Chapter 4: Access Control

Xianghang Mi

中国科学技术大学
University of Science and Technology of China

Last Time: Authentication

- Authentication Concepts
  - Verifying the assertion of an identity
- Authentication Scenarios: The Cyberspace, Cyber-Physical Systems, etc
- Authentication Primitives
  - Password, Token, Biometrics
- Authentication Protocols
  - TLS
  - FIDO
  - OAuth

Outline

- Concepts
- Scenarios
- Policies
- Applications

Outline

- Concepts
- Scenarios
- Policies
- Applications

Access Control

- Access control: ensures that all direct accesses to object are authorized
- Protects against accidental and malicious threats by regulating the reading, writing and execution of data and programs
- Need:
  - User identification and authentication
  - Information specifying the access rights is protected form modification

Definitions of Access Control

- RFC 4949 defines access control as: “a process by which use of system resources is regulated according to a security policy and is permitted only by authorized entities (users, programs, processes, or other systems) according to that policy”

- NISTIR 7298 defines access control as: “the process of granting or denying specific requests to: (1) obtain and use information and related information processing services; and (2) enter specific physical facilities”

Concepts

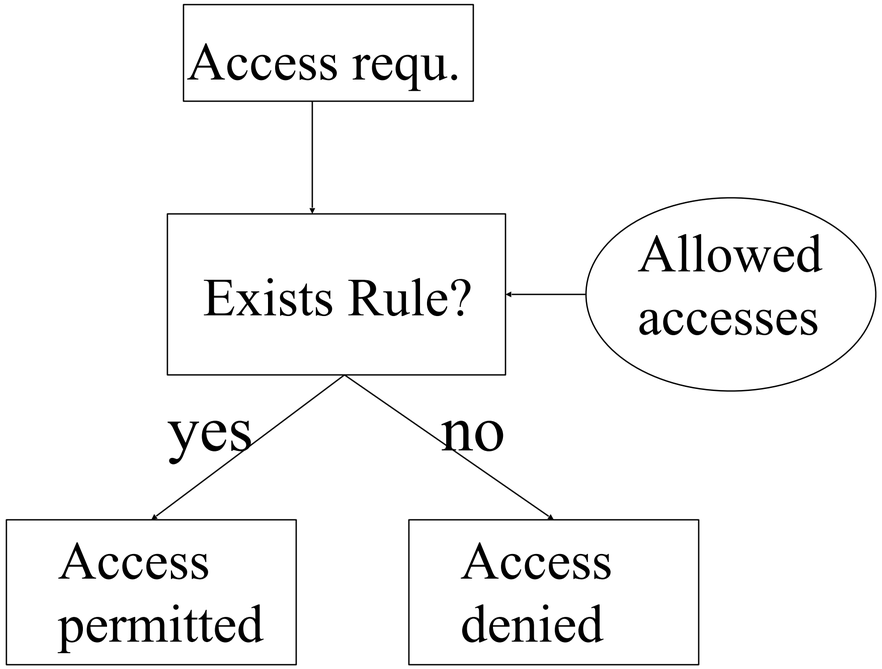
- **Protection objects**: system resources for which protection is desirable
  - Memory, file, directory, hardware resource, software resources, etc.
- **Subjects**: active entities requesting accesses to resources
  - User, owner, program, etc.
- **Access mode**: type of access
  - Read, write, execute

Requirements

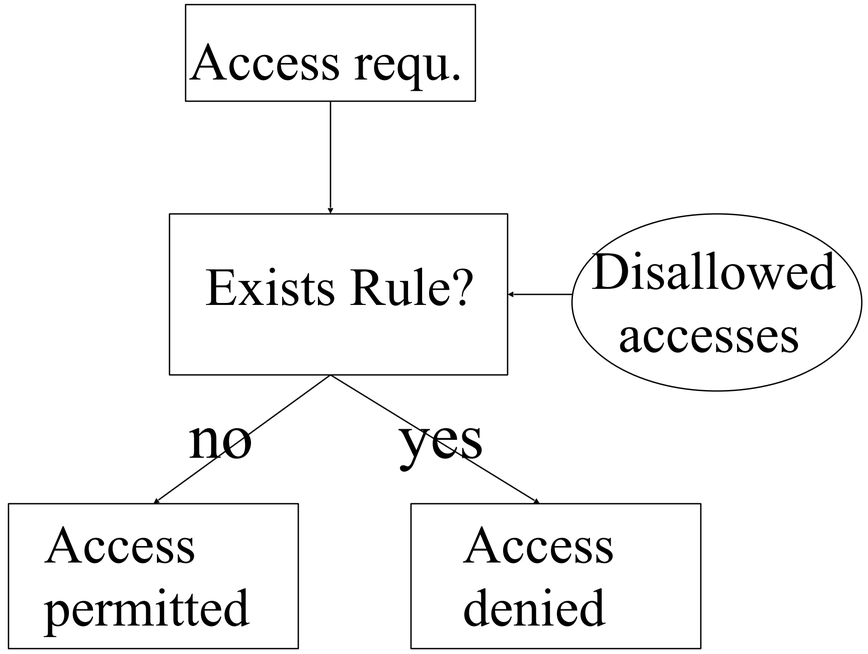
- Cannot be bypassed
- Enforce least-privilege and need-to-know restrictions
- Enforce organizational policy

Closed v.s. Open Systems

Closed system
(minimum privilege)



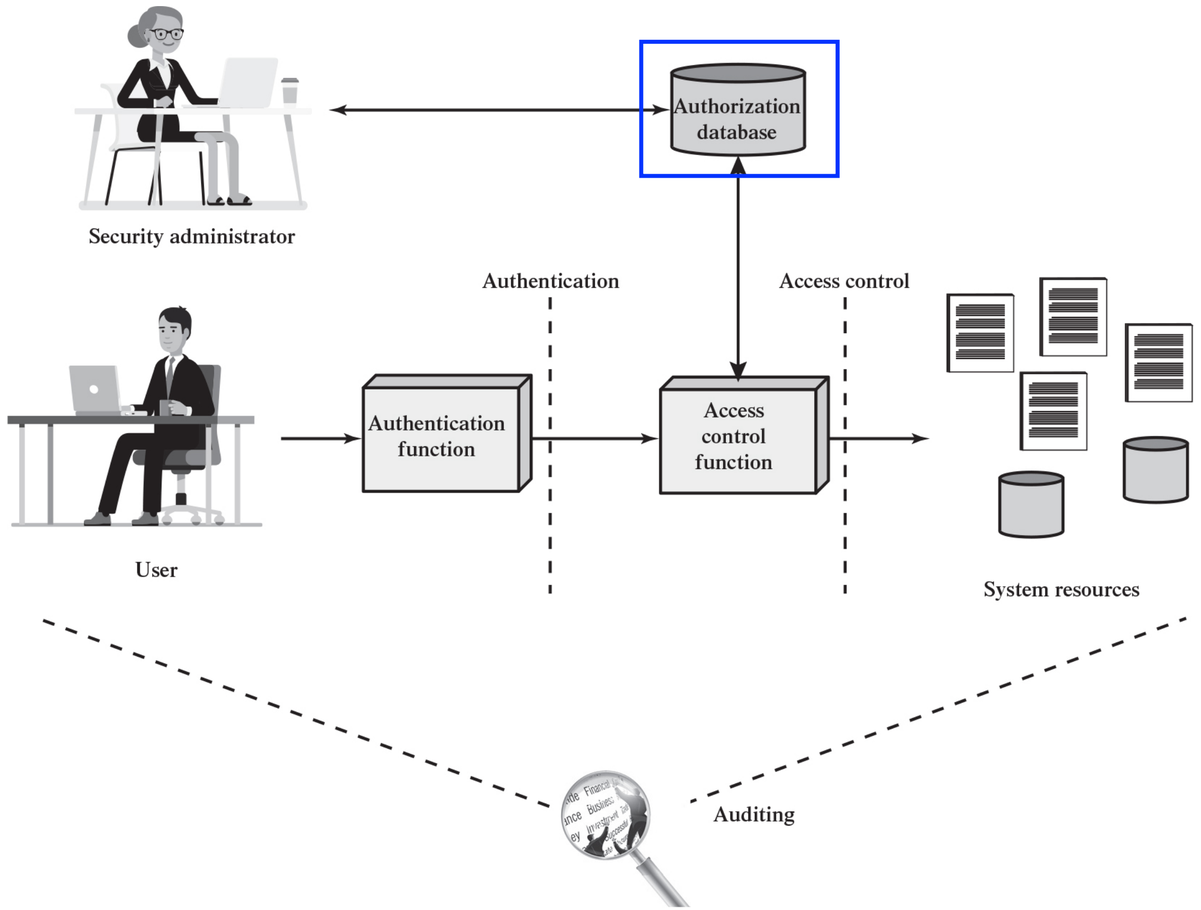
Open System
(maximum privilege)



Components

- Access control components:
  - Access control policy: specifies the authorized accesses of a system
  - Access control mechanism: implements and enforces the policy
- Separation of components allows to:
  - Define access requirements independently from implementation
  - Compare different policies
  - Implement mechanisms that can enforce a wide range of policies

Relationship Among Access Control and Other Security Functions



Source: Based on [S A N D94].

