DAC vs. MAC

- **Most people familiar with discretionary access control (DAC)**
  - Example: Unix user-group-other permission bits
  - Might set a file `private` so only group `friends` can read it
- **Discretionary means anyone with access can propagate information:**
  - `Mail sigint@enemy.gov < private`
- **Mandatory access control**
  - Security administrator can restrict propagation
  - Abbreviated MAC (NOT a message authentication code)

Bell-Lapadula model

- **View the system as subjects accessing objects**
  - The system input is requests, the output is decisions
  - Objects can be organized in one or more hierarchies, H (a tree enforcing the type of decendents)
- **Four modes of access are possible:**
  - execute – no observation or alteration
  - read – observation
  - append – alteration
  - write – both observation and modification
- **The current access set, b, is (subj, obj, attr) tripples**
  - E.g., (user `dm`, file `grates.txt`, `r`) if I'm currently reading file
- **An access matrix M encodes permissible access types (subjects are rows, objects columns)**

Security levels

- **A *security level* is a (c, s) pair:**
  - $c =$ classification – E.g., unclassified, secret, top secret
  - $s =$ category-set – E.g., Nuclear, Crypto
- (c_1, s_1) ***dominates*** (c_2, s_2) **iff** $c_1 \geq c_2$ **and** $s_2 \subseteq s_1$
- **Subjects and objects are assigned security levels**
  - level_S(S), level_O(O) – security level of subject/object
  - current-level(S) – subject may operate at lower level
  - $f = (\text{level}_\text{S}, \text{level}_\text{O}, \text{current-level})$ – set of 3 level functions
- **State of system is 4-tuple** (b, M, f, H)

Security properties

- **The simple security or *ss-property*:**
  - For any $(S, O, A) \in b$, if A includes observation, then level(S) must dominate level(O)
  - E.g., an unclassified user cannot read a top-secret document
- **The star security or **-property*:**
  - If a subject can observe O_1 and modify O_2, then level(O_2) dominates level(O_1)
  - E.g., cannot copy top secret file into secret file
  - More precisely, given $(S, O, A) \in b$:
    if $A = r$: level(O) is dominated by current-level(S)
    if $A = a$: level(O) dominates current-level(S)
    if $A = w$: level(O) = current-level(S)

Straw man MAC implementation

- **Take an ordinary Unix system**
- **Put labels on all files and directories**
- **Each user has a security level**
- **Determine current security level dynamically**
  - When user logs in, start with lowest curent-level
  - Increase current-level as higher-level files are observed
  - If user's level does not dominate current, kill program
  - If program writes to file it doesn't dominate, kill it
- **Is this secure?**

No: Covert channels

- **System rife with *storage channels***
  - Low current-level process executes another program
  - New program reads sensitive file, gets high current-level
  - High program exploits covert channels to pass data to low
- **E.g., High program inherits file descriptor**
  - Can pass 4-bytes of information to low prog. in file offset
- **Other storage channels:**
  - Exit value, signals, terminal escape codes, . . .
- **If we eliminate storage channels, is system secure?**

No: Timing channels

- **Example: CPU utilization**
  - To send a 0 bit, use 100% of CPU is busy-loop
  - To send a 1 bit, sleep and relinquish CPU
  - Repeat to transfer more bits
- **Example: Resource exhaustion**
  - High prog. allocate all physical memory if bit is 1
  - Low program tries to allocate memory; if it fails, bit is 1
- **More examples: Disk head position, processor cache/TLB polution, ...**

An approach to eliminating covert channels

- **Observation: Covert channels come from sharing**
  - If you have no shared resources, no covert channels
  - Extreme example: Just use two computers
- **Problem: Sharing needed**
  - E.g., read unclassified data when preparing classified
- **Approach: Strict partitioning of resources**
  - Strictly partition and schedule resources between levels
  - Occasionally reapportion resources based on usage
  - Do so infrequently to bound leaked information
  - In general, only hope to bound bandwidth covert channels
  - Approach still not so good if many security levels possible

Declassification

- **Sometimes need to prepare unclassified report from classified data**
- **Declassification happens outside of system**
  - Present file to security officer for downgrade
- **Job of declassification often not trivial**
  - E.g., Microsoft word saves a lot of undo information
  - This might be all the secret stuff you cut from document

Biba integrity model

- **Problem: How to protect integrity**
  - Suppose text editor gets trojaned, subtly modifies files, might mess up attack plans
- **Observation: Integrity is the converse of secrecy**
  - In secrecy, want to avoid writing less secret files
  - In integrity, want to avoid writing higher-integrity files
- **Use integrity hierarchy parallel to secrecy one**
  - Now only most privilegted users can operate at lowest integrity level
  - If you read less authentic data, your current integrity level gets raised, and you can no longer write low files

DoD Orange book

- **DoD requirements for certification of secure systems**
- **4 Divisions:**
  - D – been through certification and not secure
  - C – discretionary access control
  - B – mandatory access control
  - A – like B, but better verified design
  - Classes within divisions increasing level of security

Divisions C and D

- **Level D: Certifiably insecure**
- **Level C1: Discretionary security protection**
  - Need some DAC mechanism (user/group/other, ACLs, etc.)
  - TCB needs protection (e.g., virtual memory protection)
- **Level C2: Controlled access protection**
  - Finer-graunlarity access control
  - Need to clear memory/storage before reuse
  - Need audit facilities
- **Many OSes have C2-security packages**
  - Was C2 Solaris "more secure" than normal Solaris?

Division B

- **B1 - Labeled Security Protection**
  - Every object and subject has a label
  - Some form of reference monitor
  - Use Bell-LaPadula model and some form of DAC
- **B2 - Structured Protection**
  - More testing, review, and validation
  - OS not just one big program (least priv. within OS)
  - Requires covert channel analysis
- **B3 - Security Domains**
  - More stringent design, w. small ref monitor
  - Audit required to detect imminent violations
  - requires security kernel + 1 or more levels *within* the OS

Division A

- **A1 – Verified Design**
  - Design must be formally verified
  - Formal model of protection system
  - Proof of its consistency
  - Formal top-level specification
  - Demonstration that the specification matches the model
  - Implementation shown informally to match specification

Limitations of Orange book

- **How to deal with floppy disks?**
- **How to deal with networking?**
- **Takes too long to certify a system**
  - People don't want to run n-year-old software
- **Doesn't fit non-military models very well**
- **What if you want high assurance & DAC?**