CS 3923 / CS 6813 - Internet Security & Privacy

Access Control -- part 1

Matrices, ACLs, Linux, Windows, and Android

Justin Cappos

[*] Slides based upon materials maintained by Justin Cappos at NYU

NEW YORK UNIVERSITY

Extra Credit Opportunities

1. Extra Credit - Password Manager

- Explore 3 password managers and compare them
- Choose one to use
- Submit a 1 page PDF about your experiences

2. Attend CSAW - do a 1 page write-up

Access Control - Definition

Access control is a series of mechanisms used by management, to specify what users can do, which resources they can access, and what operations they can perform on a system. More generally, it permits managers of a system to direct or restrain the behavior, use and content of a system.

Elements of Access Control

- Access Controls: The security features that control how users and systems communicate and interact with one another.
- Object: A passive entity that contains information
- Subject: An active entity that requests access to an object or the data in an object
- Access: The flow of information between subject and object

Access Control – More Formally

- Any system consists of objects and subjects (active objects such as processes, users etc.) which access these objects.
- The security policy of a system defines
  - What a subject is allowed to do
  - What may be done with an object
- In other words – Access Control
- Two issues –
  - How do you specify an access control policy?
  - How do you enforce an access control policy?

Remember…

- The three main security principles also pertain to access control:
  - Availability

    Mechanisms put into place to ensure the objects are accessible to subjects
  - Integrity

    Protecting objects from being altered in any unauthorized fashion
  - Confidentiality

    Assurance that information is not disclosed to unauthorized subjects

Access Control - Abstraction

- Access control is established by implementing three distinct functions
  - Identification
  - Authentication
  - Authorization

(Note that identity management is the broad term that includes the use of different products to identify, authenticate, and authorize users through automated means.)

Identification

- Method of establishing the subject’s identity*.
    - Use of user name or other public information.
    - Need to conform to identification component requirements.
        - Each value should be unique, for user accountability;
        - A standard naming scheme should be followed;
        - The value should be non-descriptive of the user’s position or tasks; and
        - The value should not be shared between users.

*Note: Examples of subject's include user, program, process.

Authentication

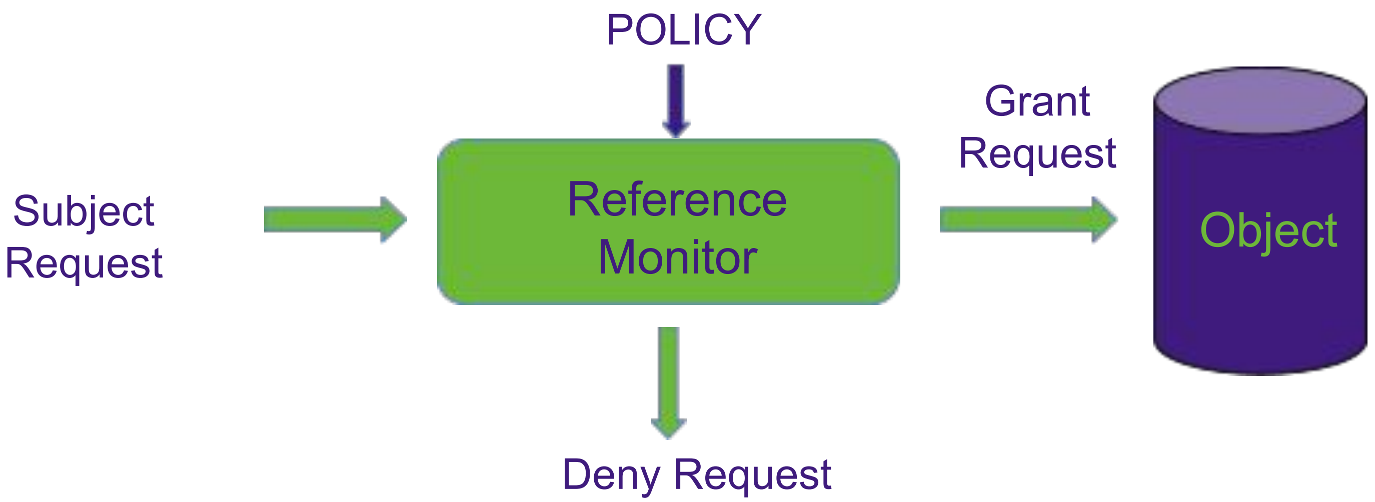
- Method of proving the identity.
  - Something a subject is, has, or does.
  - Use of biometrics, passwords, passphrase, token, private cryptographic key, or other private information.

Authorization

- Determines that the proven identity has some set of characteristics associated with it that gives it the right to access the requested objects.
    - Access Criteria can be thought of as:
        - Roles
        - Groups
        - Location
        - Time
        - Transaction Types

Access Control – Conceptual Model

- Assumptions
  - System knows who the user is
    - Authentication via credentials
  - Access requests pass through the gatekeeper, aka, reference monitor
    - System must not allow monitor to be bypassed



- An actual system may not include an explicit reference model
  - But we need to define functionality of the reference monitor and design mechanisms for its implementation.

Access Control Models

- How is access control decided?
- Three main techniques
  - Discretionary
  - Mandatory
  - Non-Discretionary (Role Based)

Access Control Models (continued)

- Discretionary Access Control (DAC)
  - A system that uses discretionary access control allows the owner of the resource to specify which subjects can access which resources.
  - Access control is at the discretion of the owner.

Access Control Models (continued)

- Mandatory Access Control (MAC)
  - Access control is based on a security labeling system. Users have security clearances and resources have security labels that contain data classifications.
  - This model is used in environments where information classification and confidentiality is very important (e.g., the military).

Access Control Models (continued)

- Non-Discretionary (Role Based) Access Control Models
  - Roles indicate restrictions. Identities have roles
  - Role Based Access Control (RBAC) uses a centrally administered set of controls to determine how subjects and objects interact.
  - It is the best system for an organization that has high turnover.
  - Also very commonly used in cloud-native / asymmetrical decentralized systems

Access Control Models (continued)

- Relation Based Access Control Models
  - Relation Based Access Control (ReBAC) uses the relationship between subjects to enforce an access control policy
  - For example, your “friends” can see your social network content but others cannot.

Access Control Techniques

- There are a number of different access controls and technologies available to support the different models.
  - Rule Based Access Control
  - Constrained User Interfaces
  - Content Dependent Access Control
  - Context Dependent Access Control
  - Access Control Matrix

Access Control Techniques (continued)

- Rule-Based Access Control:
    - Uses rules that indicate what can and cannot happen between a subject and an object.
    - For example, “No entry after 5PM.”
    - Not necessarily identity based.
    - Traditionally, rule-based access control has been used in MAC systems as an enforcement mechanism.
    - For example, “This app can use my microphone, only while I run it this time.”