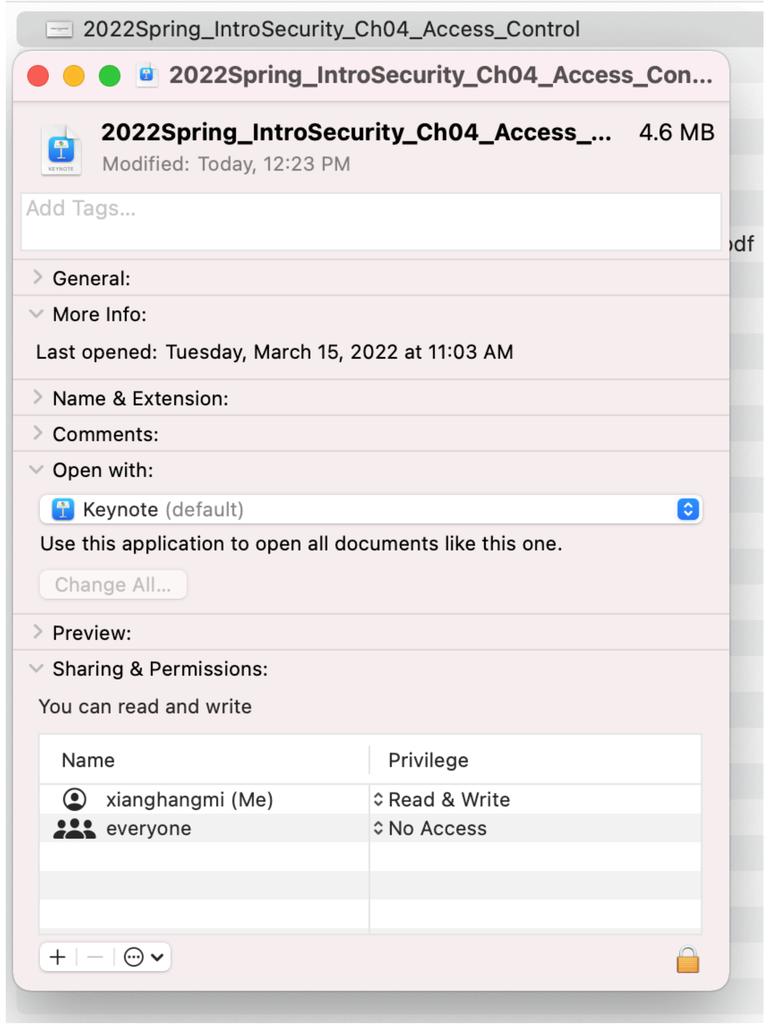
Outline

- Concepts
- Scenarios
- Polices
- Applications

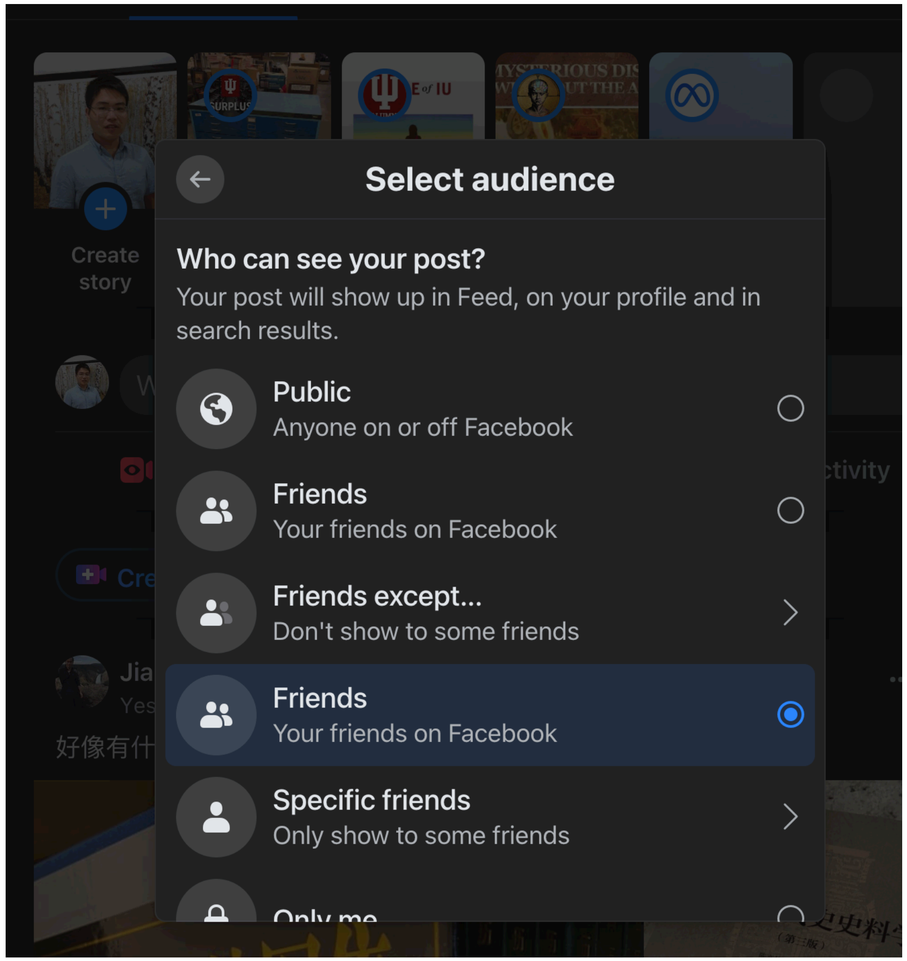
Physical Access Control

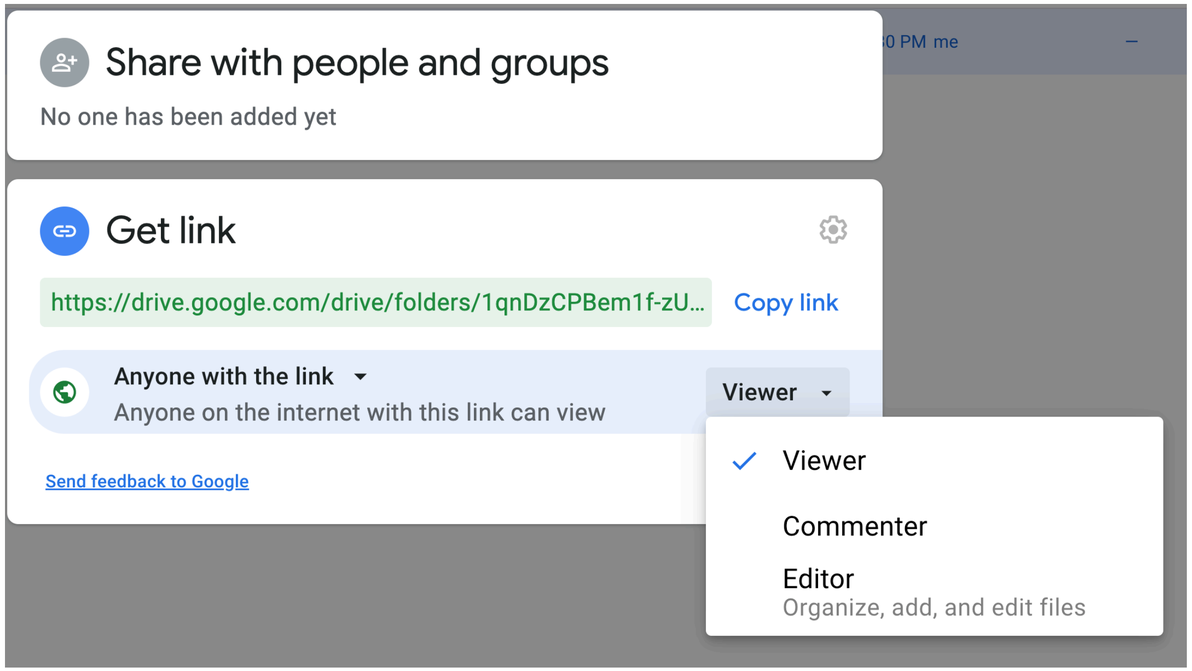
File System Access Control

```
onionlab@onionlab0:~$ ls -all /data/
total 48
drwxrwxr-t  9 root     data      4096 Feb 25 14:26 .
drwxr-xr-x 23 root     root      4096 Dec 23 15:52 ..
drwxrwxr-x  3 onionlab onionlab  4096 Feb 13 21:28 biometrics
drwx------  2 root     root     16384 Nov  9 22:43 lost+found
drwxrwxr-x  5 onionlab onionlab  4096 Dec 12 11:41 polluted_websites
drwxrwxr-x  3 onionlab onionlab  4096 Jan  9 21:04 rpaas
drwxrwxr-x  3 onionlab onionlab  4096 Dec  4 10:55 rpaas_china
drwxrwxr-x  2 onionlab onionlab  4096 Feb 27 16:51 ssh_keys
drwxr-xr-x 11 onionlab onionlab  4096 Dec 10 11:11 time_machine
```



Sharing Access Control





Outline

- Concepts
- Scenarios
- Policies
- Applications

How to Define Access Control Policies?

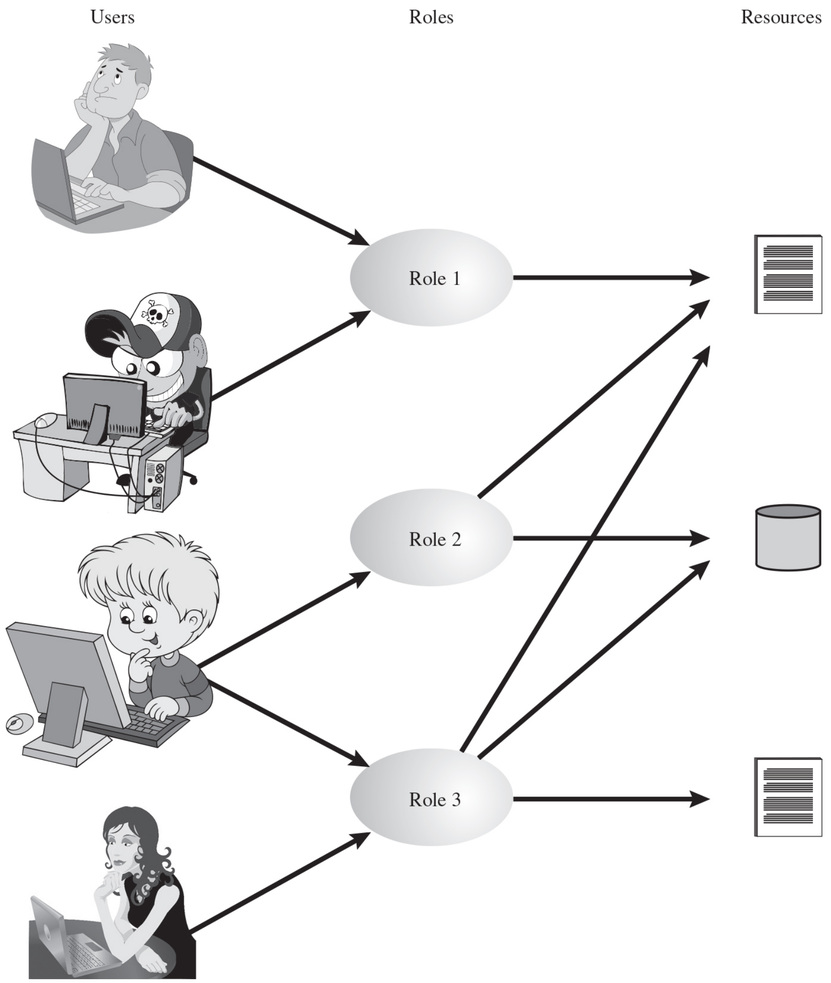
- Discretionary access control (DAC)
  - Controls access based on the identity of the requestor and on access rules (authorizations) stating what requestors are (or are not) allowed to do
- Mandatory access control (MAC)
  - An access control policy that is **uniformly enforced across all subjects and objects** within the boundary of an information system.
- Role-based access control (RBAC)
  - Controls access based on the roles that users have within the system and on rules stating what accesses are allowed to users in given roles
- Attribute-based access control (ABAC)
  - Controls access based on attributes of the user, the resource to be accessed, and current environmental conditions

