CS 155 Spring 2010

Access Control and Operating System Security

John Mitchell

Lecture goal: Cover background and concepts used in Android security model

IEEE Security and Privacy, Jan.-Feb. 2009

2

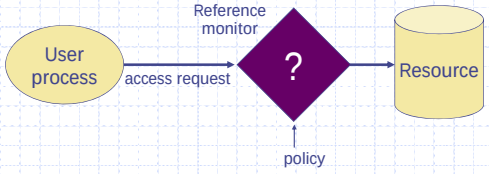
Outline

- Access Control Concepts
  - Matrix, ACL, Capabilities
- OS Mechanisms
  - Multics
    - Ring structure
  - Amoeba
    - Distributed, capabilities
  - Unix
    - File system, Setuid
  - Windows
    - File system, Tokens, EFS
- Web browser (briefly)
  - "OS of the future"
  - Protect content based on origins instead of user id
- Android security model
  - OS user-isolation applied to applications
  - Reference monitor for inter-component communications

3

Access control

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



4

Access control matrix [Lampson]

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

5

Two implementation concepts

- 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 Kerberos, ...

6

Capabilities

- Operating system concept
  - "... of the future and always will be ..."
- Examples
  - Dennis and van Horn, MIT PDP-1 Timesharing
  - Hydra, StarOS, Intel iAPX 432, Eros, ...
  - Amoeba: distributed, unforgeable tickets
- References
  - Henry Levy, Capability-based Computer Systems
    http://www.cs.washington.edu/homes/levy/capabook/
  - Tanenbaum, Amoeba papers

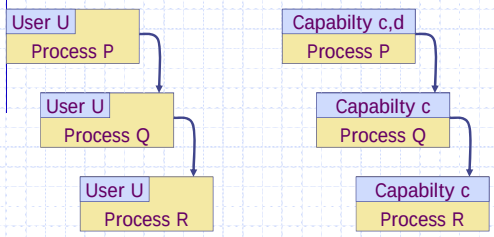
7

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

8

ACL vs Capabilities



9

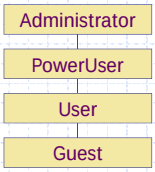
ACL vs Capabilities

- 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

10

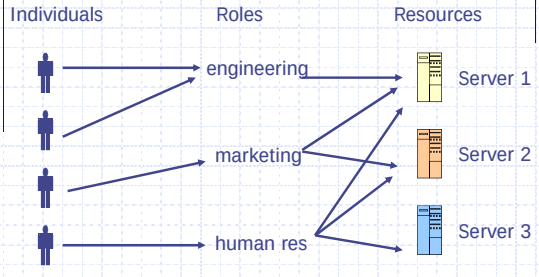
Roles (also called Groups)

- Role = set of users
  - Administrator, PowerUser, User, Guest
  - Assign permissions to roles; each user gets permission
- Role hierarchy
  - Partial order of roles
  - Each role gets permissions of roles below
  - List only new permissions given to each role



11

Role-Based Access Control



Advantage: user's change more frequently than roles

12

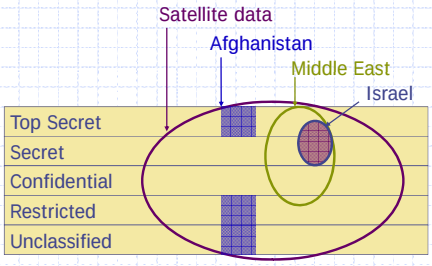
Multi-Level Security (MLS) Concepts

- Military security policy
  - Classification involves sensitivity levels, compartments
  - Do not let classified information leak to unclassified files
- Group individuals and resources
  - Use some form of hierarchy to organize policy
- Other policy concepts
  - Separation of duty
  - "Chinese Wall" Policy

13

Military security policy

- Sensitivity levels
- Compartments



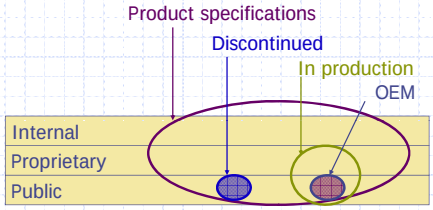
14

Military security policy

- Classification of personnel and data
  - Class = ⟨rank, compartment⟩
- Dominance relation
  - $D_1 \leq D_2$ iff $rank_1 \leq rank_2$ and $compartment_1 \subseteq compartment_2$
  - Example: ⟨Restricted, Israel⟩ ≤ ⟨Secret, Middle East⟩
- Applies to
  - Subjects – users or processes
  - Objects – documents or resources