Access Control Techniques (continued)

- Constrained User Interfaces:
  - Restrict a user by not allowing them certain types of access, or the ability to request certain functions or information
  - Three major types
    - Menus and Shells
    - Database Views
    - Physically Constrained Interfaces

Access Control Techniques (continued)

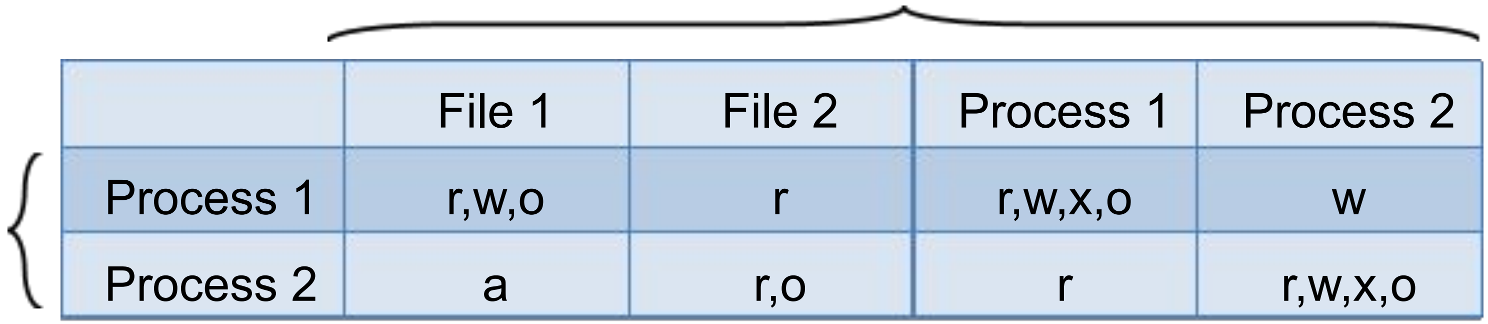
- Content Dependent Access Control:
  - Access to an object is determined by the content within the object.
- Context Based Access Control:
  - Makes access decision based on the context of a collection of information rather than content within an object.

Access Control Techniques (continued)

- Access Control Matrix:
  - Is a table of subjects and objects indicating what actions individual subjects can take upon individual objects.
    - each row represents a subject,
    - each column represents an object, and
    - each entry is the set of access rights for that subject to that object.

Access Control Matrix (ACM) - Example

- Consider system with two files and two processes. Set of rights is - r,w,x,a,o (read, write, execute, append, own).



Objects

Subjects

	File 1	File 2	Process 1	Process 2
Process 1	r,w,o	r	r,w,x,o	w
Process 2	a	r,o	r	r,w,x,o

- As the number of entries increases, the complexity of the file system increases quickly, hence this is system is inefficient for general use.

Implementation Concepts for ACM’s

- Authorization Table
  - Report non-empty entries of ACM in a table with three columns.
- Access control list (ACL)
  - Store each column of ACM with the object it represents
- Capabilities

  Will be discussed next time

* Authorization tables are generally used in database management systems.

* ACLs are widely used, often with groups.

Access Control Lists (ACL’s)

Intuition: An access control list (acl) is a set of permissions that correspond to an object. Each permission usually specifies a principle and a right.

acl(File A): {(Alice: write), (Bob: read, execute)}

In the above example Alice has the permission to write File A. Bob has the permission to read and execute File A.

ACL - Example

- For ACM shown earlier, corresponding ACL’s are:
- *acl(file 1) = {(proc.1, {r,w,o}) (proc. 2, {a})}*
- *acl(file 2) = {(proc.1, {r}) (proc. 2, {r,o})}*
- *acl(proc.1) = {(proc.1,{r,w,x,o}) (proc.2, {r})*
- *acl(proc.2) = {(proc.1,{rw}) (proc.2, {r,w,x,o})*

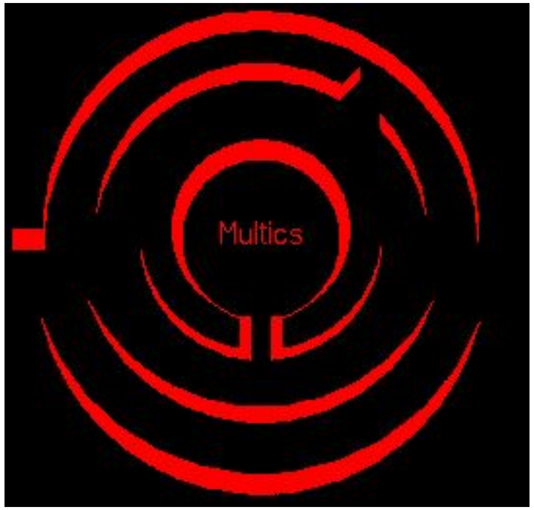
Abbreviated ACL’s

- Although the same amount of storage is used with ACL's, it is now distributed.
- To further reduce storage, one can abbreviate ACL’s as in UNIX.
- One can also assign default access to groups of subjects as well as specific rights to individual subjects.
    - Two ways of doing this:
        - What is not prohibited is permitted
        - What is not permitted is prohibited.
        - Latter almost always better!! Why?
    - Example: Unix hosts.allow and hosts.deny files

OS Mechanisms

- Multics
  - Ring structure
- Unix
  - File system, Setuid
- Windows
  - File system, Tokens, EFS
- Android
  - Apps are users, mediate communication

