System Security

Kehuan Zhang

IERG4130 2022

Topics to Be Covered

- Operating System Security
  - Password
  - Access Control
- System Security in Real World
  - Mobile Security (will be provieded as video)

Operating System Security

Functionalities of Operating System

- Resource Management
- What resources?
  - Users $\rightarrow$ separation and management of Multi-users
  - Computation resources $\rightarrow$ CPU time and process scheduling
  - Storage $\rightarrow$ File system, access control
  - Memory $\rightarrow$ process Isolation, virtual address vs. physical address
  - I/O $\rightarrow$ provide drivers, but also include access control
    - ★ Under Unix/Linux, I/O devices are just special files
- Key Concept in terms of security
  - **authentication**: verify user identity $\rightarrow$ password
  - **authorization**: what resource a user can access $\rightarrow$ access control

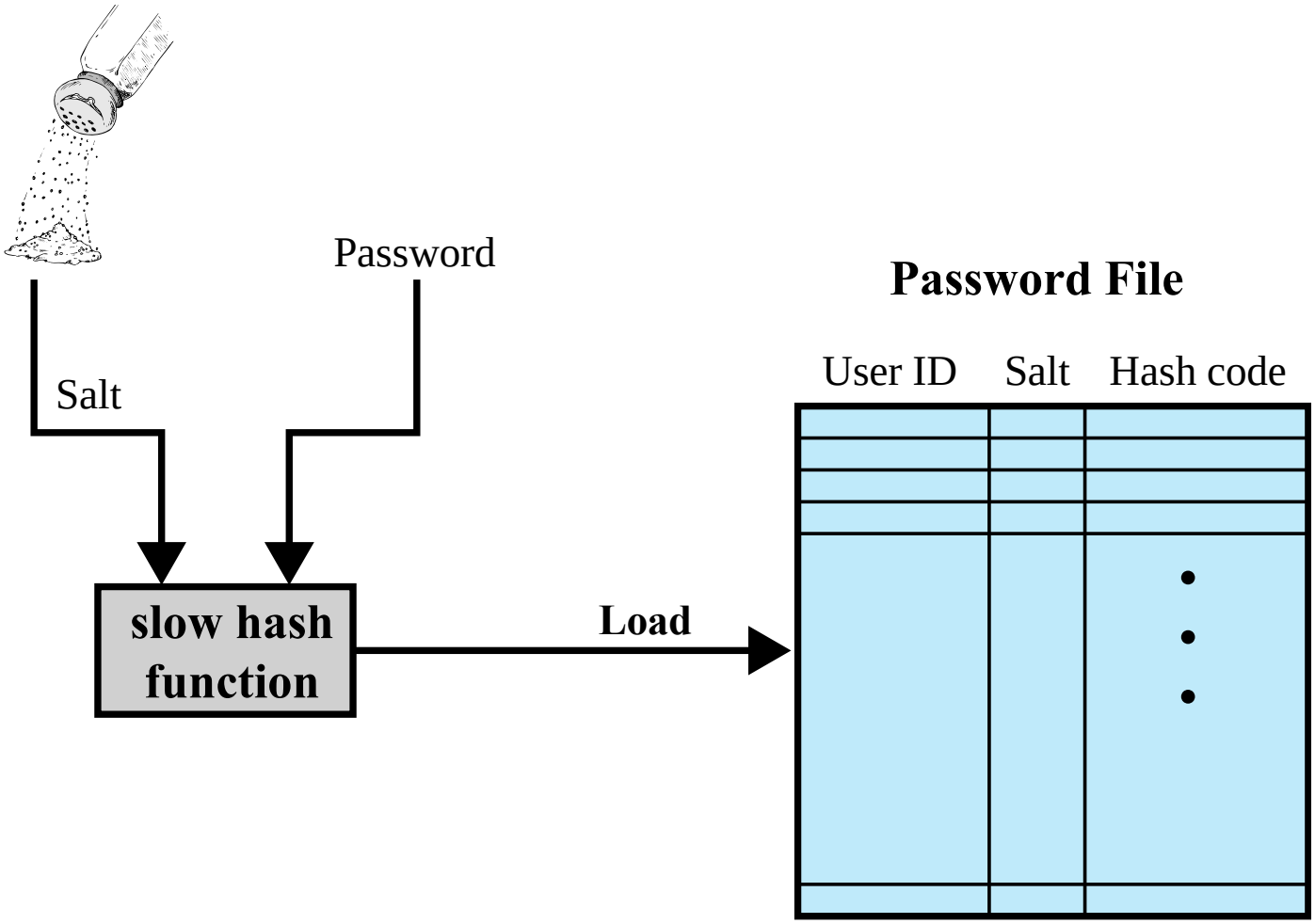
Basic Methods (Factors) to Do Authentication

- Something you know
  - E.g., password, private key
- Something you have
  - E.g., CULink card, Hong Kong ID card, cookie in Web
- Who you are
  - Biometrics derived from your biological features, like fingerprint, Iris
- What you do
  - Also related to biometrics, e.g., how you type, how you walk
- Where you are
  - Location or context-based, e.g., based on IP or host you are working on
- Or combination of above
  - Also called two- or multi-factor authentication