In real-world implementation, access control policies are defined in a hybrid mode

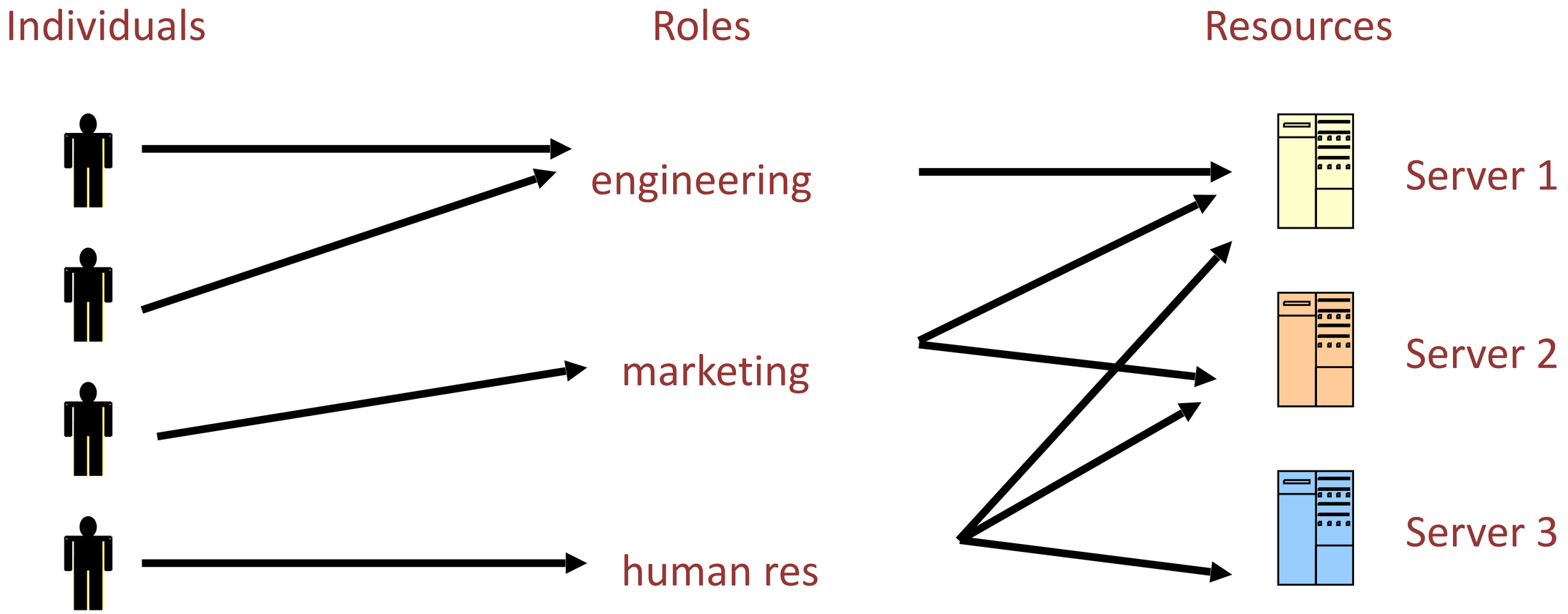
Discretionary Access Control (DAC)

- Scheme in which an entity may be granted access rights that permit the entity, by its own violation, to enable another entity to access some resource
- Often provided using an access matrix
  - One dimension consists of identified subjects that may attempt data access to the resources
  - The other dimension lists the objects that may be accessed
  - Each entry in the matrix indicates the access rights of a particular subject for a particular object

Role-Based Access Control



Role-Based Access Control



Advantage: users change more frequently than roles

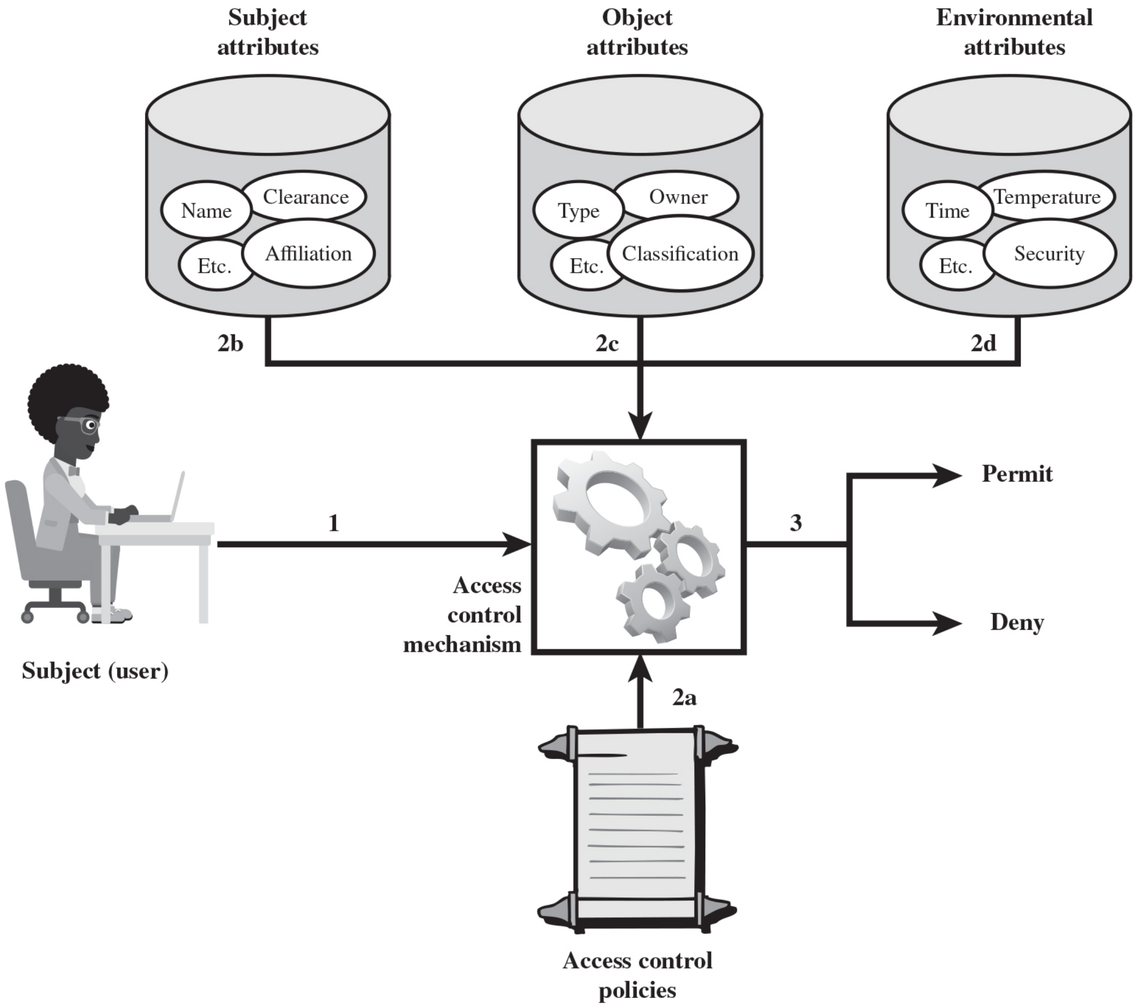
Attribute-Based Access Control (ABAC)

- Can define authorizations that express conditions on properties of both the resource and the subject
- Strength is its flexibility and expressive power
- Main obstacle to its adoption in real systems has been concerns about the performance impact of evaluating predicates on both resource and user properties for each access
- Web services have been pioneering technologies through the introduction of the eXtensible Access Control Markup Language (XACML)
- There is considerable interest in applying the model to cloud services

ABAC Model: Attributes

- Subject attributes
  - A subject is an active entity that causes information to flow among objects or changes the system state
  - Attributes define the identity and characteristics of the subject
- Object attributes
  - An object (or resource) is a passive information system-related entity containing or receiving information
  - Objects have attributes that can be leverages to make access control decisions
- Environment attributes
  - Describe the operational, technical, and even situational environment or context in which the information access occurs
  - These attributes have so far been largely ignored in most access control policies

ABAC Scenario



How to Materialize Access Control Policies?