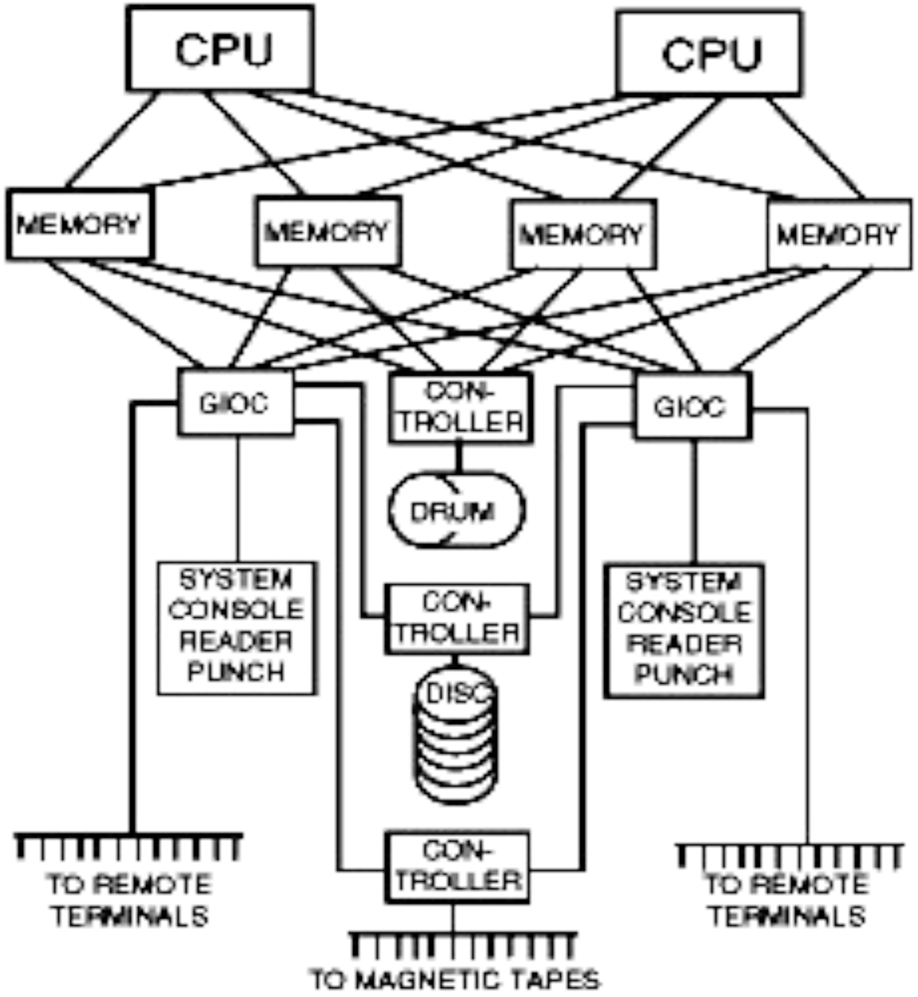
OS Mechanisms - Multics



- Operating System
  - Designed 1964-1967
    - MIT Project MAC, Bell Labs, GE
  - At peak, ~100 Multics sites
  - Last system, Canadian Department of Defense, Nova Scotia, shut down October, 2000
- Extensive Security Mechanisms
  - Influenced many subsequent systems

Multics Time Period

- Timesharing was new concept
  - Serve Boston area with one 386-based PC



Multics Innovations

- Segmented, Virtual memory
  - Hardware translates virtual address to real address
- High-level language implementation
  - Written in PL/1, only small part in assembly language
- Shared memory multiprocessor
  - Multiple CPUs share same physical memory
- Relational database
  - Multics Relational Data Store (MRDS) in 1978
- Security
  - Designed to be secure from the beginning
  - First B2 security rating (1980s), only one for years

Multics Access Model

- Ring structure
    - A ring is a domain in which a process executes
    - Numbered 0, 1, 2, … ; Kernel is ring 0
    - Graduated privileges
        - Processes at ring i have privileges of every ring j > i
- Segments
    - Each data area or procedure is called a segment
    - Segment protection {b1, b2, b3} with b1 > b2 > b3
        - Process/data can be accessed from rings b1 … b2
        - A process from rings b2 … b3 can only call segment at restricted entry points

Multics Process

- Multiple segments
  - Segments are dynamically linked
  - Linking process uses file system to find segment
  - A segment may be shared by several processes
- Multiple rings
  - Procedure, data segments each in specific ring
  - Access depends on two mechanisms
    - Per-Segment Access Control
      - File author specifies the users that have access to it
    - Concentric Rings of Protection
      - Call or read/write segments in outer rings
      - To access inner ring, go through a “gatekeeper”
- Interprocess communication through “channels”

Multics Summary

- Interesting forerunner to modern systems
  - Principled security guarantees
  - Modern processors still have 'ring' model
- Unwieldy in practice

OS Mechanisms

- Multics
    - Ring structure
- Unix
    - File system, Setuid
- Windows
    - File system, Tokens, EFS
- Android
    - Apps are users, mediate communication

OS Mechanisms – Unix

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

Unix Special Users

- Special user with extra privileges –root.
    - UID is 0.
    - Can do (almost) anything!!
    - Holy grail of hackers!
- Other special users
    - daemon or sys – handles some network services
    - ftp – used for anonymous FTP access.
    - uucp – manages UUCP system.
    - guest – used for site visitors.
    - lp - used by printer system
    - Other special users exist

Unix Groups

- Every user belongs to one or more groups.
- The GID of primary group the user belongs to is stored in passwd file.
- Groups useful for access control features.
- /etc/groups contains a list of all groups in the system along with GID’s.
- Some special groups –
  - wheel - group of administrators
  - uucp, lp, etc. – groups corresponding to special users.

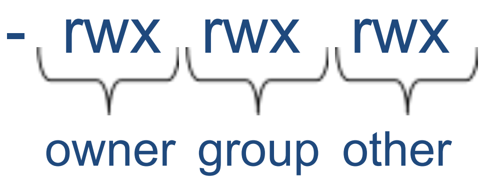
Unix File Access Control

- Each file entry in a directory is a pointer to a data structure called *inode*.

mode	Type of file and access rights
uid	User who owns the file
gid	Group which owns the file
atime	Access time
mtime	Modification time
itime	Inode alteration
Block count	Size of file (sort of)
	Pointer to physical location

Unix File Permission Bits

- File permissions obtained by *ls –l* command
- First character indicates type of file
  - ○ - plain file
  - ○ d directory
  - ○ c character device (tty or printer)
  - ○ b block device
  - ○ l symbolic link
  - ○ Etc



- rwx rwx rwx

owner group other

Unix File Permission Bits (continued)

- Next nine characters taken in groups of three indicate who can do what with the file
  - R – Permission to read
  - W – Permission to write
  - X – Permission to execute
- The three classes of permission correspond respectively to
  - Owner
  - Group
  - Other

$$-\underbrace{\text{rwx}}_{\text{owner}}\underbrace{\text{rwx}}_{\text{group}}\underbrace{\text{rwx}}_{\text{other}}$$

File Permission Bits – Special Cases

- File permission bits do not apply to symbolic links.
- If you have x access but no r access you can execute the program without reading it (not on Linux).
- Execute permission in a directory means you can list the files in a directory.
- What does denying this mean for security?
- File permission bits also commonly specified in octal notation.
  - 0777 means –rwxrwxrwx
  - 0600 means -rw-------, etc.

