and SGID Programs

- SUID programs run with the effective UID of the owner of the executable file.
- For a SUID program, the execute permission of the owner is given as **s** instead of **x**, and **S** instead of **-**.

- SGID programs run with the effective GID of the owner of the executable file.
- For a SGID program, the execute permission of the group is given **s** instead of **x**, and **S** instead of **-**.

SUID to root

- When root owns an executable file and the SUID bit is set, the process will get superuser status during execution.
- Important SUID programs:
  - /bin/passwd: change password.
  - /bin/su: change UID program.

- SUID programs need to be written very carefully so that their privileges cannot be misused and they only do what is intended.