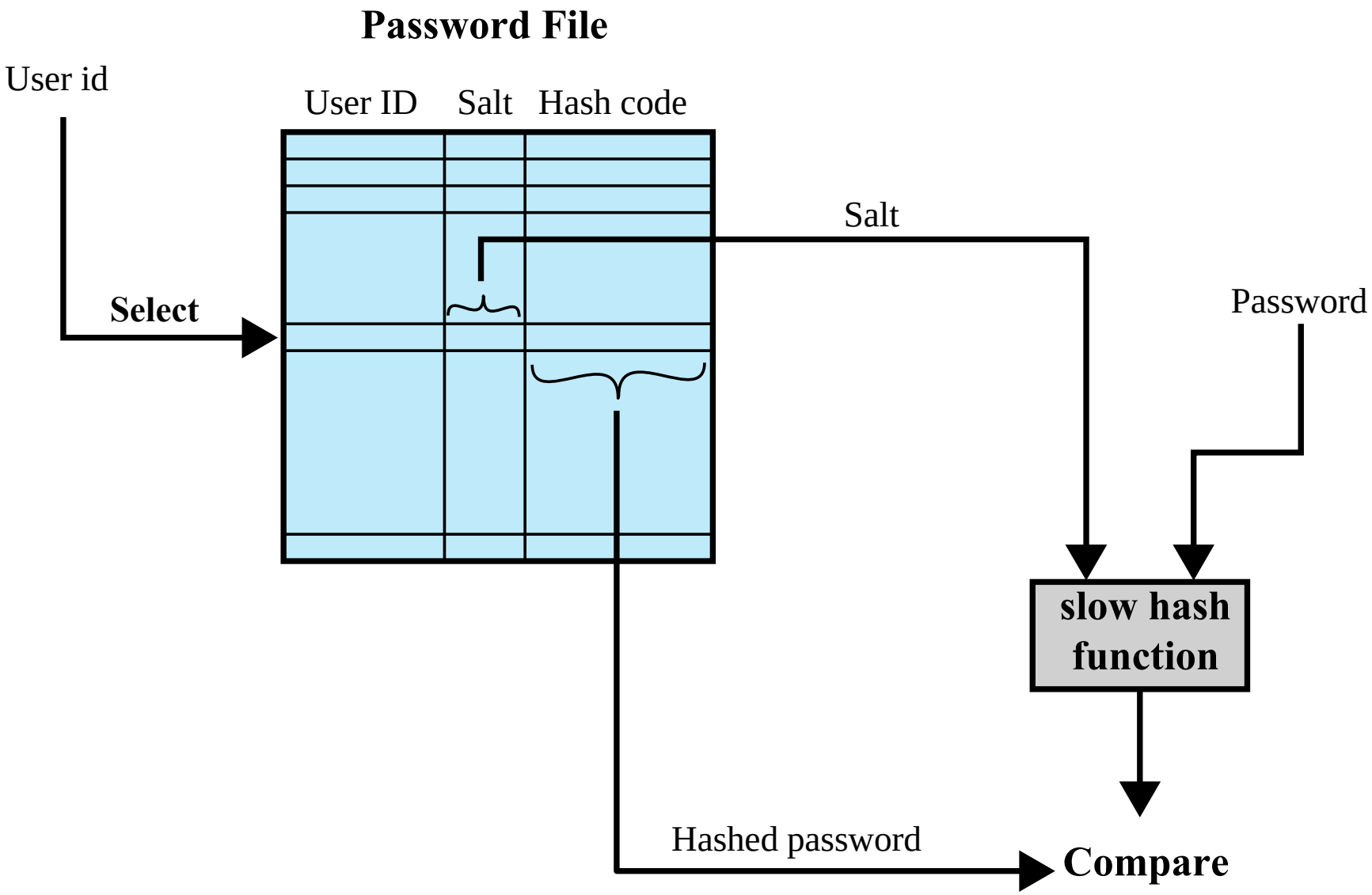
Attacks to Password

- Offline dictionary attack (e.g., cracking a hashed password)
  - Need at least 2^{64} or or 11 characters to be secure, but needs to be longer in practice, since on average 2-bit/character in English
- Specific account attack (e.g., dictionary attack on account)
  - Defense: Max attempt counter, password complexity requirements
- Popular password attack (try few passwords on many accounts)
  - Defense: Password complexity requirements
- Password guessing against single user (do research then guess)
  - Defense: User training, password complexity requirements
- Workstation hijacking (physically use logged-in workstation)
  - Defense: Physical security, auto-lock timers
- Exploiting user mistakes (Post-Its, sharing, unchanged defaults, . . .)
  - Defense: Training, single-use expiring passwords for new accounts
- Exploiting multiple password use
  - Defense for individual: Password managers with strong crypto
- Electronic monitoring (sniffing network, keylogger, etc.)
  - Defense: Encryption, challenge-response schemes, training

UNIX Password Scheme - Creation



UNIX Password Scheme - Verification



Features of UNIX Password Scheme

- Storage:
  - Originally hashed value in /etc/passwd file is public-readable
  - Now put /etc/shadow which needs root privilege
- Algorithm
  - Originally: small hash, few iterations
  - Later: MD5 hash, more iterations
  - Now: SHA 512 hash, configurable iterations
- The salt serves three purposes:
  - Prevents duplicate passwords to be noticeable.
  - Effectively increases the length of the password.

How to Set Up Password Securely?