Question

- If owner has fewer privileges than other or group users:
  - What happens?
    - Owner gets access?
    - Owner does not?

Question

- If owner has fewer privileges than other or group users:
  - What happens?
    - Owner gets access?
    - Owner does not?
- Prioritized resolution of differences

```
if user = owner then owner permission
else if user in group then group permission
else other permission
```

Umask and Default Permissions

- *umask* (User file creation mode mask) is a four digit octal number used to determine file permissions for newly created files.
- It defines permission you do not want to be given (the bit-wise complement of the permission you want a file to have by default).
- Set up at time of log in, in environment variables
- 0002 – umask means 0775 permissions.
- 0077 umask means ?
- 0022 umask means ?

Process Operations and IDs

- Root
  - ID=0 for superuser - root; can access any file
- Fork and Exec
  - Inherit three IDs, except execution of file with setuid bit
- Setuid system calls
  - seteuid(newid) can let a process change it's effective UID!
- Details are actually more complicated
  - Several different calls: setuid, seteuid, setreuid

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
        - 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

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:
    - If Off: user has write permission on directory, can rename or remove files, even if not owner
    - If On: only file owner, directory owner, and root can rename or remove file in the directory

More on suid Bit

- Sometimes unprivileged users must perform tasks that are privileged.
  - Change user's shell thereby modify /etc/passwd
- UNIX allows certain programs to change UID to their owner when executed.
  - SUID programs – change UID to owner.
  - SGID programs – change GID to owners group.
- *ls –l* command indicates if SUID or SGID
  - -rwsr-xr-x indicates SUID
  - -rwxr-sr-x indicates SGID

Limitations of Unix File System

- Abbreviated ACL’s in general and UNIX in particular may not be flexible enough for many circumstances.
- Consider the following example:
  - 5 users: Anne, Beth, Cathy, Della and Elle.
  - Anne wants Beth to have read-only access.
  - She wants Cathy to write
  - Della to only read and write
  - Elle to only execute
  - Above not possible with Unix file permission bits!!

Augmenting Abbreviated ACL’s

- AIX uses extended permissions to augment base permissions.
  - attributes:
    - base permissions: owner (bishop): rw-
    - group (sys): r--
    - others: ---
  - extended permissions enabled users to:
    - specify rw- u:heberlei
    - permit -w- u:nelson, g=sys
    - permit rw- u:levitt
    - deny -w- u:heberlei, g=faculty

Other augmentations exist

- SELinux
- AppArmor
- getfacl, setfacl
  - https://www.golinuxcloud.com/setfacl-getfacl-command-in-linux/
- chown, etc. on Mac
  - http://www.techrepublic.com/blog/mac/introduction-to-os-x-access-control-lists-acls/1048
- seccomp
  - https://kubernetes.io/docs/tutorials/security/seccomp/

Unix Summary

- Advantages:
  - Some protection from most users
  - Flexible enough to make actions possible
- Drawbacks:
  - Too tempting to use root privileges
  - No way to assume some root privileges without all root privileges

    (At least with what is described here)

OS Mechanisms

- Multics
  - Ring structure
- Unix
  - File system, Setuid
- Windows
  - File system, Tokens, EFS
- Android
  - Apps are users, mediate communication

OS Mechanisms–Windows (NTFS+)

- Some basic functionality similar to Unix
  - Specify access for groups and users
    - Read, modify, change owner, delete, etc.
- Some additional concepts
  - Tokens
  - Security attributes
- Generally
  - More flexibility than Unix
    - Can define new permissions
    - Can give some but not all administrator privileges

NT Domains

- A domain is a set of computers with a central security authority
- PDC and the BDC (Backup) must be Windows NT.
- A domain can be set up to:
  - Ease viewing and access to resources.
  - Share a common user account database and security policy.
  - Enforce a common security stance across physical, divisional, or corporate boundaries.
  - Elimination of the need for every machine to provide its own authentication service.
- Users authenticated to the domain, can gain access to resources, such as printing, file sharing or applications, across all of the servers.

Access Control Lists

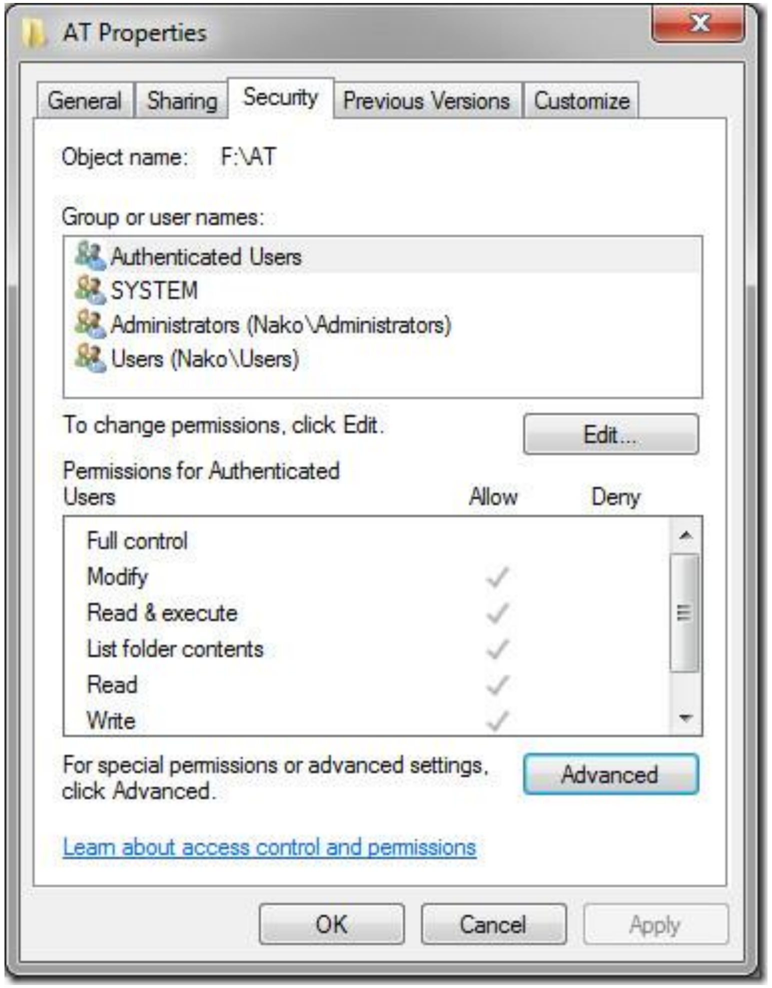
- Each object contains a security descriptor, which has
  - Security Identifier of the person who owns the object.
  - The regular ACL for access permissions.
  - The system ACL (SACL) which is used for auditing,
  - A group security identifier.

