Repetition: Lect 3_A
Ch 4. Identification and Authentication

Objectives

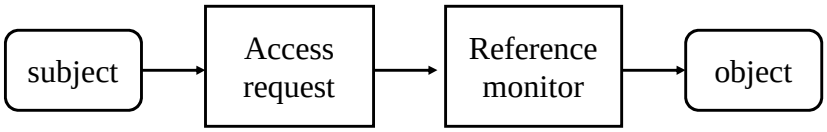
- Revisit familiar (login) mechanisms
- Intro to password protection
- Password bootstrapping
- Biometrics
- Password hacking – rainbow tables, GPU
- Authentication Tokens

Ch. 5 ACCESS CONTROL

- Subject and objects
- Access operations
- Access control structures
- Intermediate controls
- Lattice of security levels (multi-level security)

Fundamental model of access control

- subject (active) users, processes, …
- object (passive) files, resources, …
- access operation read, write, …
- reference monitor grants/denies access



Cont'd

Subjects and objects allow two perspectives for specifying access control:

- what a subject is allowed to do
- what may be done with an object

Access operations

Elementary level: observe (look at)
alter (change)

(Bell-LaPadula security model)

access rights: e, a, r, w

	execute	append	read	write
observe			x	x
alter		x		x

Ex:Object execution without read access

Example: Cryptographic engine whose key is used but is not readable

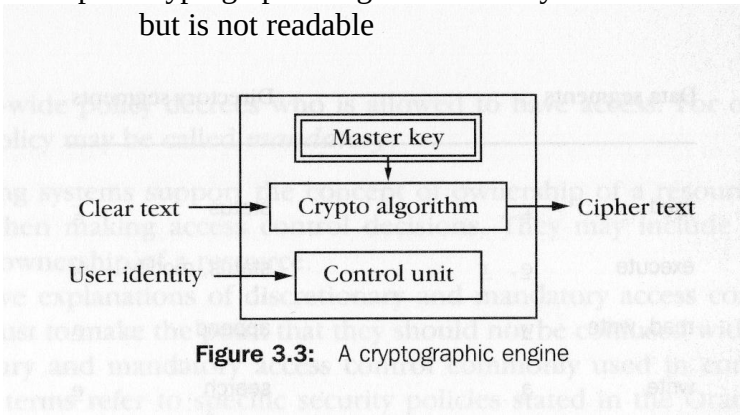


Figure 3.3: A cryptographic engine

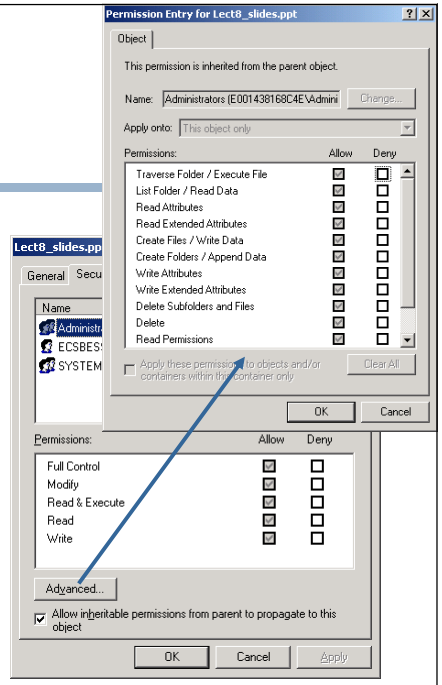
Unix/Linux interpretation of access rights

unix	**object**	
Access rights	**file**	**directory**
Read (r)	Read file	list directory contents
Write, (w)	Write file	create/rename a file
Execute, (x)	Execute program	search directory

Access rights can be set for three categories: Owner, Group, Others
e.g. rwxr-xr-- means: owner can read,write,execute, group members can Read and execute, and others can only read.