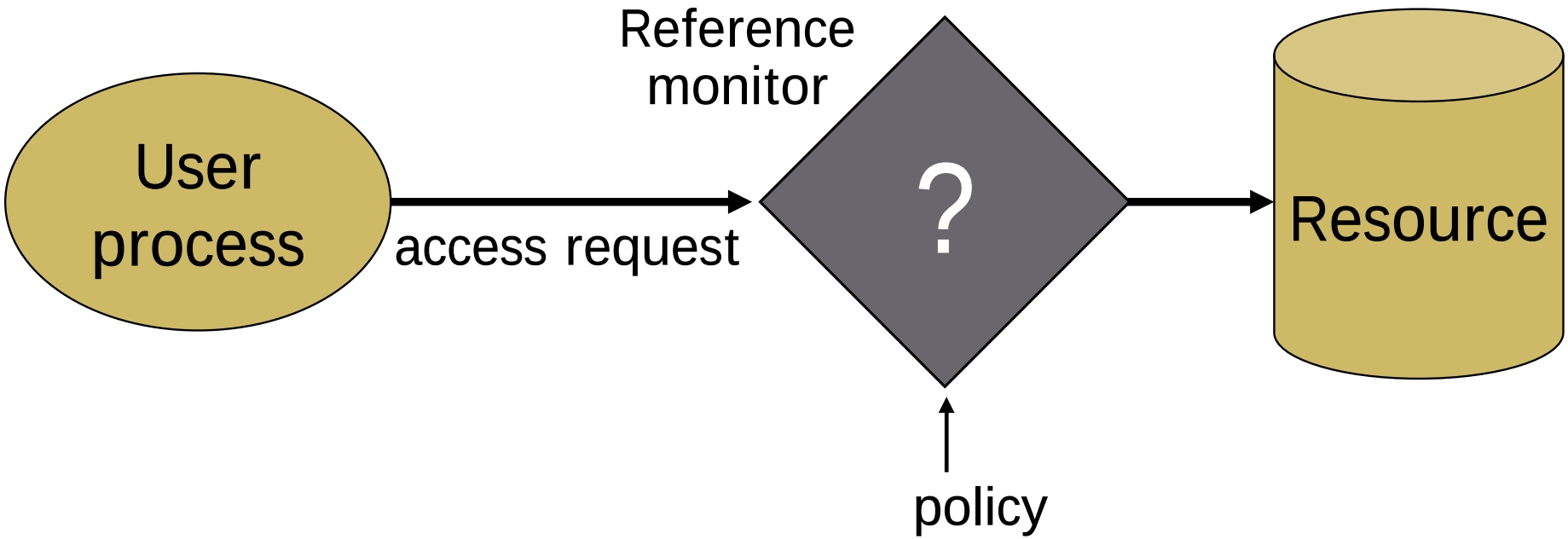
- User education
  - Users can be told the importance of using hard to guess passwords and can be provided with guidelines for selecting strong passwords
- Computer generated passwords
  - Users have trouble remembering them (good for single-use, bad for long-term)
- Reactive password checking
  - System periodically runs its own password cracker to find guessable passwords
- Complex password policy
  - User is allowed to select their own password, however the system checks to see if the password is allowable, and if not, rejects it
  - Goal is to eliminate guessable passwords while allowing the user to select a password that is memorable

Common Practices for Password Security

1. Choose password strategically according to target scenarios
   - Complex password for important, while simple password for unimportant
2. Have a template to generate password
   - A fixed string with different suffix, upcase or lower cases
   - Including special character to make password more complex
   - Have a method to generate password out some pieces of information.
     - ★ E.g., part1+part2+part3, each part is same or has some limited choices
3. Use apps on iphone, android or browsers to manage the password
4. Write down complex password on a notebook, and keep that notebook in secure places
   - Can have some characters masked
     - ★ So you can recall what are missing but others could not
5. Change periodically, half year or one year
6. Enable two factor authentication to mitigate weak password problem

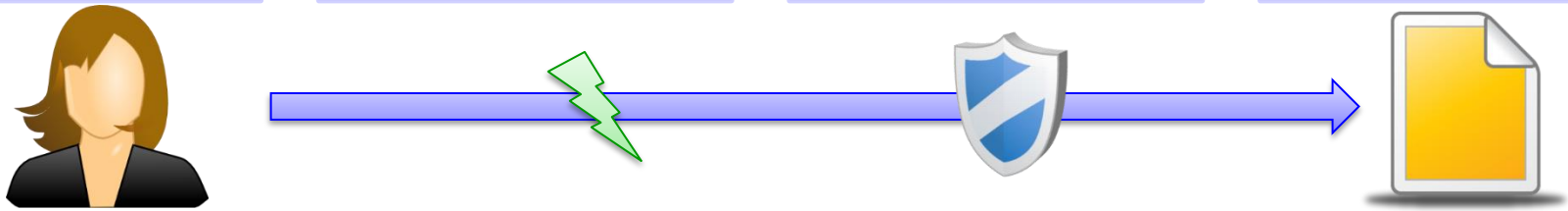
Access Control

- Assumptions
  - System knows who the user is
    - ★ Authentication via name and password, other credential
  - Access requests pass through gatekeeper
    - ★ System must not allow monitor to be bypassed



Basic Terms

Subject (initiator)	**Verb** (request)	**Right** (permission)	**Object** (target)
• The thing making the request (e.g. the user)	• The operation to perform (e.g., read, delete, etc.)	• A specific ability for the subject to do the action to the object.	• The thing that's being hit by the request (e.g., a file).