Subjects, Objects, and Access Rights

- Subject
  - An entity capable of accessing objects
  - A set of attributes
  - A set of roles it belongs to
  - Three classes
    - Owner
    - Group
    - World
- Object
  - A resource to which access is controlled
  - Entity used to contain and/or receive information
- Access right
  - Describes the way in which a subject may access an object
  - Could include:
    - Read
    - Write
    - Execute
    - Delete
    - Create
    - Search

Access Control Matrix

Objects

Subjects

	File 1	File 2	File 3	…	File n
User 1	read	write	-	-	read
User 2	write	write	write	-	-
User 3	-	-	-	read	read
…					
User m	read	write	read	write	read

Access Control Matrix Representation of RBAC

	R_1	R_2	• • •	R_n
U_1	✖			
U_2	✖			
U_3		✖		✖
U_4				✖
U_5				✖
U_6				✖
•				
•				
•				
U_m	✖			

	OBJECTS								
ROLES	R_1	R_2	R_n	F_1	F_2	P_1	P_2	D_1	D_2
R_1	control	owner	owner control	read *	read owner	wakeup	wakeup	seek	owner
R_2		control		write *	execute			owner	seek *
•									
•									
•									
R_n			control		write	stop			

How To Implement the Access Control Matrix?

- Access control list (ACL)
  - Store column of matrix with the resource
- Capability
  - User holds a "ticket" for each resource
  - Two variations
    - store row of matrix with user, under OS control
    - unforgeable ticket in user space

	File 1	File 2	…
User 1	read	write	-
User 2	write	write	-
User 3	-	-	read
…			
User m	Read	write	write

Access control lists are widely used, often with groups

Some aspects of capability concept are used in many systems, e.g., access token in OAuth

ACL vs Capabilities

- Access control list
  - Associate list with each object
  - Check user/group against list
  - Relies on authentication: need to know user
- Capabilities
  - Capability is unforgeable ticket
    - Random bit sequence (or managed by OS)
    - Can be passed from one process to another
  - Reference monitor checks ticket
    - Does not need to know identify of user/process

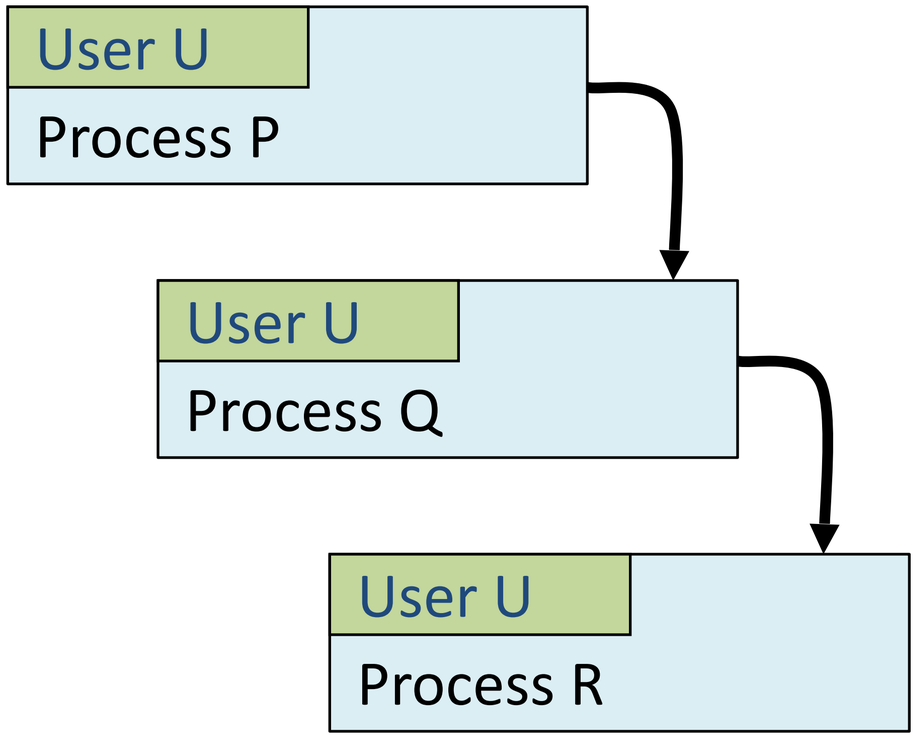
ACL vs Capabilities

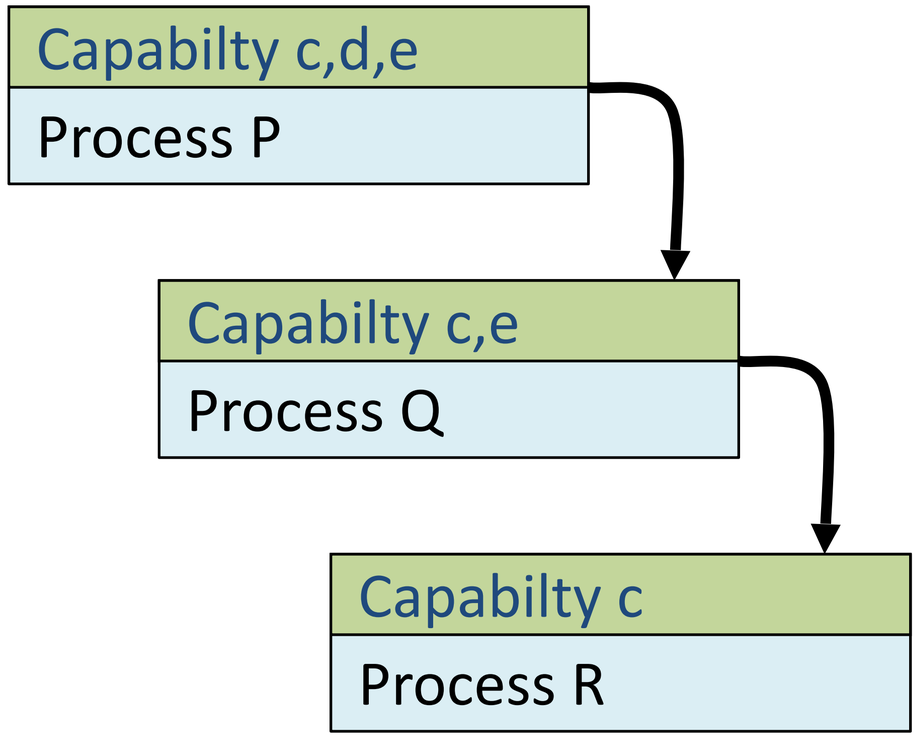
- Access control list
  - 高铁检票
  - 宴会邀请
- Capabilities
  - 景区门票
  - 地铁卡
  - 饭票粮票
  - OAuth access token

ACL vs Capabilities

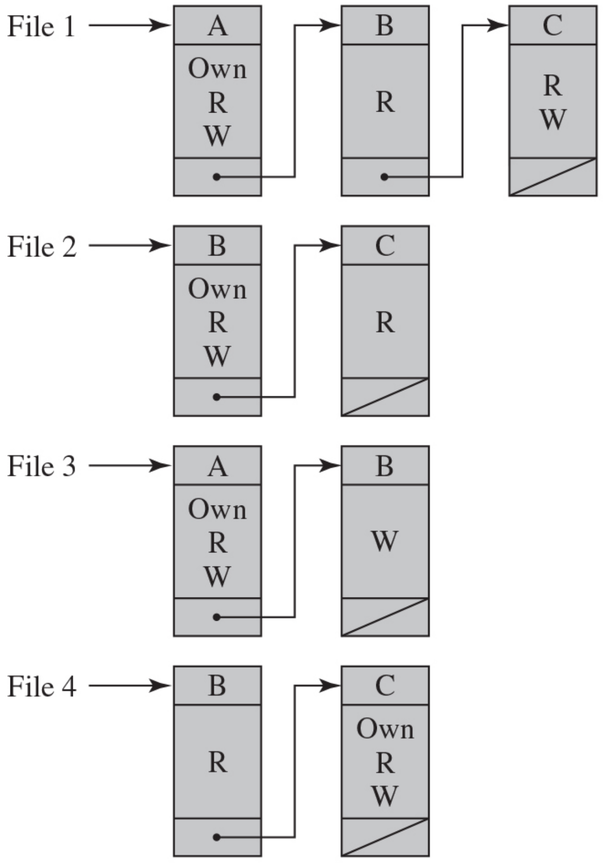
- Delegation
  - Cap: Process can pass capability at run time
  - ACL: Try to get owner to add permission to list?
    - More common: let other process act under current user
- Revocation
  - ACL: Remove user or group from list
  - Cap: Try to get capability back from process?
    - Possible in some systems if appropriate bookkeeping
      - OS knows which data is capability
      - If capability is used for multiple resources, have to revoke all or none …
    - Indirection: capability points to pointer to resource
      - If C → P → R, then revoke capability C by setting P=0

Process creation: ACL vs Capabilities

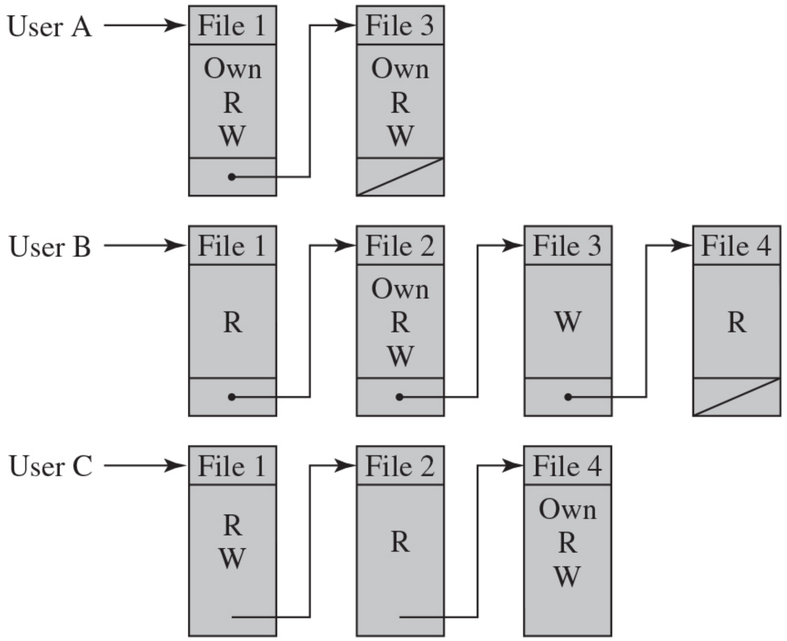




An Implementation of Access Control Matrix



(b) Access control lists for files of part (a)