Access Control Entries

- ACL may be composed of Access Control Entries (ACE) which are composed of:
  - Basic permissions (six individual permissions)
    - Read (R), Write (W), Execute (X), Delete (D), Change Access Permissions (P), Take Ownership (O)
  - Standard permissions which are combinations derived from the basic permissions.
- ACE types:
  - Access-denied ACE - Used in ACLs to deny access rights
  - Access-allowed ACE - Used in ACLs to allow access rights
  - System-audit ACE - Used in SACLs to generate an audit record when the trustee attempts to exercise the specified access rights.

Sample Permission Options

- Security ID (SID)
  - Identity (replaces UID)
    - SID revision number
    - 48-bit authority value
    - variable number of Relative Identifiers (RIDs), for uniqueness
  - Users, groups, computers, domains, and domain members all have SIDs



AT Properties

Object name: F:\AT

Group or user names:

- Authenticated Users
- SYSTEM
- Administrators (Nako\Administrators)
- Users (Nako\Users)

To change permissions, click Edit.

Permissions for Authenticated Users	Allow	Deny
Full control		
Modify	✓	
Read & execute	✓	
List folder contents	✓	
Read	✓	
Write	✓	

For special permissions or advanced settings, click Advanced.

Learn about access control and permissions

Permission Inheritance

- Static permission inheritance (Win NT)
    - Initially, subfolders inherit permissions of folder
    - Folders and subfolders are changed independently
    - Replace Permissions on Subdirectories command
        - Eliminates any differences in permissions
- Dynamic permission inheritance (Win 2000)
    - Child inherits parent permission, remains linked
    - Parent changes are inherited, except for explicit settings
    - Inherited and explicitly-set permissions may conflict
        - Resolution rules
            - Positive permissions are additive
            - Negative permission (deny access) takes priority

Tokens

- Security Reference Monitor
  - uses tokens to identify the security context of a process or thread
- Security context
  - privileges, accounts, and groups associated with the process or thread
- Impersonation token
  - thread can adopt a different security context, usually of another user

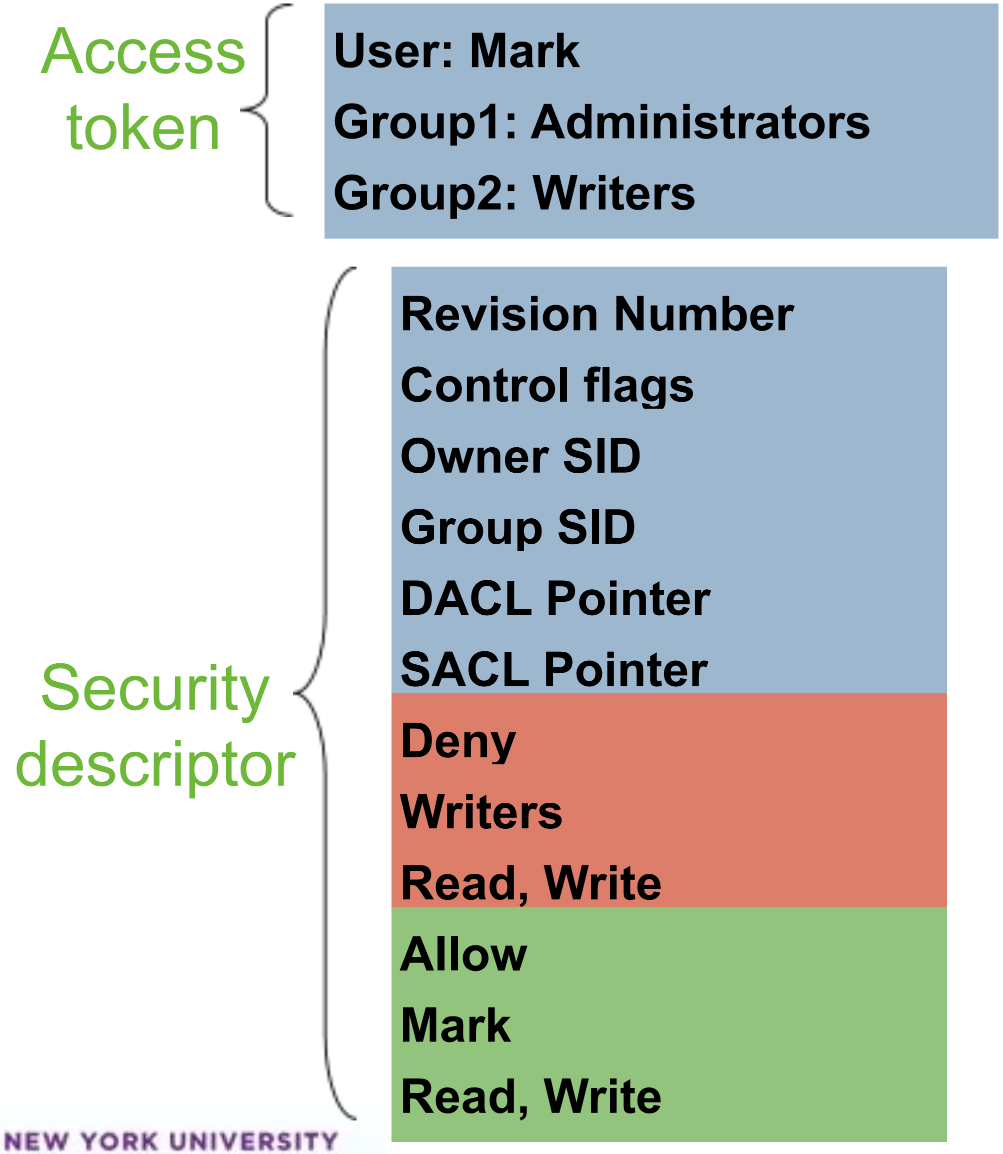
Impersonation Tokens

- Process uses security attributes of another
  - Client passes impersonation token to server
- Client specifies impersonation level of server
  - Anonymous
    - Token has no information about the client
  - Identification
    - server obtains the SIDs of client and client's privileges, but server cannot impersonate the client
  - Impersonation
    - server identifies and impersonates the client
  - Delegation
    - lets server impersonate client on local, remote systems

Security Descriptor

- Information associated with an object:
  - Specifies who can perform actions and what actions they can perform on an object
- Several fields
  - SIDs for the owner and primary group of an object
  - A Discretionary Access Control List (DACL)
    - access rights allowed or denied to users or groups
  - A System Access Control List (SACL)
    - types of access attempts that generate audit records for the object.
  - A set of control bits that qualify the meaning of a security descriptor or its individual members.