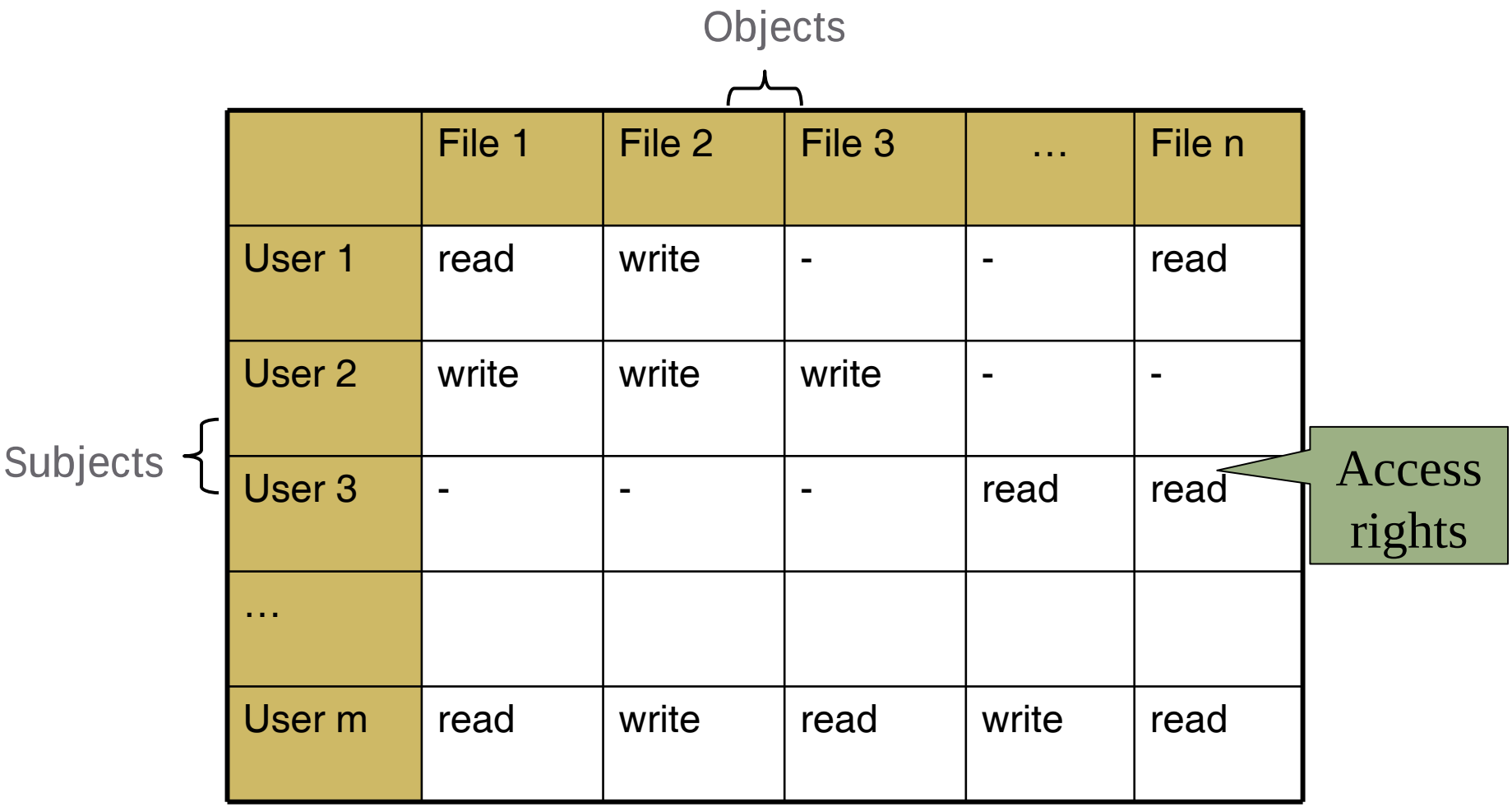
Categories of Access Control Policies

- Discretionary AC (DAC): There's a list of permissions attached to the subject or object (or possibly a giant heap of global rules).
- Mandatory AC (MAC): Objects have classifications, subjects have clearances, subjects cannot give additional permissions.
  - An overused/abused term
- Role-Based AC (RBAC): Subjects belong to roles, and roles have all the permissions.
  - The current Enterprise IT buzzword meaning "good" security
- Attribute-Based AC (ABAC): Subjects and objects have attributes, rules engine applies predicates to these to determine access
  - Allows fine-grained expression
  - Usually complex, seldom implemented

Discretionary Access Control (DAC)

- Scheme in which an entity may 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

Access Control Matrix



	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

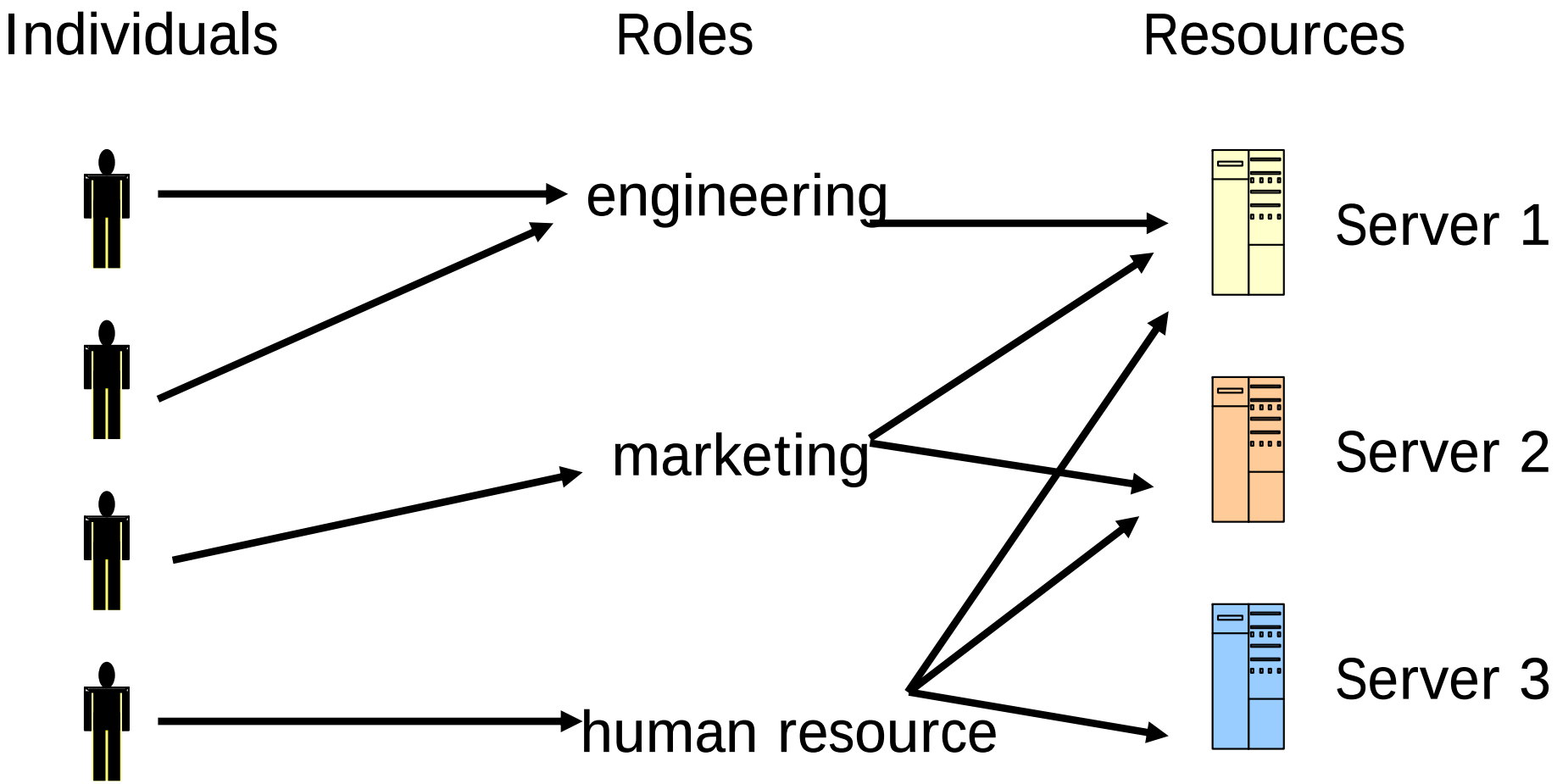
Two Implementation Strategy: ACL vs. Capabilities