15

Commercial version



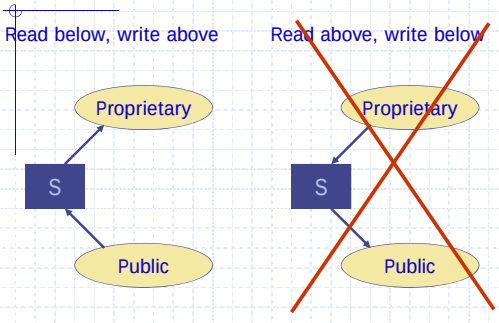
16

Bell-LaPadula Confidentiality Model

- When is it OK to release information?
- Two Properties (with silly names)
  - Simple security property
    - A subject S may read object O only if C(O) ≤ C(S)
  - *-Property
    - A subject S with read access to O may write object P only if C(O) ≤ C(P)
- In words,
  - You may only *read below* your classification and only *write above* your classification

17

Picture: Confidentiality



18

Biba Integrity Model

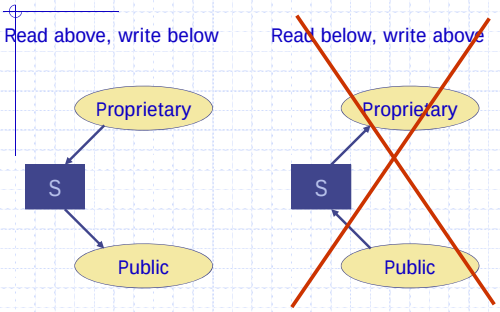
- Rules that preserve integrity of information
- Two Properties (with silly names)
  - Simple integrity property
    - A subject S may write object O only if C(S) ≥ C(O)

    (Only trust S to modify O if S has higher rank ...)
  - *-Property
    - A subject S with read access to O may write object P only if C(O) ≥ C(P)

    (Only move info from O to P if O is more trusted than P)
- In words,
  - You may only *write below* your classification and only *read above* your classification

19

Picture: Integrity



20

Other policy concepts

- Separation of duty
  - If amount is over $10,000, check is only valid if signed by two authorized people
  - Two people must be *different*
  - Policy involves role membership and ≠
- Chinese Wall Policy
  - Lawyers L1, L2 in same firm
  - If company C1 sues C2,
    - L1 and L2 can each work for either C1 or C2
    - No lawyer can work for opposite sides in any case
  - Permission depends on use of other permissions

These policies cannot be represented using access matrix

21

Example OS Mechanisms

- Multics
- Amoeba
- Unix
- Windows

22

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

http://www.multicians.org/security.html

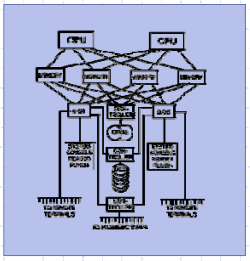
E.I. Organick, The Multics System: An Examination of Its Structure, MIT Press, 1972

23

Multics time period

F.J. Corbato

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



24

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 lang
- 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

25

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

26

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"

27

Amoeba

Server port	Obj #	Rights	Check field

- Distributed system
  - Multiple processors, connected by network
  - Process on A can start a new process on B
  - Location of processes designed to be transparent
- Capability-based system
  - Each object resides on server
  - Invoke operation through message to server
    - Send message with capability and parameters
    - Sever uses object # to indentify object
    - Sever checks rights field to see if operation is allowed
    - Check field prevents processes from forging capabilities

28

Capabilities

Server port	Obj #	Rights	Check field

- Owner capability
  - When server creates object, returns owner cap.
    - All rights bits are set to 1 (= allow operation)
    - Check field contains 48-bit rand number stored by server
- Derived capability
  - Owner can set some rights bits to 0
  - Calculate new check field
    - XOR rights field with random number from check field
    - Apply one-way function to calculate new check field
  - Server can verify rights and check field
    - Without owner capability, cannot forge derived capability

Protection by user-process at server; no special OS support needed

29

Unix file security

- 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

30

Question

- Owner can have fewer privileges than other
  - 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