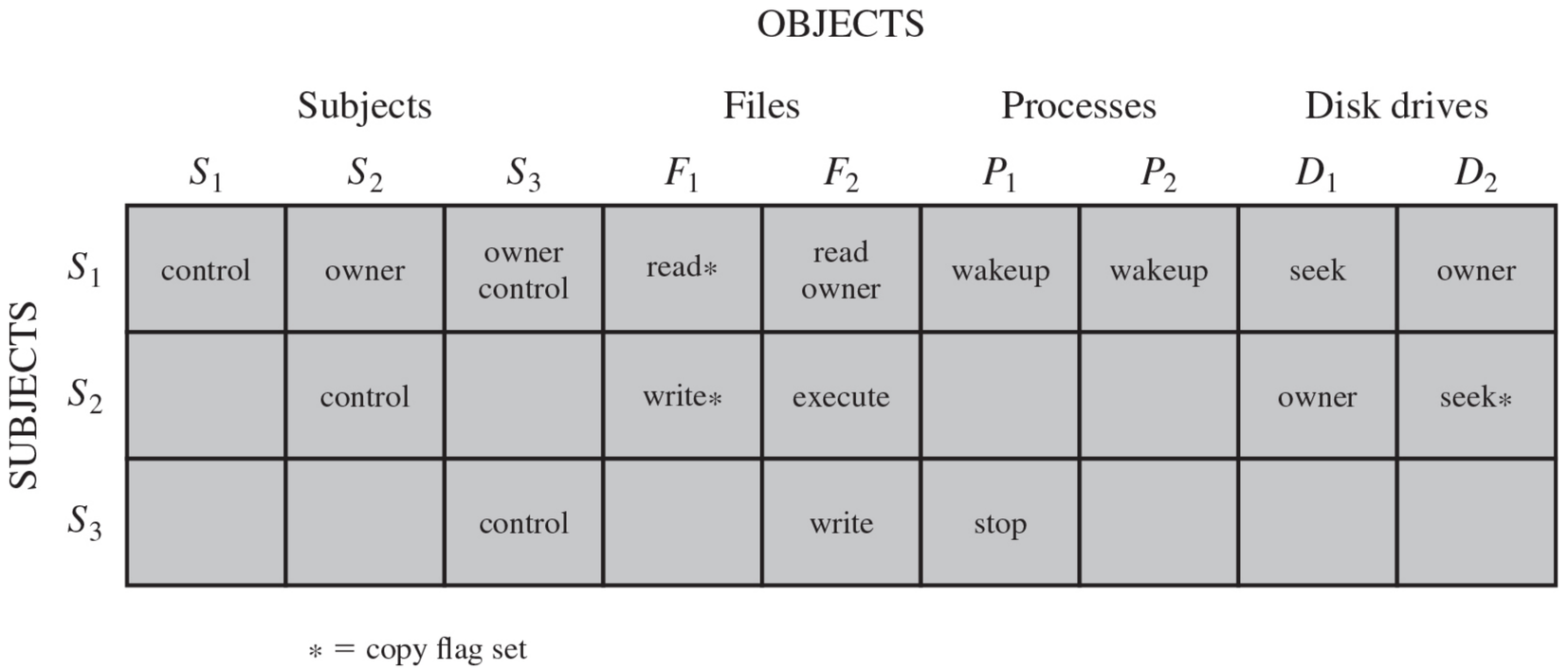
(c) Capability lists for files of part (a)

Access Control Matrix through a Table

Subject	Access Mode	Object
A	Own	File 1
A	Read	File 1
A	Write	File 1
A	Own	File 3
A	Read	File 3
A	Write	File 3
B	Read	File 1
B	Own	File 2
B	Read	File 2
B	Write	File 2
B	Write	File 3
B	Read	File 4

Subject	Access Mode	Object
C	Read	File 1
C	Write	File 1
C	Read	File 2
C	Own	File 4
C	Read	File 4
C	Write	File 4

Extended Access Control Matrix

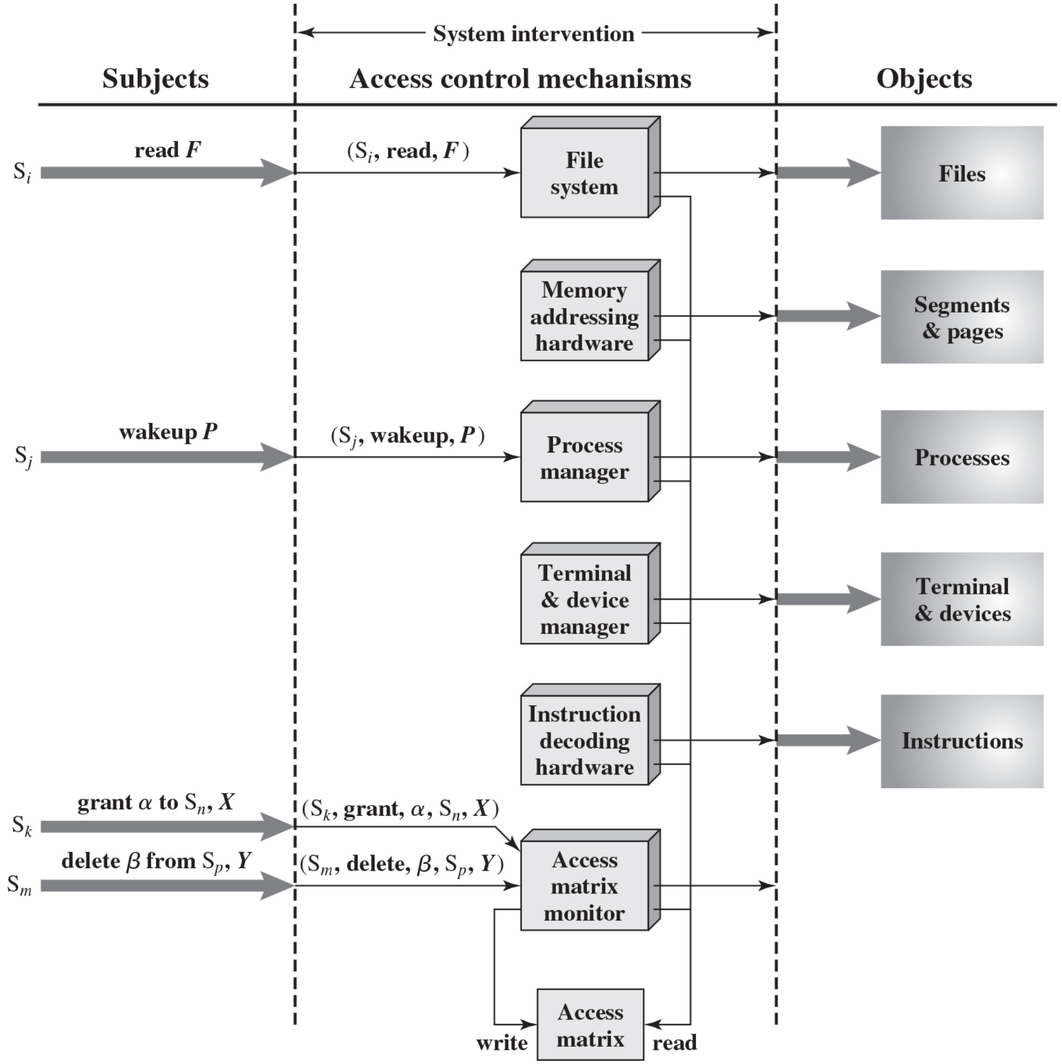


OBJECTS

SUBJECTS	Subjects			Files		Processes		Disk drives	
	S_1	S_2	S_3	F_1	F_2	P_1	P_2	D_1	D_2
S_1	control	owner	owner control	read*	read owner	wakeup	wakeup	seek	owner
S_2		control		write*	execute			owner	seek*
S_3			control		write	stop			

* = copy flag set

An Organization of the Access Control Function



Access control summary

- Access control involves reference monitor
  - Check permissions: ⟨user info, action⟩→ yes/no
  - Important: there should be no way to bypass this check
- Access control matrix
  - Two implementations: access control lists vs capabilities
  - Advantages and disadvantages of each
- Role-based access control
  - Use group as “user info”; use group hierarchies

Discussion?

- Access control matrix
  - What are the advantages of access control lists (ACL)?
  - What are the advantages of capabilities?
- Role-based access control
  - Why is this helpful?

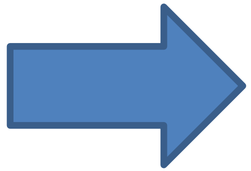
Outline

- Concepts
- Scenarios
- Policies
- Applications: Unix, Android, Browser

Unix

- What access control concepts are used?
  - Truncated access control list
  - A form of role-based access control

	File 1	File 2	…
User 1	read	write	-
User 2	write	write	-
User 3	-	-	read
…			
Role r	Read	write	write



	File 1	File 2	…
Owner	read	write	-
Group	write	write	-
Other	-	-	read

Unix access control

- Process has user id
  - Inherit from the creating process
  - Process can change id
    - Restricted set of options
  - Special “root” id
    - All access allowed
- File has access control list (ACL)
  - Grants permission to users
  - Three “roles”: owner, group, other

	File 1	File 2	…
Owner	read	write	-
Group	write	write	-
Other	-	-	read

Unix file access control list

- Each file has owner and group
- Permissions set by owner
  - Read, write, execute
  - Owner, group, other
  - Represented by vector of four octal values
- Only owner, root can change permissions
  - This privilege cannot be delegated or shared
- Setid bits – Discuss in a few slides

```
onionlab@onionlab0:~$ getfacl ./script.deb.sh
# file: script.deb.sh
# owner: onionlab
# group: onionlab
user::rwx
group::rwx
other::r-x
```

Process effective user id (EUID)

- Each process has three Ids (+ more under Linux)
  - Real user ID (RUID)
    - same as the user ID of parent (unless changed)
    - used to determine which user started the process
  - Effective user ID (EUID)
    - from set user ID bit on the file being executed, or sys call
      - /etc/passwd
    - determines the permissions for process
      - file access and port binding
  - Saved user ID (SUID)
    - So previous EUID can be restored
- Real group ID, effective group ID, used similarly