- Access control list
  - Associate list with each object (i.e., store above matrix based on columns)
  - Check user/group against list
  - Relies on authentication: need to know user
- Capabilities
  - User holds a "ticket" for each resources
  - Two variations
    - ★ Store previous matrix based on rows (associated with user), under OS control
    - ★ Un-forgeable ticket in user space managed by OS, and can be passed from one process to another
    - ★ Reference monitor only checks the ticket (so no need to verify user identity)

Role Based Access Control (RBAC)

- Assign users to roles
  - Administrator, PowerUser, User, Guest
- Assign permission to roles
  - When a role changes, everyone gets the change.
  - When a user's role changes, that user gets a whole new set of permissions.
- Roles can have hierarchy
  - Partial order of roles
  - Each role gets permissions of roles below
  - List only new permissions given to each role

An Example of RBAC



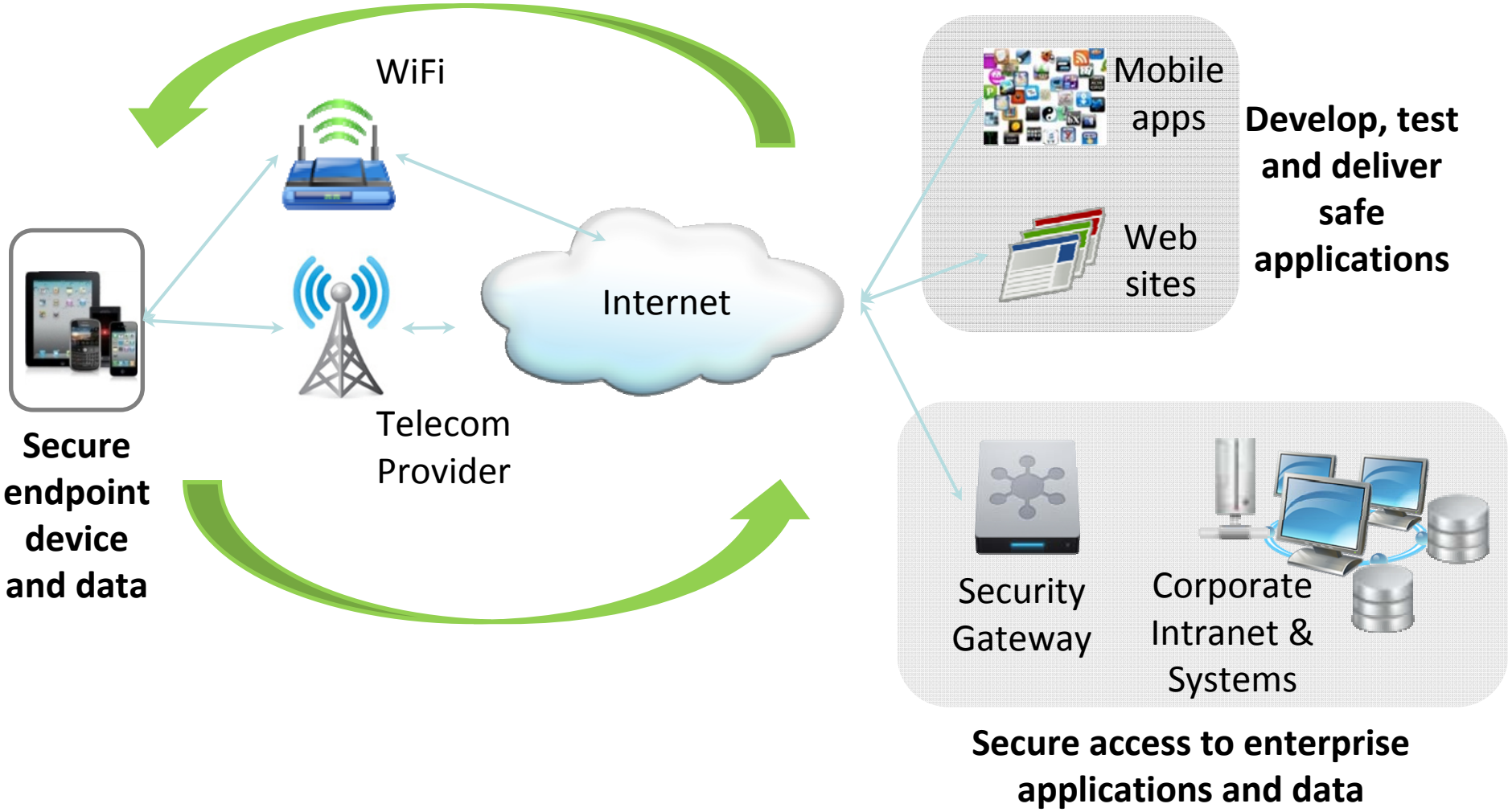
- Advantage: user's change more frequently than roles

Side-Channel Attacks to the Access Control

- With the access control policy securely enforced, it is difficult to get the data directly
- However, attackers can access the data indirectly, which is called "side-channel" attack
- Some Examples:
  - Infer victim's Google search keyword from encrypted traffic
  - Cache based side-channel attacks
  - Reading Material: Meltdown and Spectre attack
    - `https://meltdownattack.com/`
- A **shared** resource could be exploited by side-channel attacks
- A related concept – covert channel attack
  - By "covert" it means a kind of secret channel (which is not indented for communication)
  - E.g: modulate packets size/length/numbers etc. with stolen data
    - The payloads in these packets are irrelevant
    - The secret is embedded in the "side-channel"
- Sometimes, the same information could be used as side-channel and covert-channel
  - It depends on whether you are extracting or embedding the information