31

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

32

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 calls
  - seteuid(newid) can set EUID to
    - Real ID or saved ID, regardless of current EUID
    - Any ID, if EUID=0
- Details are actually more complicated
  - Several different calls: setuid, seteuid, setreuid

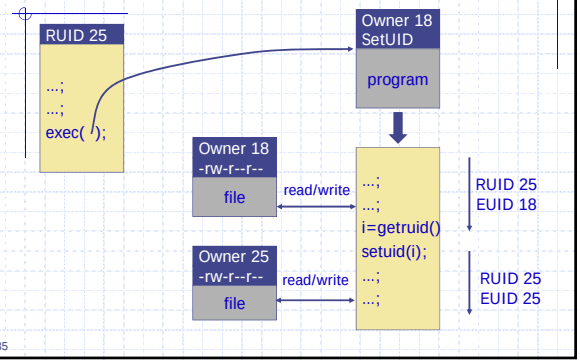
33

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

34

Example



35

Setuid programming

- Be Careful!
  - Root can do anything; don' t get tricked
  - Principle of least privilege – change EUID when root privileges no longer needed
- Setuid scripts
  - This is a bad idea
  - Historically, race conditions
    - Begin executing setuid program; change contents of program before it loads and is executed

36

Unix summary

- Good things
  - Some protection from most users
  - Flexible enough to make things possible
- Main bad thing
  - Too tempting to use root privileges
  - No way to assume some root privileges without all root privileges

37

Access control in Windows (NTFS)

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

38

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, domain members all have SIDs

39

Permission Inheritance

- Static permission inheritance (Win NT)
  - Initially, subfolders inherit permissions of folder
  - Folder, subfolder 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 explicit settings
  - Inherited and explicitly-set permissions may conflict
    - Resolution rules
      - Positive permissions are additive
      - Negative permission (deny access) takes priority

40

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 uses temporarily to adopt a different security context, usually of another user

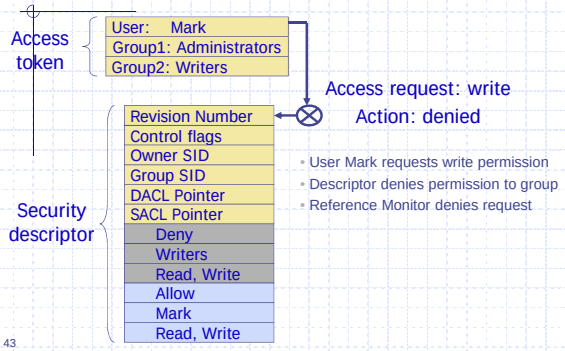
41

Security Descriptor

- Information associated with an object
  - who can perform what actions on the object
- Several fields
  - Header
    - Descriptor revision number
    - Control flags, attributes of the descriptor
      - E.g., memory layout of the descriptor
  - SID of the object's owner
  - SID of the primary group of the object
  - Two attached optional lists:
    - Discretionary Access Control List (DACL) – users, groups, ...
    - System Access Control List (SACL) – system logs, ..

42

Example access request



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

43

Impersonation Tokens (=setuid?)

- 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 obtain the SIDs of client and client's privileges, but server cannot impersonate the client
  - Impersonation
    - server identify and impersonate the client
  - Delegation
    - lets server impersonate client on local, remote systems

44

SELinux Security Policy Abstractions

- Type enforcement
  - Each process has an associated domain
  - Each object has an associated type
  - Configuration files specify
    - How domains are allowed to access types
    - Allowable interactions and transitions between domains
- Role-based access control
  - Each process has an associated role
    - Separate system and user processes
  - Configuration files specify
    - Set of domains that may be entered by each role

45

An Analogy

Operating system

- Primitives
  - System calls
  - Processes
  - Disk
- Principals: Users
  - Discretionary access control
- Vulnerabilities
  - Buffer overflow
  - Root exploit

Web browser

- Primitives
  - Document object model
  - Frames
  - Cookies / localStorage
- Principals: “Origins”
  - Mandatory access control
- Vulnerabilities
  - Cross-site scripting
  - Universal scripting

46

Components of browser security policy

- Frame-Frame relationships
  - canScript(A,B)
    - Can Frame A execute a script that manipulates arbitrary/nontrivial DOM elements of Frame B?
  - canNavigate(A,B)
    - Can Frame A change the origin of content for Frame B?
- Frame-principal relationships
  - readCookie(A,S), writeCookie(A,S)
    - Can Frame A read/write cookies from site S?