Process Operations and IDs

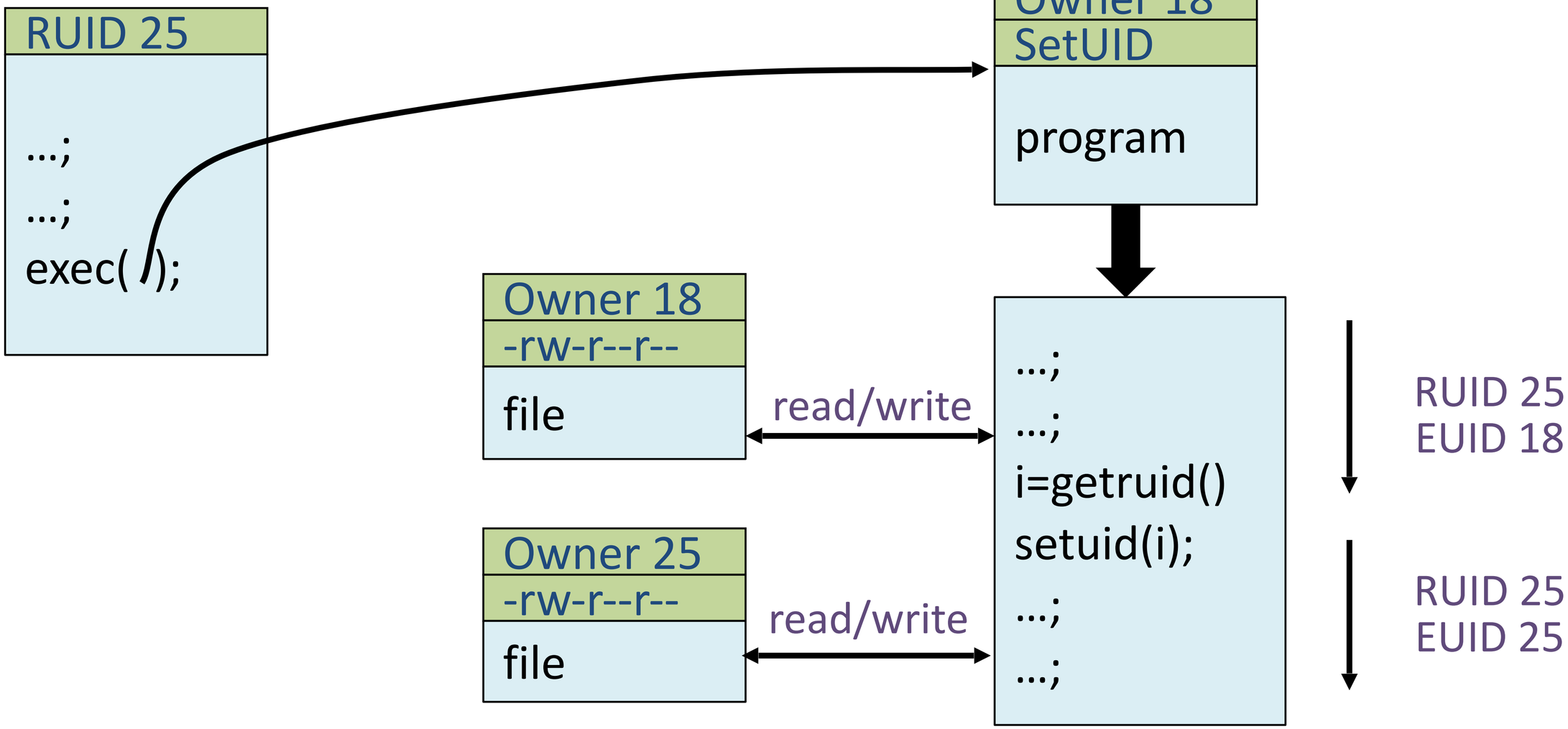
- Root
  - ID=0 for superuser root; can access any file
- Fork and Exec
  - Inherit three IDs, except exec of file with setuid bit
- Setuid system call
  - seteuid(newid) can set EUID to
    - Real ID or saved ID, regardless of current EUID
    - Any ID, if EUID is root
- Details are actually more complicated
  - Several different calls: setuid, seteuid, setreuid

Setid bits on executable Unix file

- Three setid bits
  - Setuid – set EUID of process to ID of file owner
  - Setgid – set EGID of process to GID of file
  - Sticky
    - Off: if user has write permission on directory, can rename or remove files, even if not owner
    - On: only file owner, directory owner, and root can rename or remove file in the directory

Example

```
onionlab@onionlab2022:~$ ls -all /usr/bin/passwd
-rwsr-xr-x 1 root root 59976 Jun 18  2021 /usr/bin/passwd
```



Unix access control summary

- Good things
  - Some protection for most users
  - Flexible enough to make practical systems possible
- Main limitation
  - Coarse-grained ACLs – user, group, other
  - Too tempting to use root privileges
  - No way to assume some root privileges without all

Weakness in unix isolation, privileges

- Network-facing Daemons
  - Root processes with network ports open to all remote parties, e.g., sshd, ftpd, sendmail, …
- Rootkits
  - System extension via dynamically loaded kernel modules
- Environment Variables
  - System variables such as LIBPATH that are shared state across applications. An attacker can change LIBPATH to load an attacker-provided file as a dynamic library

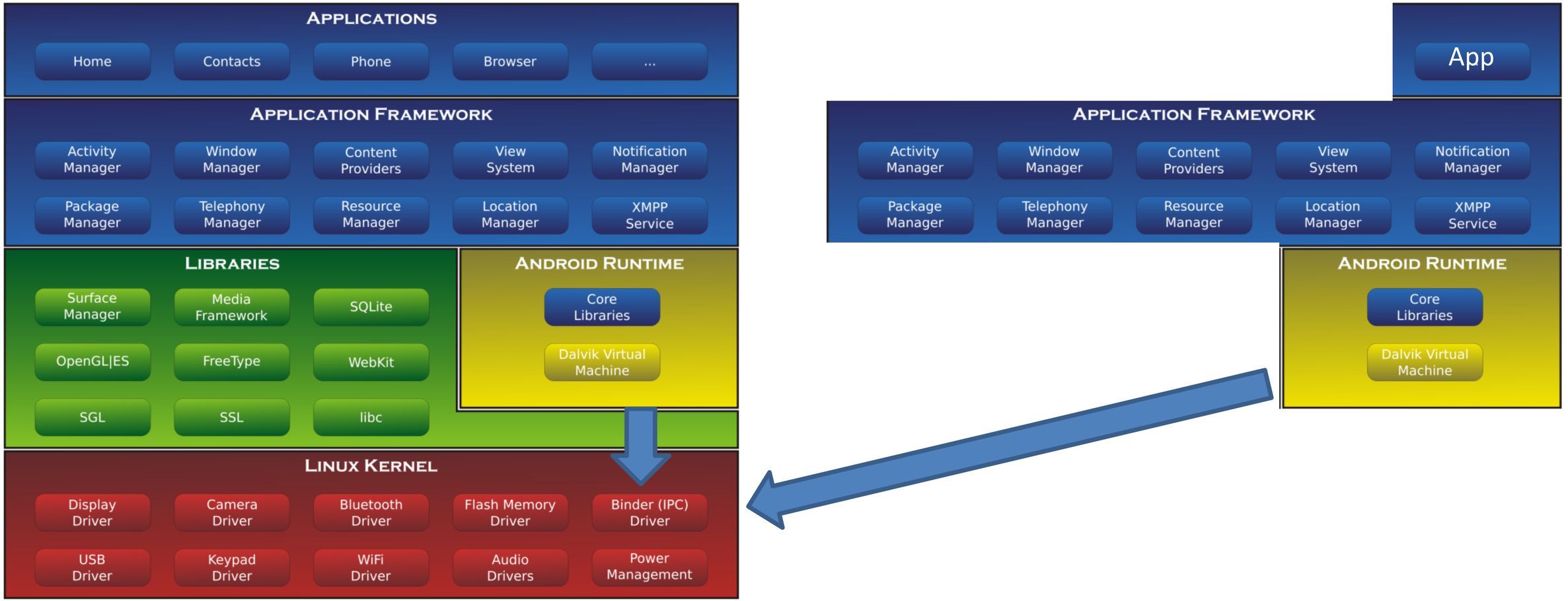
Weakness in unix isolation, privileges

- Shared Resources
  - Since any process can create files in /tmp directory, an untrusted process may create files that are used by arbitrary system processes
- Time-of-Check-to-Time-of-Use (TOCTTOU)
  - Typically, a root process uses system call to determine if initiating user has permission to a particular file, e.g. /tmp/X.
  - After access is authorized and before the file open, user may change the file /tmp/X to a symbolic link to a target file /etc/shadow.