System Security in Real World: Mobile Security

Visualizing Mobile Security



Data on Mobiles Device Are Important

- Mobile devices make attractive targets:
  - People store much personal info on them: email, calendars, contacts, pictures, etc.
  - Sensitive organizational info too...
  - Can fit in pockets, easily lost/stolen
  - Built-in billing system: SMS/MMS (mobile operator), in-app purchases (credit card), etc.
- Many new devices have near field communications (NFC), used for contactless payments, etc.
- Your device becomes your credit card
  - Location privacy issues

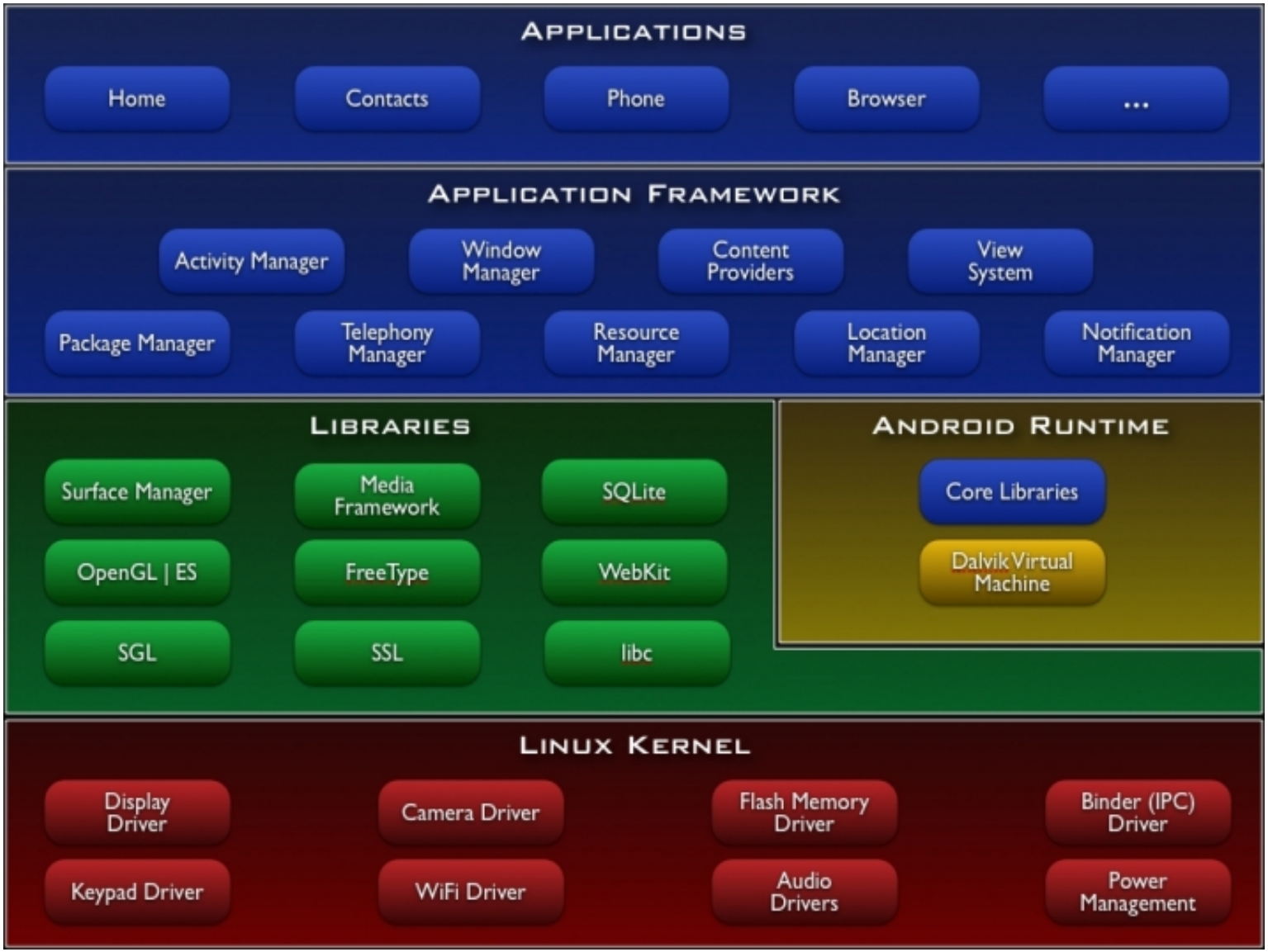
Mobile Threats and attacks

- Mobile Device Loss / Theft
  - A serious problem in early days when smartphone was expensive
- Mobile Malware
  - Trojans, spyware, adware, botnets
- Device Search and Seizure
  - Law enforcement, e.g., US custom officers, can search your mobile device without warrant
- Location Disclosure (personal privacy info)

Android Security Goals and Mechanisms

- Security goals
  - Protect user data
  - Protect system resources (hardware, software)
  - Provide application isolation
- Application Isolation and Permission Requirement
  - Mandatory application sandbox for all applications
  - Secure inter-process communication
  - System-built and user-defined permissions
  - Application signing
- Examples:
  - Prevents user A from reading user B's files
  - Ensures that user A does not access user B's CPU, memory resources
  - Ensures that user A does not access user B's devices (e.g. telephony, GPS, Bluetooth)