47

Principles of secure design

- Compartmentalization
  - Principle of least privilege
  - Minimize trust relationships
- Defense in depth
  - Use more than one security mechanism
  - Secure the weakest link
  - Fail securely
- Keep it simple
- Consult experts
  - Don’t build what you can easily borrow/steal
  - Open review is effective and informative

48

Compartmentalization

- Divide system into modules
  - Each module serves a specific purpose
  - Assign different access rights to different modules
    - Read/write access to files
    - Read user or network input
    - Execute privileged instructions (e.g., Unix root)
- Principle of least privilege
  - Give each module only the rights it needs

49

This slide borrowed from Selley, Shinde, et al., Vulnerability Study of the Android

Android security resources

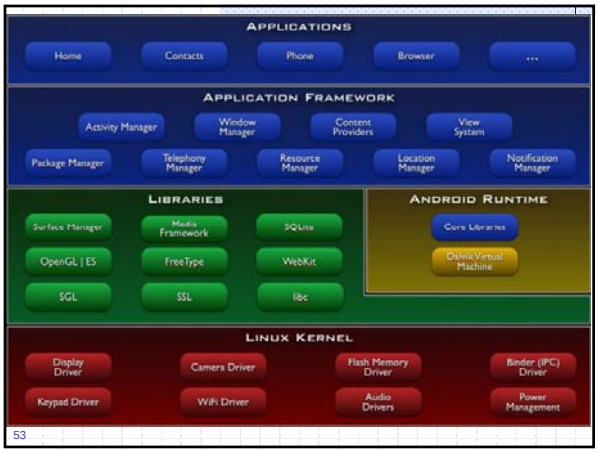
- Cannings talk (Android team)
  - http://www.usenix.org/events/sec09/tech/
- Understanding Android Security (PennState):
  - http://ieeexplore.ieee.org/search/srchabstract.jsp?tp=&arnumber=4768655 (posted on CourseWare, Stanford-only)
- Earlier tutorial at CCS (PennState)
  - http://siis.cse.psu.edu/android_sec_tutorial.html
- Unlock flaw
  - Can unlock phone using back button, when called
  - http://www.youtube.com/watch?v=CcOz1yZ5cj8

51

Android

- Open-source platform (Open Handset Alliance)
- Native development, Java development
- Phones carried by 32+ carriers, 20+ countries
- Platform outline:
  - Linux kernel
  - Webkit- based browser
  - SQL-lite
  - Open SSL, Bouncy Castle crypto API and Java library
  - Bionic LibC (small code, good performance, no GPL)
  - Apache Harmony libraries (open source Java impl)
  - Many others: video stuff, Bluetooth, vibrate phone, etc.

52



53

Android challenges

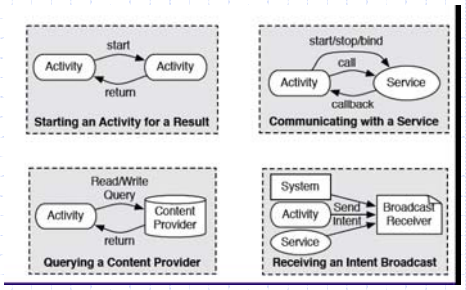
- Battery life
  - Developers must conserve power
  - Applications store state so they can be stopped (to save power) and restarted – helps with DoS
  - Most foreground activity is never killed
- Android market
  - Not reviewed by Google (diff from Apple)
  - No way of stopping bad applications from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

54

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
- Intents – asynchronous messaging system
  - Fire an intent to switch from one activity to another
  - Example: email app has inbox, compose activity, viewer activity
    - User click on inbox entry fires an intent to the viewer activity, which then allows user to view that email
- Content provider
  - Store and share data using a relational database interface
- Broadcast receiver
  - "mailboxes" for messages from other applications

55



Source: Penn State group, Android security tutorial

56

Signing

- Developers sign applications
  - Self-signed certificates
    - Not form of identity
    - Used to allow developer who built application to update application
    - Based on Java key tools and jar signer