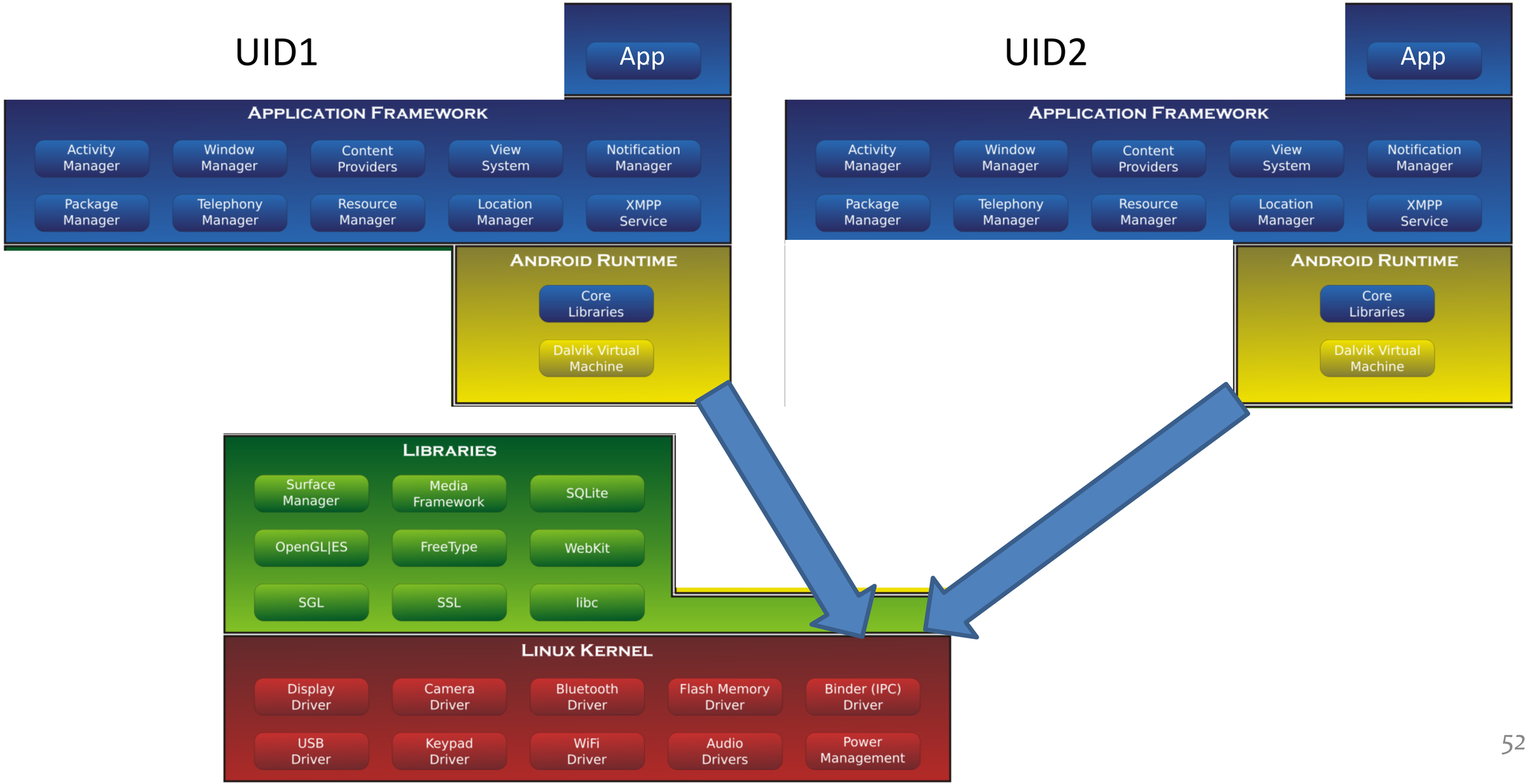
Android process isolation

- Android application sandbox
  - Isolation: Each application runs with its own UID in own VM
    - Provides memory protection
    - Communication limited to using Unix domain sockets
    - Only ping, zygote (spawn another process) run as root
  - Interaction: reference monitor checks permissions on inter-component communication
  - Least Privilege: Applications announces permission
    - User grants access at install time (request permissions during runtime in latest Android versions)

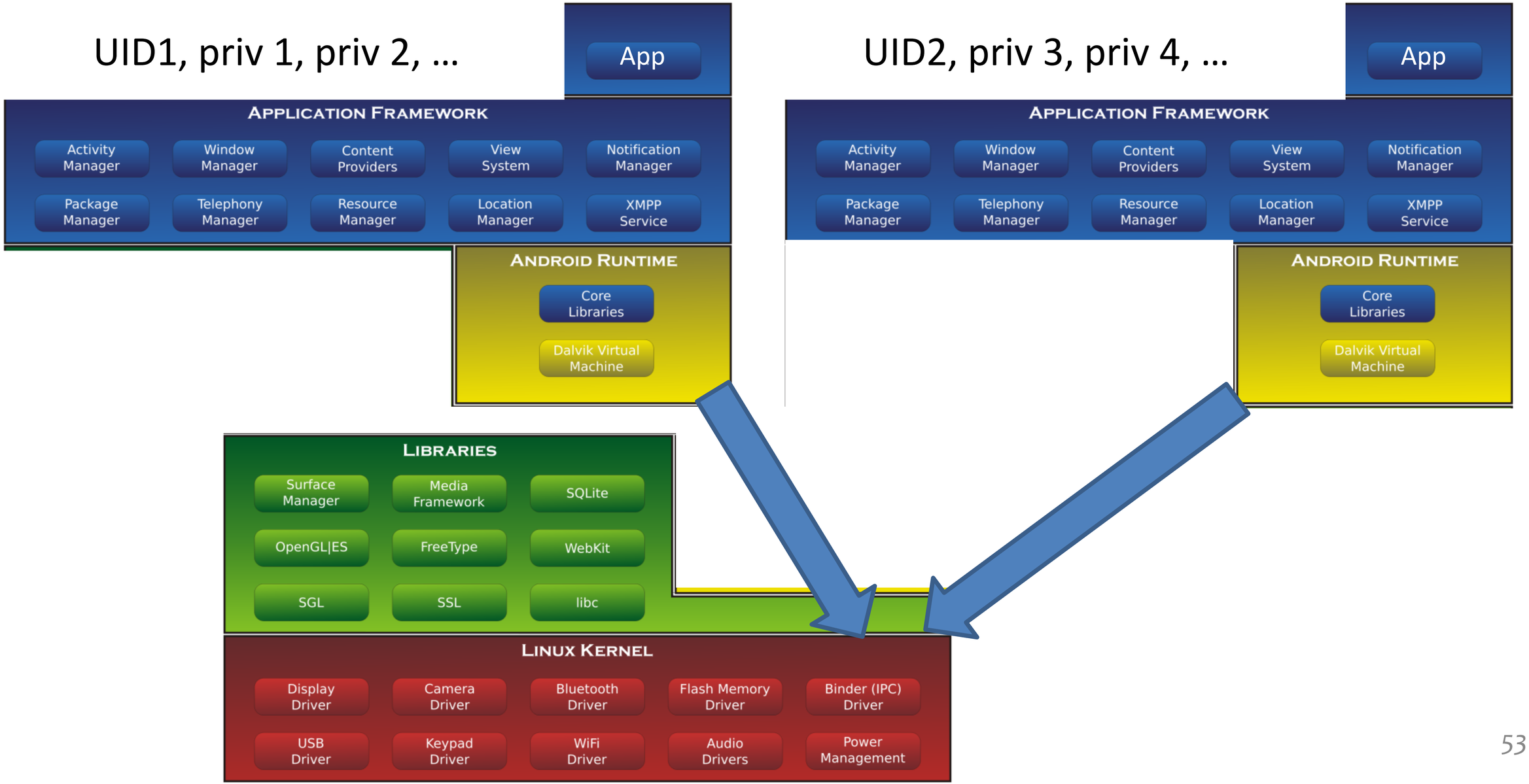
Isolation: different apps under different UIDs