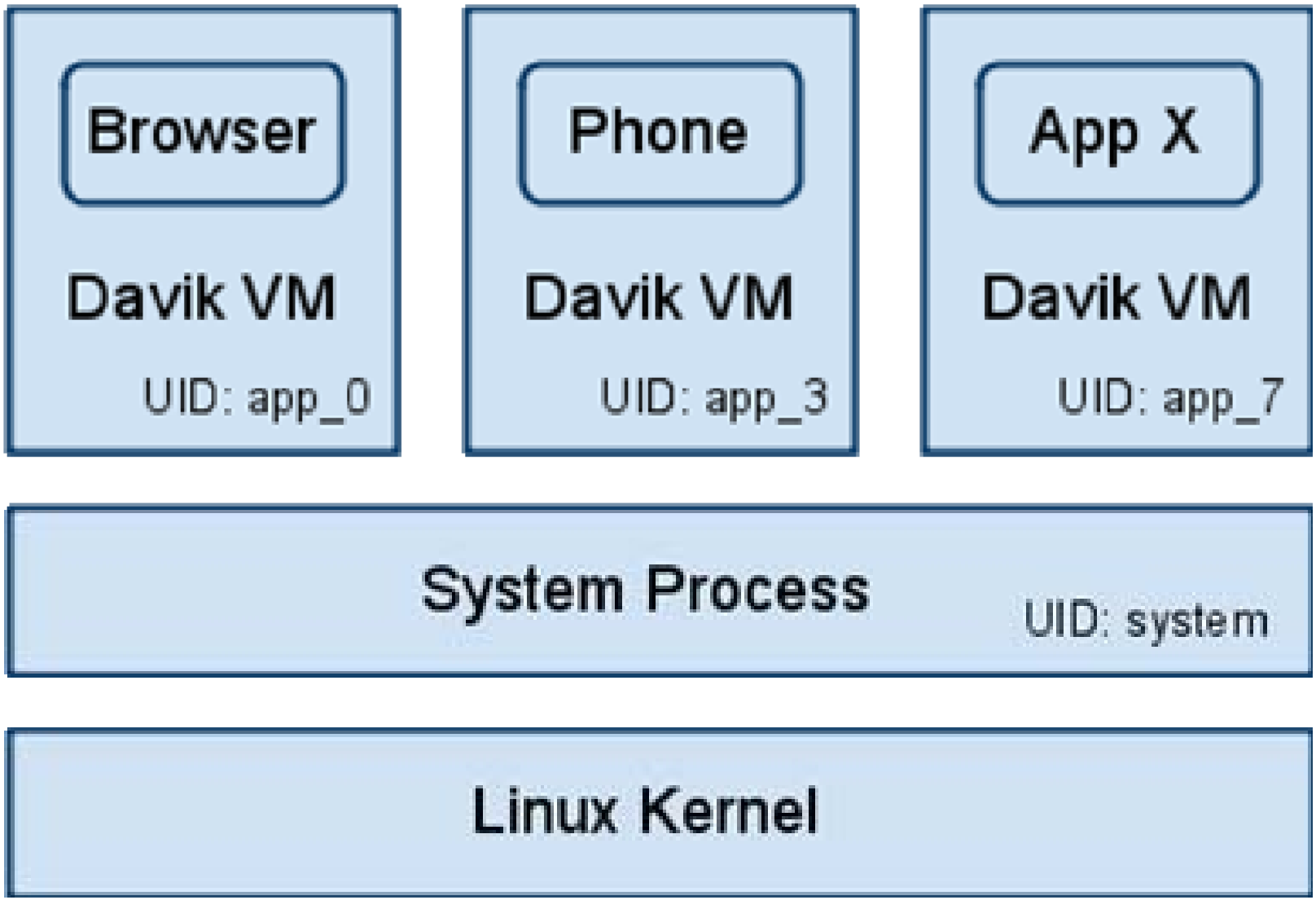
Android Security Software Stack



Application Sandbox in Android

- Each component assumes that the components below are properly secured.
- All code above the Linux Kernel is restricted by the Application Sandbox
- Linux kernel is responsible sandboxing application
  - "mutually distrusting principals"
  - Default access to only its own data
- The app Sandbox apps can talk to other apps only via Intents (message) , IPC, and Content Providers
- To escape sandbox, permissions is needed

Android Sandbox

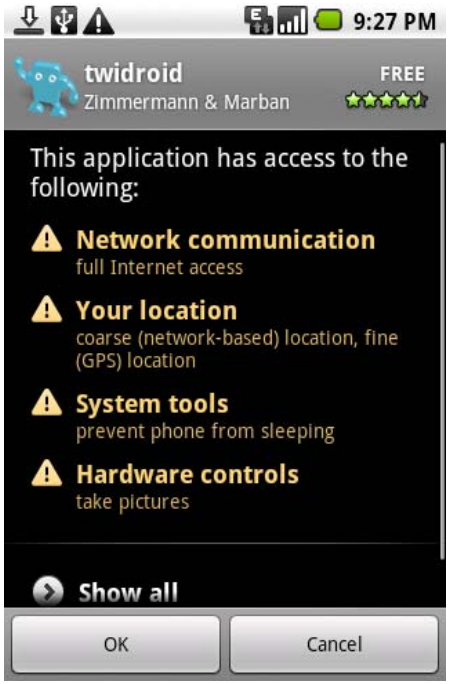


Data Encryptions and Password Protection

- Android 3.0+ provides full file system encryption, so all user data can be encrypted / decrypted automatically by the kernel
- For a lost or stolen device, full file system encryption on Android devices uses the device password to protect the encryption key, so modifying the bootloader or operating system is not sufficient to access user data without the user's device password.
- Password Protection
  - Android can require a user-supplied password prior to providing access to a device. In addition to preventing unauthorized use of the device, this password protects the cryptographic key for full file system encryption.

Permissions

- Permissions
  - In Android, each application runs as its own user. Unless the developer explicitly exposes files to other applications, files created by one application cannot be read or altered by another application.