Windows 7/8/10 rights

- read
- write
- execute
- delete
- change permission
- change ownership



Windows XP/Win 7/8 rights - Cont'd

- Manipulating access rights:
  - grant
  - revoke
- Delegation policies

  one subject invokes another subject

Win 7/8/10/Windows 2000

- Very powerfull and flexible access control
- Many features:
  - **Positive:** Likely that you can configure the access control you want
  - **Negative:** It is easy to get it wrong because of the many possible choices

Ownership of access rights

- **Owner** decrees who is allowed access.
  ⇒ **Discretionary (DAC)**
- A **system-wide police** decrees what access is allowed

  ⇒ **Mandatory (MAC)**

Example of a trusted OS: SELinux

- Discretionary Access Control (DAC) in Linux provides not enough choices for controlling objects.
- Mandatory Access Control (MAC) allows you to define permissions for how all processes (called *subjects*) interact with other parts of the system such as files, devices, sockets, ports, and other processes (called *objects* in SELinux).

SELinux Overview

- Implement Flask architecture.
- SELinux is implemented in the Linux kernel using the Linux Security Module (LSM) framework.
- To support fine-grained access control, SELinux implements two technologies: *Type Enforcement* (TE) and *Role-based Access Control* (RBAC).

http://www.nsa.gov/selinux/info/docs.cfm
http://fedoraproject.org/wiki/SELinux
SE Linux is partly used in Android

Access Control Structures

- set S subjects
- „ O objects
- „ A access operations

The access rights are uniquely defined by the access control matrix, $M=(M_{SO})$ with $M_{SO} \subset A$, $s \in S$, $o \in O$

Hard to implement (in this way) !

Example: Access Matrix

	RAM	port	File 1	File 2	File 3	…
kernel	r,w,e	r,w	r,w,e	r,w,e	r,w,e	
service		r,w	e			
Kalle				r,w,e	e	
Mia				r,e	r,w,e	

Access control vs capability

- Access control list: focus on object
- Capability: focus on subject

Access Contol Lists (ACLs)

- Stores access rights to an object with the object itself
- Common to place users in groups

unix: user, group, others

ACLs are very powerful for access control but in a system with many users they are hard to manage (manually)

Capabilities

Give each subject a capability

Example:

Alice: edit.exe:execute; fun.com:read

Bill: fun.com:read, write; …

- difficult to overview
- difficult to revoke a capability

Intermediate Controls

- Groups and negative permissions (Fig. 5.6, 5.7)
- Protection rings: an "importance number"
  access if subject number ≤ object number

Example:

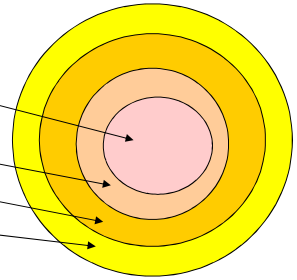
number: subject

0: OS Kernel

1: OS

2: services

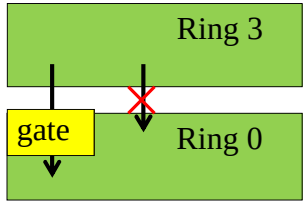
3: user process



Rings and gates

Idea: Program cannot call code of *higher privilege* directly but it can go via a gate

- A Gate is a special memory address where lower-privilege code can call higher
  - E.g. Enables OS to control where applications call it (system calls)



ARM & Intel

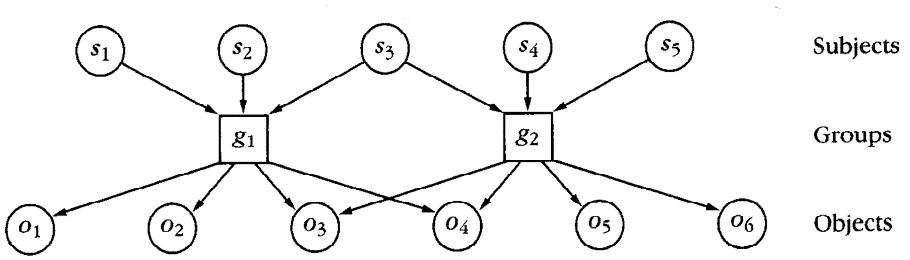
ARM

- 2 (3) rings
  - User space
  - Supervisor space
  - (TrustZone)

Intel/AMD (x86 compatible)

- 4 (5) rings
  - 0-3: see previous slides
  - ring "-1": for virtualization

Fig 5.6



Groups serve as an intermediate access control layer

Role-Based Access Control (RBAC)

- Subjects derive their rights from their current role.
- Ex: User+the current job $\Rightarrow$ role

LAYERS:

- Roles: collection of procedures
- Procedures: high-level access control methods.
- Data types: each object is of a certain data type; access only through procedures for this data type.

RBAC common in database management systems.