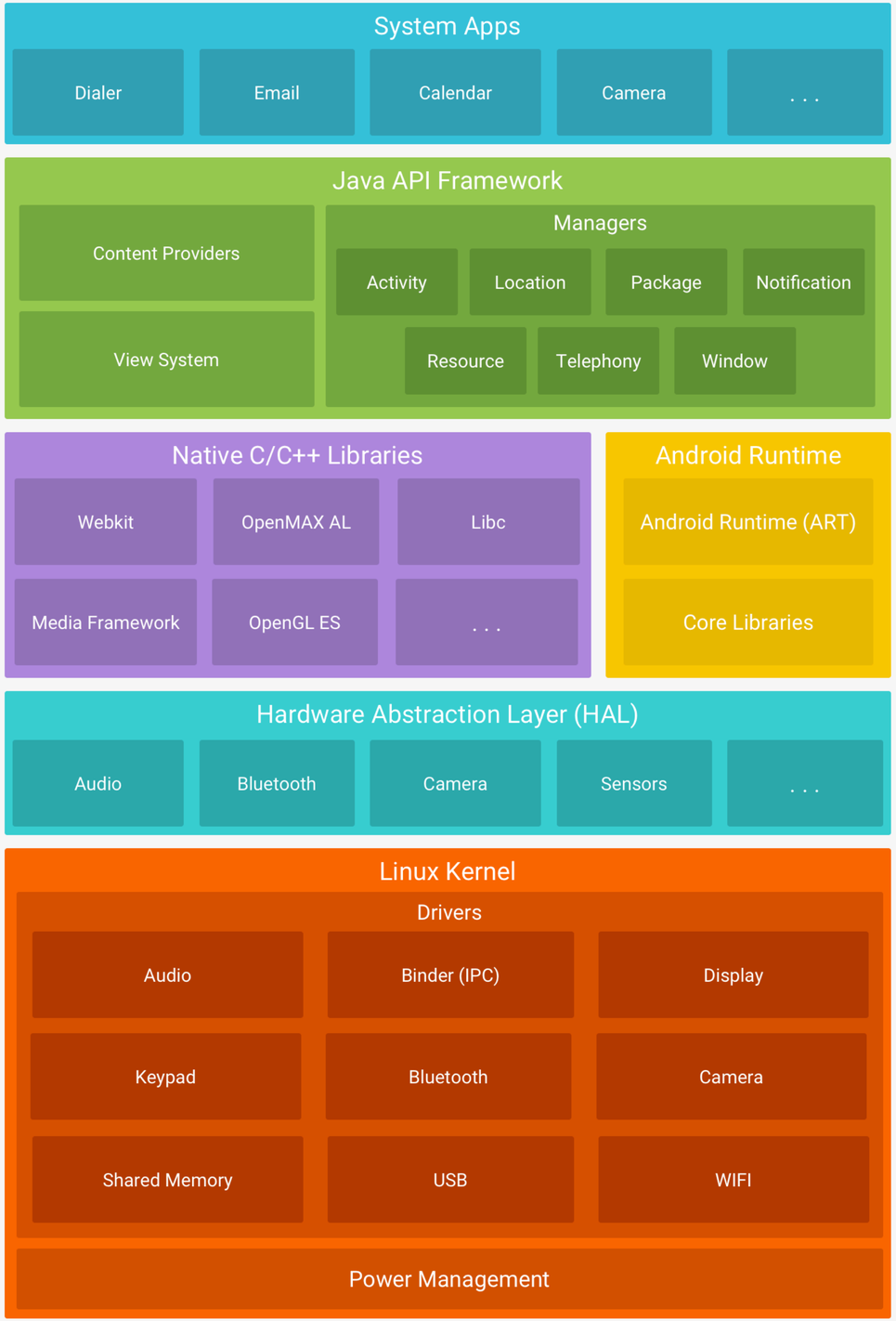


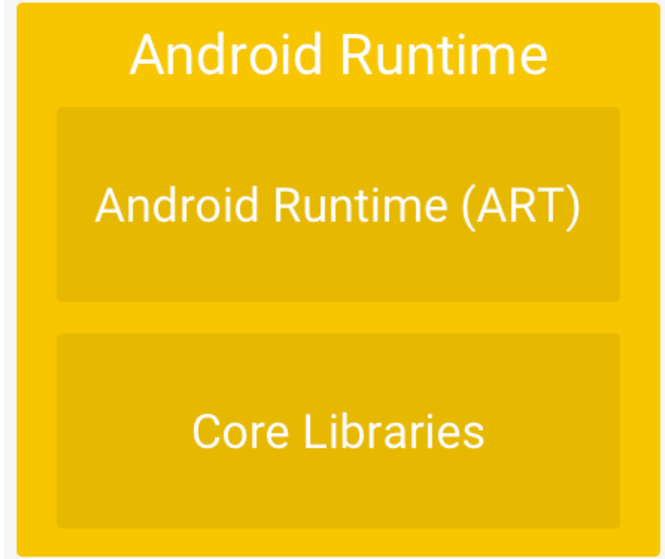
Isolation: different apps under different UIDs



Privileges set at install time




System Apps
Dialer
Email
Calendar
Camera
. . .
Java API Framework
Content Providers
View System
Managers
Activity
Location
Package
Notification
Resource
Telephony
Window
Native C/C++ Libraries
Webkit
OpenMAX AL
Libc
Media Framework
OpenGL ES
. . .
Android Runtime
Android Runtime (ART)
Core Libraries
Hardware Abstraction Layer (HAL)
Audio
Bluetooth
Camera
Sensors
. . .
Linux Kernel
Drivers
Audio
Binder (IPC)
Display
Keypad
Bluetooth
Camera
Shared Memory
USB
WIFI
Power Management


Android Runtime
Android Runtime (ART)
Core Libraries

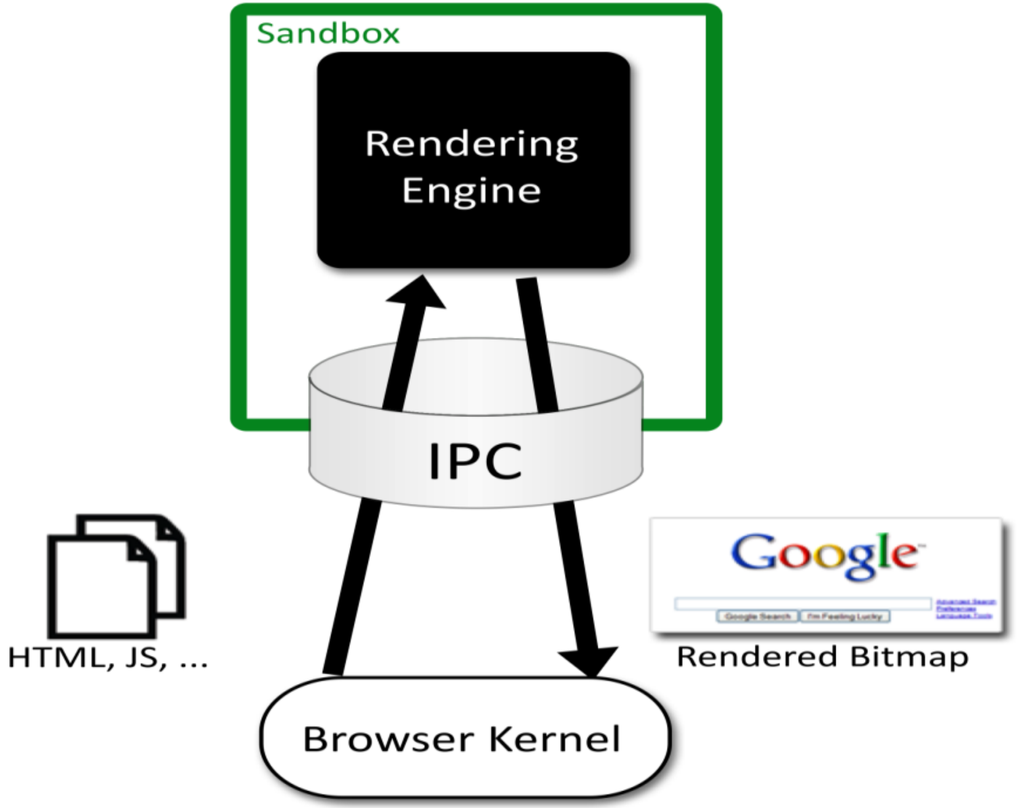
Let's look at browser example

- Browser is an execution environment
  - Has access control policies similar to an OS
- Browser runs under control of an OS
  - Use least privilege to keep the browser code secure against attacks that would break the browser enforcement of web security policy

Topic here: implementation of browser using least privilege

Chromium Security Architecture

- Browser ("kernel")
  - Full privileges (file system, networking)
- Rendering engine
  - Can have multiple processes
  - Sandboxed
- Browser plugin
  - Can request cross-origin permissions



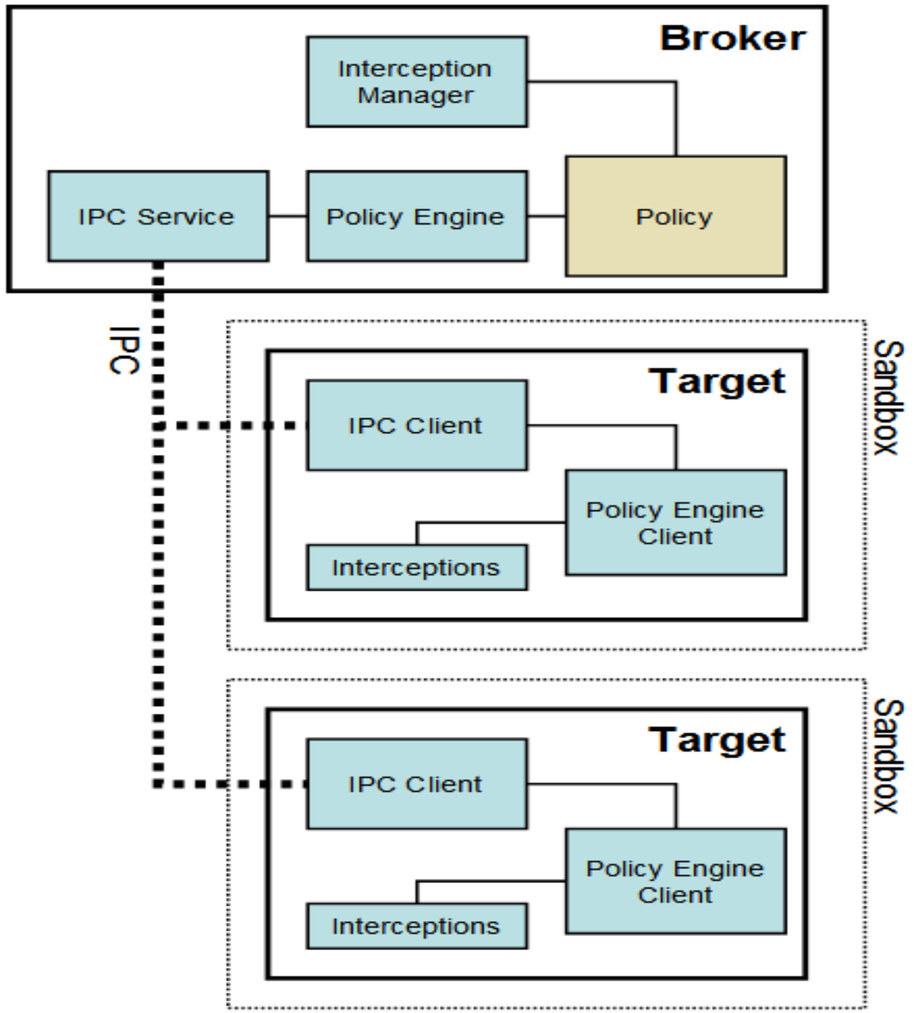
Task Allocation

Rendering Engine	Browser Kernel
HTML parsing	Cookie database
CSS parsing	History database
Image decoding	Password database
JavaScript interpreter	Window management
Regular expressions	Location bar
Layout	Safe Browsing blacklist
Document Object Model	Network stack
Rendering	SSL/TLS
SVG	Disk cache
XML parsing	Download manager
XSLT	Clipboard

Both
URL parsing
Unicode parsing

Chromium

Communicating sandboxed components



See: https://chromium.googlesource.com/chromium/src/+/HEAD/docs/design/sandbox.md

Leverage OS Isolation: Linux

- The setuid sandbox
- User namespaces sandbox
  - based on (unprivileged) user namespaces in the Linux kernel.
- The seccomp-bpf sandbox
  - Designed to shelter the kernel from malicious code executing in userland.

See https://chromium.googlesource.com/chromium/src.git/+/HEAD/docs/linux/sandboxing.md

Discussion?

- How does Chrome architecture use principle of least privilege?
  - What are the isolated modules?
  - Which privileges are given to each module?
- Why is this effective?
- Are there other ways you could use operating system features to improve isolation and least privilege?

Summary

- Security principles
  - Isolation
  - Principle of Least Privilege
  - Android examples
- Access Control Concepts
  - Matrix, ACL, Capabilities
- OS Mechanisms
  - Unix: UID, ACL, Setuid
- Browser security architecture
  - Isolation and least privilege example

Q&A