Access Control based on Permissions

- Based on Whitelist model
- 2 types
  - Provided by Android OS
  - Defined by apps
- 4 protection levels
  - Normal: low risk, automatically grant
  - Dangerous: higher-risk, need to ask for user's explicit approval
  - Signature: only apps with the same certificate
  - signatureOrSystem
- But permission-based security was abused and broken

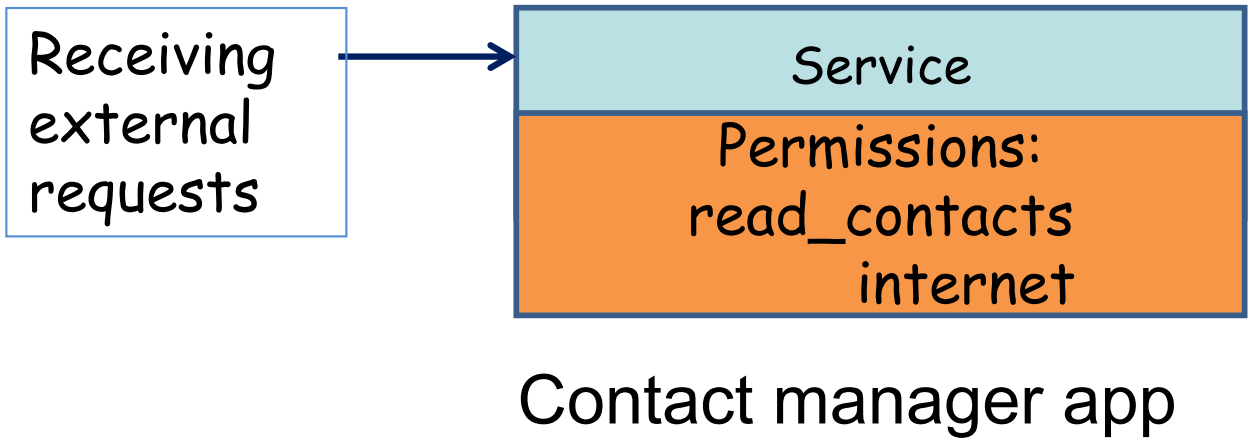
Attacks to Permission Mechanism

- Over-privilege attack
  - Malicious apps will ask for more permissions than actuallyed needed
  - Android relies on the users to decide whether to accept or not
- Confused Deputy Attack
- Collusion Attack

Confused Deputy Attack

- What is it?
  - Permissions owned and exposed by vulnerable Apps were exploited by malicious Apps

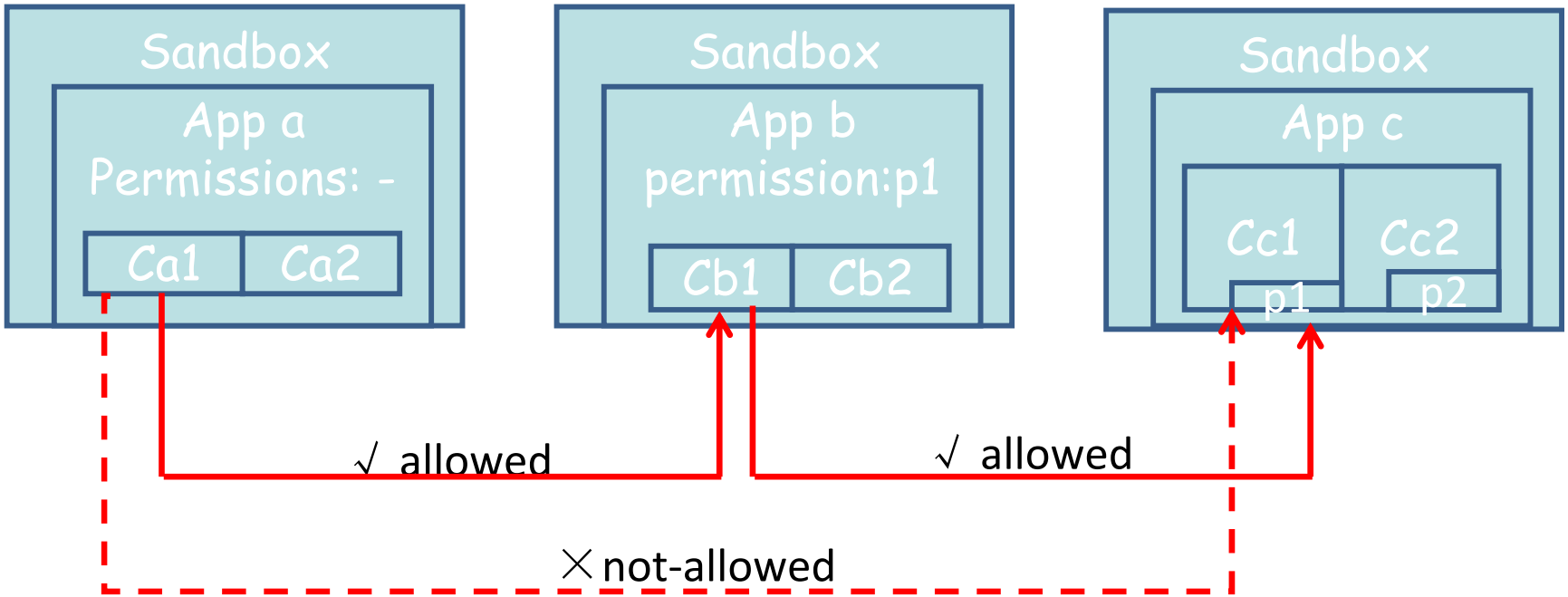
- An Example



- Defense?
  - Be careful about interfaces exposed to outside
  - Do authorization checking for incoming request

Collusion Attack

- What is it?
  - Two or more apps cooperate purposely to achieve privilege escalation
- Type 1: ICC (Inter-Component Communication)



Collusion Attack (cont.)

- Type 2: Covert Channel
  - Indirect communication : system components and file systems

- Example
  - Vibration settings
  - Volume settings
  - Screen
  - File locks

General Suggestions for Mobile Security

- Secure mobile device configuration:
  - Do not jailbreak or root your phone
  - Prohibit installation of pirate Apps or Apps without legitimate sources
  - Enable auto-lock
  - Enable password/PIN/thumbprint protection
  - Disable/discourage auto-completion for passwords
  - Enable remote wipe
  - Up-to-date OS/software
  - Encrypt sensitive data