Example Access Request



Access token

User: Mark

Group1: Administrators

Group2: Writers

Security descriptor

Revision Number

Control flags

Owner SID

Group SID

DACL Pointer

SACL Pointer

Deny

Writers

Read, Write

Allow

Mark

Read, Write

Access request: write
Action: denied

⊗

- User Mark requests write permission
- Descriptor denies permission to group
- Reference Monitor denies request

(DACL for access, SACL for audit and logging)

Priority:
- Explicit Deny
- Explicit Allow
- Inherited Deny
- Inherited Allow

Notes from Alex Sotirov's Windows talk

- Permissions are harder to track because you have to look at lots of files and each examination is a pain.
- A folder's permissions can trickle down onto contained files.
- Executable needs to be protected.
- Libraries need to be protected.
- Configuration file (registry) can be an issue.
- Threads are securable (can suspend, examine registers, modify registers including IP, resume).

Evolution of Windows Permissions

Windows 2000:

Introduction of Active directory:

- Centralized User and Group Management
- User accounts and groups can be organized hierarchically within the directory structure
- Active Directory authenticates the user's credentials
- Each file and folder has Access Control List (ACL) saying which users or groups have permissions
- Enables the use of Group Policy Objects (GPOs)
- Allows administrators to delegate specific administrative tasks

Evolution of Windows Permissions

Windows XP(SP2) and Server 2003:

Enhanced Group Policy:

- Introduced Windows Firewall - centrally configured and managed through Group Policy.
- Group Policy used to deploy, update, and remove software applications from client computers
- Group Policy could be used to deploy security templates
- Implement restrictions on the use of removable devices like USB drives, control over Windows services

Evolution of Windows Permissions

Windows Vista and Server 2008:

User Account Control (UAC):

- Mitigate the risks associated with running applications with admin privileges by default
- Prompt for consent when an action requires admin privileges
- Requires credentials for admin approval
- Standard user mode is default (Not admin user mode!)

Mandatory Integrity Control (MIC):

- Enhances security by enforcing mandatory access controls based on the integrity levels of objects
- Objects integrity levels - low, medium, high, and system
- Restricts the interaction between objects of different integrity levels

Evolution of Windows Permissions

Windows 7:

Applocker:

- Used to control which applications and scripts can be run on Windows
- Operates on a whitelist model
- Rules can be defined based on file paths, file types
- Allows you to create rules based on the digital signature of an application (Publisher rules)

Windows 8 and Server 2012:

- Dynamic Access Control: admins set up permissions based on claims

Example: Access to files could be restricted based on user properties like department or title.

Evolution of Windows Permissions

Windows 10, Server 2016/2019:

Windows Information Protection (WIP):

- Separates and encrypts work-related data from personal data
- Organizations define policies determining how work data is handled
- Integrated with Azure AD Conditional Access - administrators enforce access policies based on the classification of data.

Azure Active Directory Join:

- Device registered and managed through Azure AD- Microsoft's cloud-based identity and access management service.
- Manage user identities and device access from the cloud
- Single sign-on capabilities across Azure AD-integrated services and apps
- Admins perform device management tasks.
- Valuable in scenarios where organizations support remote work

Windows Summary

- Advantages:
  - Tokens provide contextual information
  - More flexible than Unix
- Drawbacks:
  - Poor implementation of tokens in APIs (historically, many just use identification)
  - Complex for users / developers