57

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
  - ProPolice stack protection (like Dan's last lecture)
    - First on the ARM architecture; some nasty gcc bugs …
  - Some heap overflow protections
    - Chunk consolidation in DL malloc (from OpenBSD)
- Decided against (in initial release)
  - stack and heap non-execute protections (time-to-market, battery life)
  - ASLR – performance impact
    - Many pre-linked images for performance
    - Can't install different images on different devices in the factory
    - Later developed and contributed by Bojinov, Boneh (Stanford)

58

dlmalloc (Doug Lea)

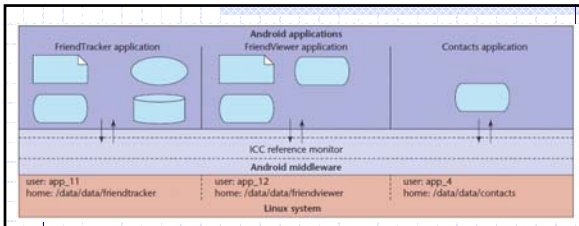
- Stores meta data in band
- Heap consolidation attack
  - Heap overflow can overwrite pointers to previous and next unconsolidated chunks
  - Overwriting these pointers allows remote code execution
- Change to improve security
  - Check integrity of forward and backward pointers
    - Simply check that back-forward-back = back, f-b-f=f
  - Team believes this increases the difficulty of heap overflow

59

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 (spawn another process) run as root
  - Applications announces permission requirement
    - Create a whitelist model – user grants access
      - But don't want to ask user often – all questions asked as install time
    - Inter-component communication reference monitor checks permissions

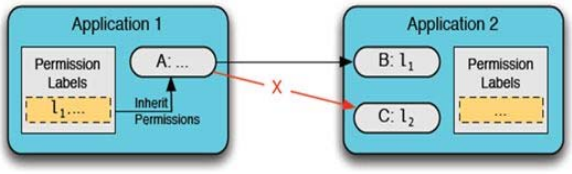
60



- Two forms of security enforcement
  - Each application executes as its own user identity
  - Android middleware has reference monitor that mediates the establishment of inter-component communication (ICC)
  - First is straightforward to implement, second requires careful consideration of mechanism, policy

61

Source: Penn State group Android security paper



MAC Policy Enforcement in Android. This is how applications access components of other applications via the reference monitor. Component A can access components B and C if permission labels of application 1 are equal or dominate labels of application 2.

62

Source: Penn State group, Android security tutorial

Android manifest

- Manifest files describing the contents of an application *package*
  - Each Android application has AndroidManifest.xml file
  - describes the contained components
- Components cannot execute unless they are listed
  - specifies rules for "auto-resolution"
  - specifies access rules
  - describes runtime dependencies
  - optional runtime libraries
  - required system permissions

63

Permission categories

- Permissions can be:
  - normal - always granted
  - dangerous - requires user approval
  - signature - matching signature key
  - signature or system - same as signature, but also system apps

64

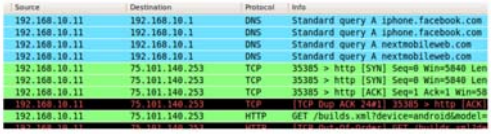
Users are always careful about downloads

65

Source: Jon Oberheide, CanSecWest presentation

Permission granularity

- fBook app example
  - Asks for permission to access network
  - User grants this assuming network is used to reach Facebook
  - Also "phones home" to another site



66

Source: Jon Oberheide, CanSecWest presentation

Additional issues

Intent Broadcast Permissions

- Addition to basic model
  - Code broadcasting Intent set access permission restricting Broadcast Receivers access the Intent
- Why: Define applications to read broadcasts
  - e.g., FRIEND_NEAR msg in PennState example
- Caution
  - If no permission label is set on a broadcast, any unprivileged application can read it.
- Recommendation
  - Always specify an access permission on Intent broadcasts (unless explicit destination).

67

Summary

- Access Control Concepts
  - Matrix, ACL, Capabilities
- OS Mechanisms
  - Multics
    - Ring structure
  - Amoeba
    - Distributed, capabilities
  - Unix
    - File system, Setuid
  - Windows
    - File system, Tokens, EFS
- Web browser (briefly)
  - "OS of the future"
  - Protect content based on origins instead of user id
- Android security model
  - OS user-isolation applied to applications
  - Reference monitor for inter-component communications
  - Starts out seeming simple but gets more complicated...

68

69