Example RBAC

Consider a hospital

subjects: persons

roles:

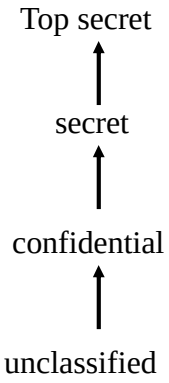
- **employee**: person employed by the hospital
- **nurse**: person that is allowed to act as nurse and is on duty (according to a stored schedule in the system)
- **surgeon**: person that is allowed to act as surgeon and is on duty

Role 1: access to patient journal: nurse or surgeon

Rele 2: …

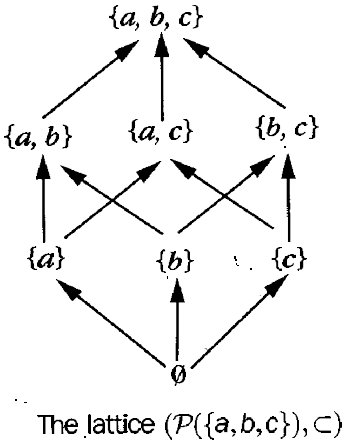
The Lattice of Security Levels

Example:
Linear lattice:



Note:
This is an example
of a partial ordering
(no arrows between each)

Lattice on P({a,b,c}) with ⊂



The lattice $(\mathcal{P}(\{a,b,c\}), \subset)$

P(set S)= set of all subsets of S

Ch 12. Security Models

- Implement your security policy
- Entities/ rules that constitute your policy

Models:

- Formal (Bell-LaPadula (BLP), Bida,...)
- Informal (Clark-Wilson)

Objectives of this chapter

- Demonstrate how security policies can be expressed in a *formal* way.
- History of computer security
- Scope and limitation of various models

State machine models

- State = representation of the system at a certain time
- State transitions = defines next state depending on present state and input (an output can also be generated)

Idea: Check if all state transitions **preserve** the security property+secure initial state.

If yes, then the system is secure !

The Bell-LaPadula model (Chap 11)

- A formal description of the allowable paths of information flow in a secure system.
- Captures the **confidentiality** aspects of access control.
- Access through access control matrix + security levels.
- Prevent information flowing downwards (military strategy)
- *multi-level security*

BLP

- a set of subjects S
- a set of objects O
- a set of access operations A={e,r,a,w}
- a set L of security levels with partial ordering ≤

BLP - states

Set of states B x M x F, where B=P(S x O x A) is the set of current accesses.

$b \in$ B is a set of tuples $b=(s,o,a)$, indicating that (subject) s performs (action) a on (object) o.

M set of access permission matrices M=(M_{so})

F set of security level assignments

BLP - set of security level assignments

$f \in F$ is a triple of mappings (f_S, f_C, f_O), where

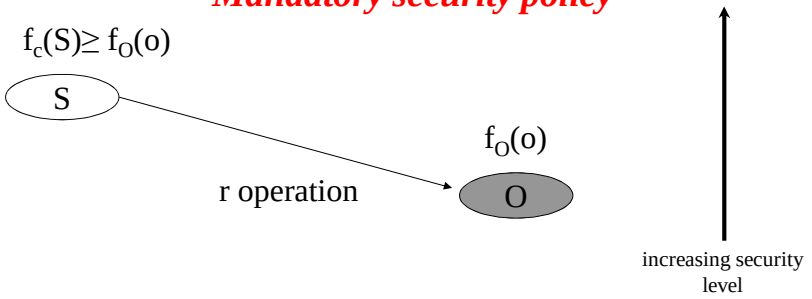
f_S: S $\rightarrow$ L maximal security level for each s(ubject)
f_C: S $\rightarrow$ L current security level for each s(ubject)
f_O: O $\rightarrow$ L classification for each o(bject)

Observe: $f_C \leq f_S$ (per definition)

Mandatory security policy

- A subject may read an object only if the subject's security level dominates the object classification.