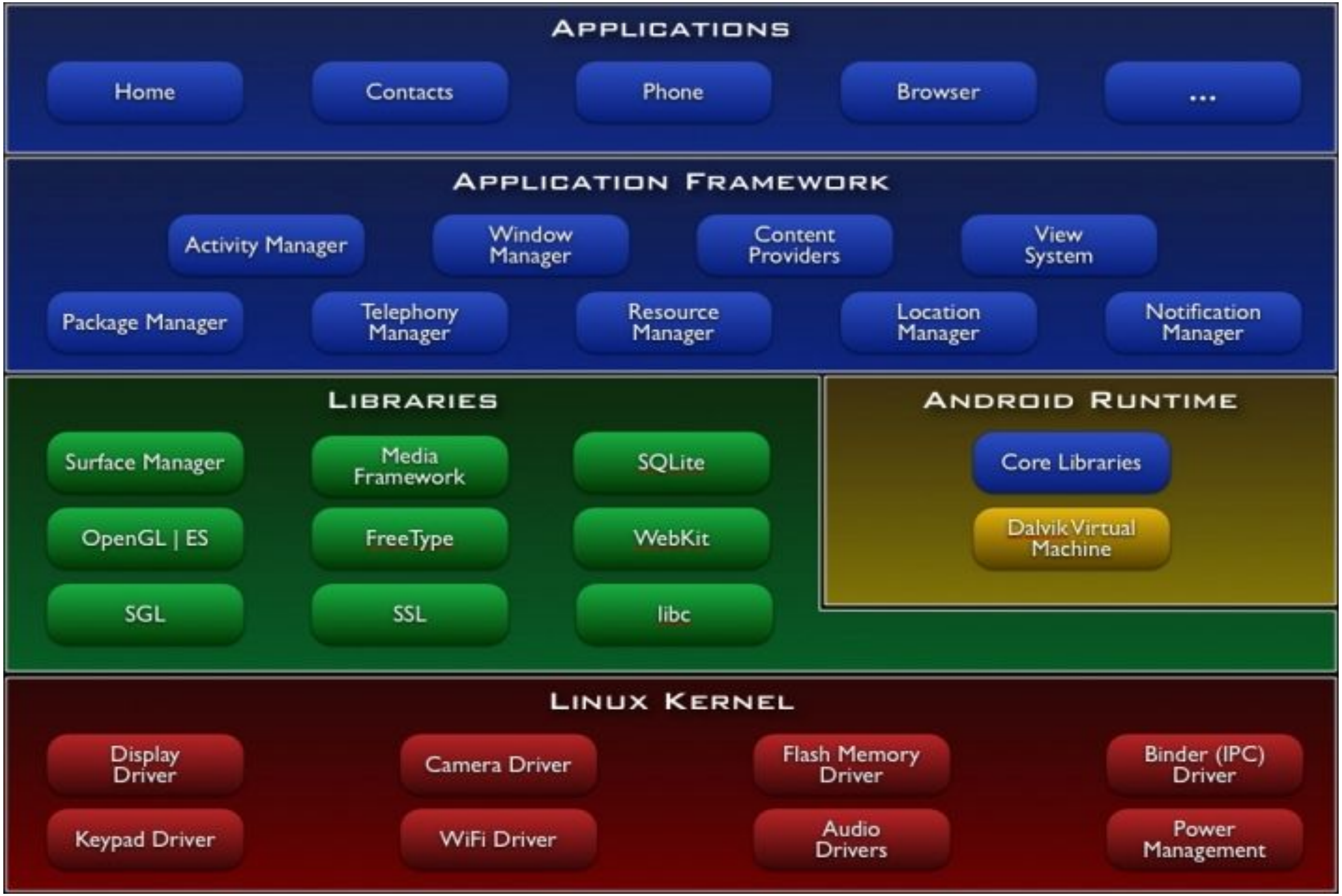
OS Mechanisms

- Multics
  - Ring structure
- Unix
  - File system, Setuid
- Windows
  - File system, Tokens, EFS
- Android
  - Apps are users, mediate communication

Android Security Model

- OS user-isolation applied to applications
- Permission restrictions focused on inter-component (application) communications

Android Architecture



Android Challenges

- Battery life
    - Developers must conserve power
    - Applications store the state, thus they can be stopped in order to save power and then restarted – helps with DoS
    - Most foreground activity is never killed
- Android market
    - Not reviewed by Google (different from Apple)
    - No way of stopping bad applications from showing up on market
    - Malware writers may be able to get code onto platform: shifts focus from remote exploit to privilege escalation

Application Development Concepts

- Activity – one-user task
  - Example: scroll through your inbox
  - Email client comprises many activities
- Service – Java daemon that runs in background
  - Example: application that streams an mp3 in background
- Intent – asynchronous messaging system
  - Fire an intent to switch from one activity to another
  - Example: email app has inbox, compose activity, viewer activity
    - User clicks on inbox entry, fires an intent to the viewer activity, which then allows the user to view the email
- Content provider
  - Store and share data using a relational database interface
- Broadcast receiver
  - "mailboxes" for messages from other applications

Exploit Prevention

- 100 open source libraries + 500 million lines new code
  - Open source -> no obscurity
- Goals
  - Prevent remote attacks
  - Secure drivers, media codecs, new and custom features
- Overflow prevention
  - Some stack and heap protection
- Decided against (in initial release)
  - stack and heap non-execute protections (due to time-to-market constraints and battery life constraints)
  - ASLR – performance impact
    - Many pre-linked images for performance
    - Can't install different images on different devices in the factory

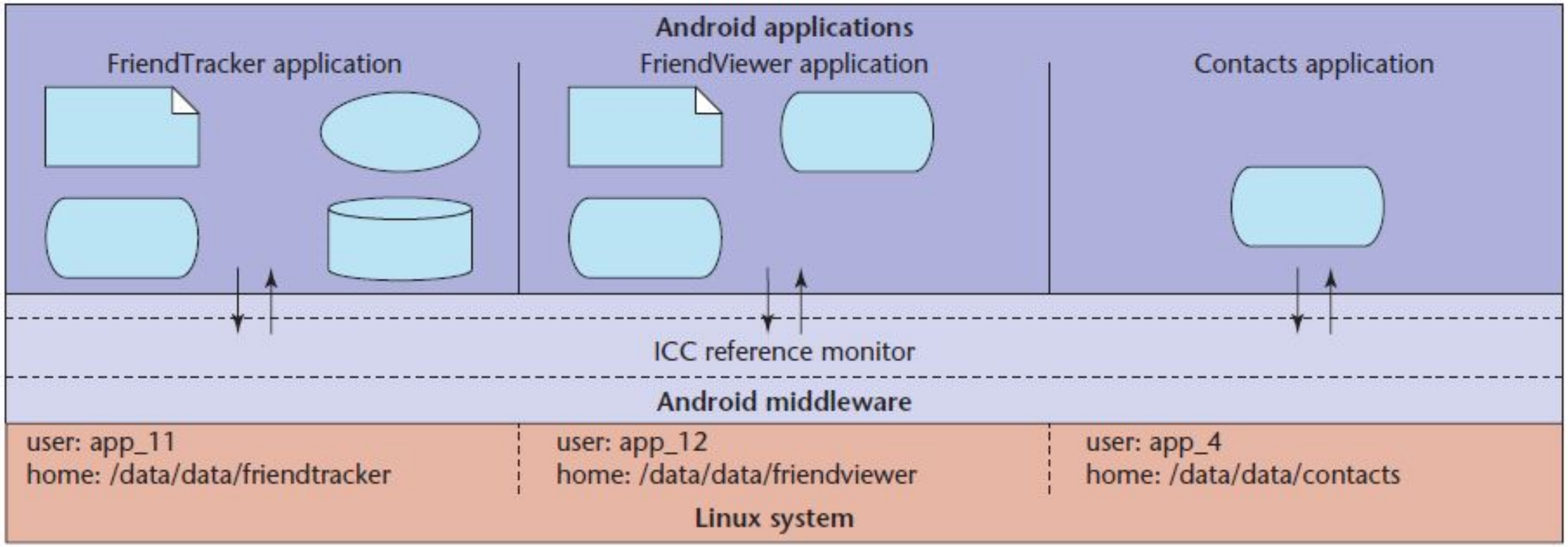
We will discuss many of these topics later

Application Sandbox

- Application sandbox
  - Each application runs with its UID in its own Dalvik virtual machine
    - Provides CPU protection, memory protection
    - Authenticated communication protection using Unix domain sockets
    - Only ping, zygote* - run as root
- Applications announce permission requirement
  - Create a whitelist model – user grants access
    - But don’t want to ask user often – all questions used to be asked at install time!!!
  - Inter-component communication reference monitor checks permission