Mandatory security policy



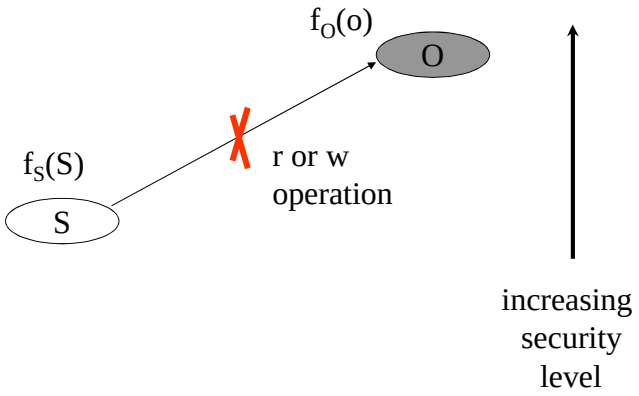
No read up security policy

Simple **s**ecurity property, **ss-property**
A state (b,M,f) satisfies the ss-property if each

(s,o,a) $\in$ b where a $\in$ {r,w} we have $f_O(o) \leq f_S(s)$.

No read-up security policy

ss-property - what does it mean?



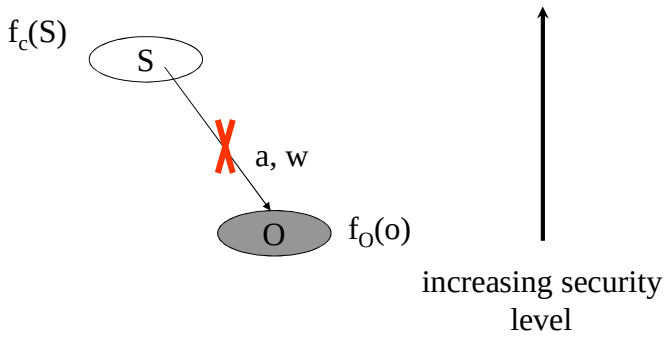
No write-down security policy

A state (b,M,f) satisfies **the *-property** if each (s,o,a) ∈ b where a ∈ {a,w} we have $f_C(s) \leq f_O(o)$.

No write-down security policy.

Furthermore, if (s,o,a) ∈ b where a ∈{a,w} then $f_O(o') \leq f_O(o)$ for all (s,o',a') ∈ b where a' ∈ {r,w}.

*-property - what does it mean?



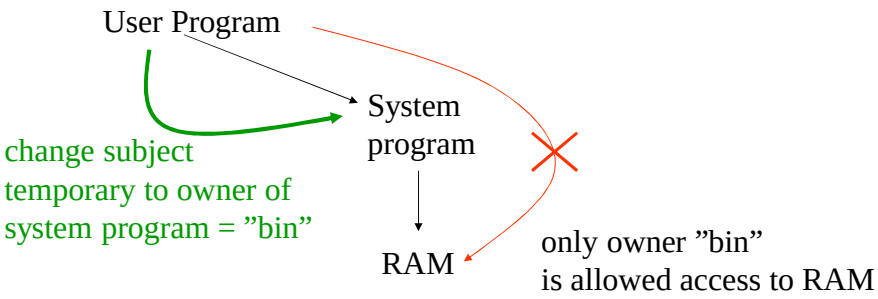
Problem: High-level subjects cannot send messages to low-level objects

Solution:

- Temporarily downgrading a subject (change of f_C)
  (Processes "forget" their contents)
- Trusted subjects
  (may violate the *-property)

Unix/Linux

Unix/Linux has the set UID mechanism



Discretionary security property

- Subject may pass an access permission on to other users.

- **ds-property**:

  A state (b,M,f) satisfies the ds-property if for each (s,o,a) ∈ b we have a ∈ M_{so}.

A state is secure if all three properties are satisfied.

Basic security theorem

If all state transitions are secure and if the initial state is secure, then every reachable state will be secure.

BLP - criticism

- Focus on confidentiality, not integrity
- Not addressing management of access control
- containing a covert channel.

Covert channel = information flow not controlled by the security mechanisms.

The Biba model

- addresses **integrity** similar to BLP
- integrity levels (Lattice (L,≤))
- “corruption of clean high level entities by dirty low-level entities”.
- Information may only flow downwards.

Not included

- Chinese Wall model
- HRU model

The HRU model

- include deletion/creation of subject/object
- include change of access rights

- subject S, object O, access rights R, access matrix $M=(M_{so})$, where $M_{so} \in R$.
- Primitive operations: create s, delete s, create o, delete o, enter r into M_{so}, delete r from M_{so}

- Verifying safety is an undecidable problem.
  (Mono-operational authorization system gives a decidable problem)
- The more complex model, the harder to verify!