*Note: spawns another process

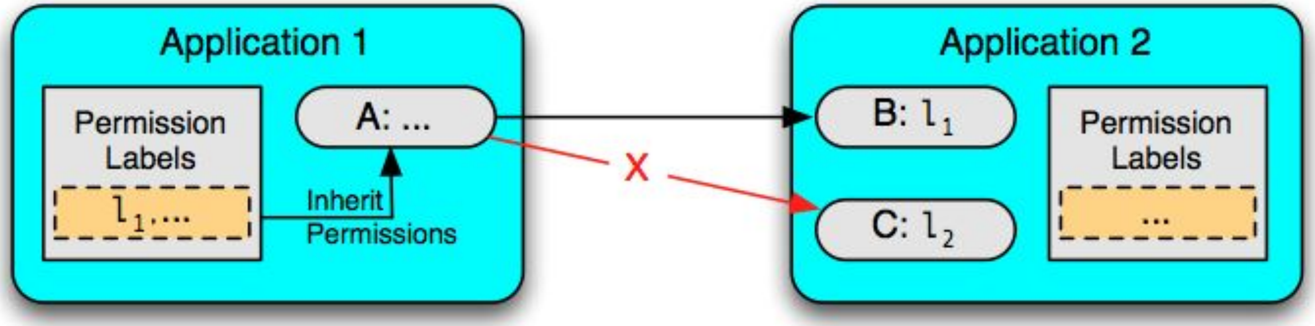
Application Sandbox



- Layers of security
  - Each application executes as its own user identity
  - Android middleware has reference monitor that mediates the establishment of inter-component communication (ICC)

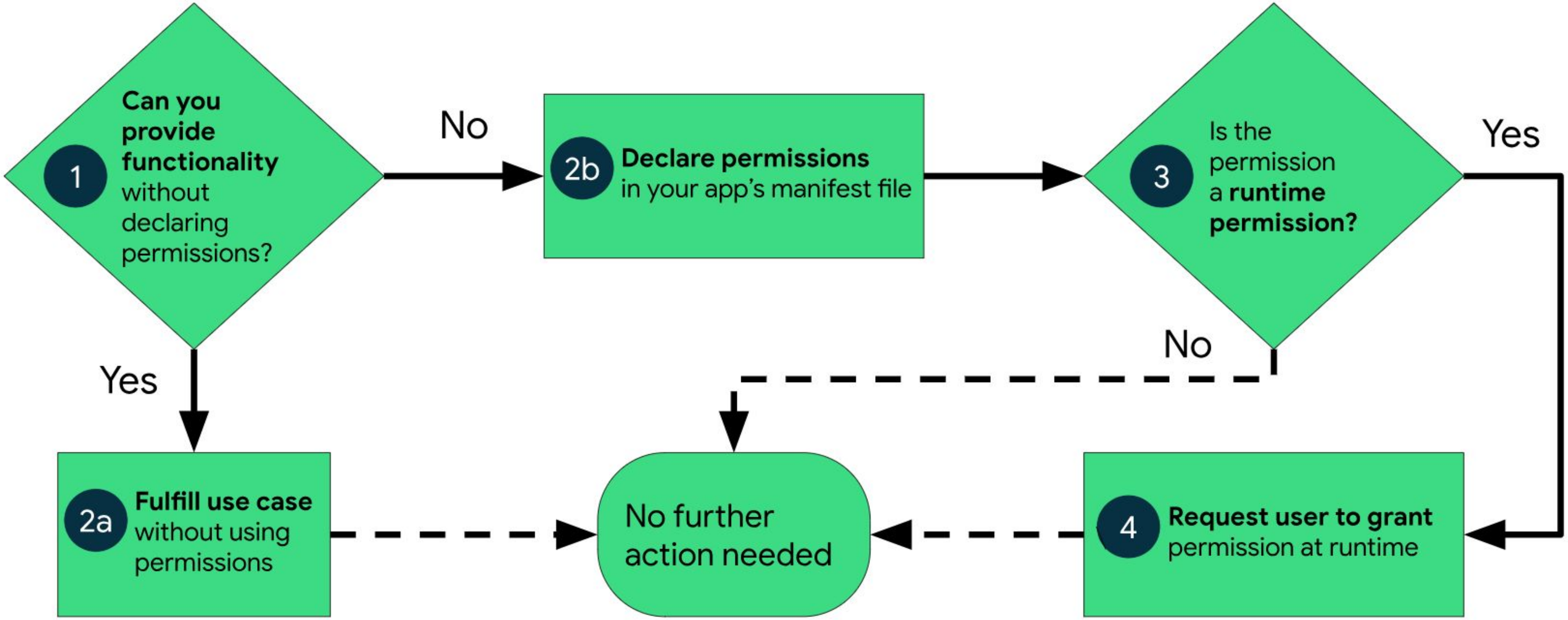
Android Security Model

- The Android manifest file allows developers to define an access control policy for access to components
  - Each component can be assigned an access permission label
  - Each application requests a list of permission labels
  - For Android 5.1.1 and below, fixed at install time
  - For Android 6.0 and higher, user sees a system dialogue to either allow or deny access when the application requests for access during runtime
    - User can change permissions one-by-one in system settings



Android Security Model 6.0+

- Over 200 Manifest permissions
- Android 10 - Permissions required to access location in background (Granted implicitly before)
- Android 11- Introduction of one time permissions, Permissions auto-reset after a certain amount of time


1 Can you provide functionality without declaring permissions?
No
2b Declare permissions in your app's manifest file
3 Is the permission a runtime permission?
Yes
Yes
No
2a Fulfill use case without using permissions
No further action needed
4 Request user to grant permission at runtime

Android Summary

- Advantages
    - Sandboxes applications, not "users"
    - Focuses on more than just 'allow / disallow'
- Drawbacks
    - (used to be) Main access control settings via a dialog box at install time
    - Outdated versions of software
    - Lots of trusted (?) library code

Assignment 2, Part 2

Attack Reference Monitors

- Goal: Better understand security mechanisms
- Task: Attack other students' Assignment 2, Part 1 reference monitors

- Create **multiple** test cases
  - One test per issue
  - **Must output only when a bug is found**
    - Accuracy bug
    - Security bug
  - If your test is wrong, you will lose points!
- Refer to the instructions on the webpage:

https://github.com/SeattleTestbed/docs/blob/master/EducationalAssignments/DefaultPartTwo.md

Assignment 2, Part 2 - Rubric

- +40 having at least one non discarded attack file in the correct format
- +60 determined by number of students whose security layer your attack code bypasses
- -5 per attack file of yours which is discarded
- -3 per attack program of another student which you incorrectly flag as wrong
- minimum grade of 0

Reading For Next Week

Read about NVD, CVE
https://www.kaseya.com/blog/national-vulnerability-database-nvd/
Browser sandboxing:
https://www.browserstack.com/guide/what-is-browser-sandboxing
Read about capability-based systems:
http://www.cs.washington.edu/homes/levy/capabook/Chapter1.pdf
How to use linux command line:
https://www.freecodecamp.org/news/linux-command-line-tutorial/
Linux directory structure:
https://www.geeksforgeeks.org/linux-directory-structure/
Review BLP, Biba, etc.
(We will need these next lecture)

Get your exams if you arrived late

Meet the TA if you didn’t take the exam.