Chinese Wall model

- Goal: No information flow that causes a conflict of interest.

- Similar framework as BLP.

The Clark-Wilson model

- *Focus: data integrity*

Integrity requirements:

- Internal consistency
- External consistency (relation to the real world, auditing)

Clark-Wilson model (cont'd)

- Programs = intermediate control layer
- abstract data types/object-oriented prog.
- "label subject and objects with programs instead of security levels" (BLP)
- integrity = being authorized to apply a program to a data item

Mechanisms for enforcing security

- **Well-formed transactions:**

  data can be manipulated only through a specific set of programs.

  Ex1: private key use on smartcards

- **Separation of duties:** users have to collaborate/collude to manipulate data

General points C-W model

- Subject has to be identified/authenticated
- Objects can be manipulated only by a restricted set of programs
- Subjects can execute only a restricted set of programs
- Proper audit is maintained
- The system has to be certified to work properly

Java, .NET, object oriented languages

Ch 6. Reference Monitor

How to make access control work in practice.

Objectives:

- Reasons for enforcing security at a low system layer.
- Overview of available security mechanisms at a low system layer.

Two good reasons for enforcing security at a low system level

- We can evaluate security at a higher level of assurance
- Reduce overhead caused by security enforcement

Operating System Integrity

- **Reference monitor**: Abstract machinery that controls all access to objects.
- **Security kernel**: the hardware, software etc. that implements the reference monitor concept.
- **Trusting computer base (TCB):** The totality of protection mechanisms

Security kernel

- Users must not be able to modify the operating system.
- Users must be able to use the operating system
- Users must not be able to misuse the operating system

  status information, controlled invocation

Modes of operation

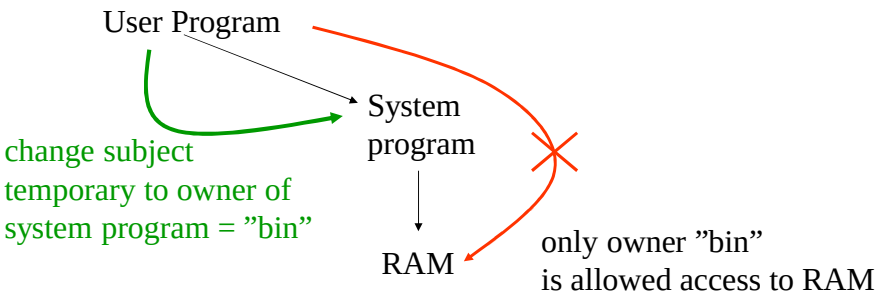
- Distinguish between computations from operating system - users
- A *status flag* allows the system to work in different *modes*.
- Unix: supervisor mode, user mode.
- Why? E.g., to stop users from writing directly to memory

Controlled invocation

- A user want to execute an operation requiring supervisor mode.
- Switch modes
- the system only performs a predefined set of operations in supervisor mode, then returns to user mode before giving control back to the user. *Controlled invocation*

Unix/Linux – controlled invocation

Set UID mechanism



Hardware Security

Computer architecture

- CPU: ALU, registers, program counter, stack pointer, status register
- Memory: RAM, ROM, ...

Security and Memory

- Volatile and non-volatile memory
- Memory contents may have been copied: caches, etc.
- Tamper-resistant memory

Threads and processes

- A process is a program in execution:
  executable code, data, execution context
- Processes communicate through the OS
- Threads are strands of execution within a process.
  - Share address space,
  - No control from OS.

Interrupts

- Trap
- trap - system saves and executes the interrupt handler.
- The interrupt handler must make sure the system is restored to a proper state!
- Interrupts of interrupts

(user interrupts an OS call by CTRL-C)

Example: Motorola 68000

- Status register - system byte
  S supervisor bit
  $I_2I_1I_0$ interrupt level bits

- OS calls TRAP#n
- Interrupts are masked
- memory mapped I/O
- 3 pins for function code indicating processor status to address decoder

Reference monitor – memory protection

Make sure users(its processes) are not accessing unauthorized memory.

- Fence registers
- base and bound registers
- tagged architecture (ARM TrustZone, ARM11)

Reference monitor – Multics OS

- Multics as an interpretation of BLP
- System items define the state
- Kernel primitives define the transitions

Note: historical value