Mandatory Access Control
Lesson Introduction

- Understand need for **mandatory access control (MAC) and multi-level security**
- Explore several **MAC models**
- Understand **assurance techniques for a trusted computing base (TCB)**

Trusted Computing Bases (TCB)

Revisiting Trusted Computing Base (TCB)

- How do we know TCB can be trusted?
- **Secure vs. trusted vs. high assurance**
  - Set of all hardware and software trusted to operate securely
  - Required for all other trust in the system security policy

Trusted Computing Bases (TCB)

Trusting Software:

- Functional correctness
  - Does what it was designed to do
- Maintains data integrity
  - Even for bad input
- Protects disclosure of sensitive data
  - Does not pass to untrusted software
- Confidence
  - Experts analyze program & assure trust
- Statement giving security we expect system to enforce
  - Do this formally when and where possible

Do what it is designed to do and nothing else! No unintended functionality.

the software does not change without authorization.

Not all users can access the same content of data. Users are only able to access the data they need to know.

TCB Design Principles

- **Least privilege for users & programs**
- **Economy**
  - Keep trusted code small as possible, easier to analyze & test
- **Open design**
  - Security by obscurity does not work

- **Complete mediation**
  - Every access checked, attempts to bypass must be prevented
- **Fail-safe defaults**
  - Default deny
- **Ease of use**
  - Users avoid security that gets in their way

If something goes wrong: close the connection.

How Do We Build a TCB:
Support Key Security Features

- **Must implement certain security relevant functions**
    - Authentication
    - Access control to files & general objects
    - Mandatory access control
    - Discretionary access control (standard file permissions)

How Do We Build a TCB:
Support Key Security Features

- **Protection of data used by OS** (OS must protect itself)
  - Security features of trusted OSes
    - Object reuse protection
    - Disk blocks, memory frames reused
    - Process can allocate disk or memory, then look to see what's left behind
    - Trusted OS should zero out objects before reuse
    - **Secure file deletion**: overwrite with varying patterns of zeros & ones
    - **Secure disk destruction**: degaussing, physical destruction

How Do We Build a TCB:
Support Key Security Features

- **Complete mediation of accesses**
- Trusted path from user to secure system
    - Prevents programs from spoofing interface of secure components
    - Prevents programs from tapping path (e.g. keyloggers)
- **Audit log showing object accesses** – only useful if you /look/ at the log
    - Detect unusual use of the system

Kernel Design

- Security kernel **enforces all security mechanisms**
- **Good isolation, small size for verifiability, keeps security code together**
- Reference monitor controls access to objects (monitors all references to objects)
- **Tamperproof** [impossible to break or disable]
- **Un-Bypassable** [always invoked, complete mediation]
- **Analyzable** [small enough to analyze & understand]

Kernel Design

What is included in the trusted computing base (TCB)?

- **All parts of OS needed** for correct enforcement of security policy
  - Handles primitive I/O, clocks, interrupt handling, hardware capabilities, label checking
- **Virtualization**
  - Virtual machine provides hardware isolation, logical OS separation

自由决定的

Discretionary Access Control

Two problems with DAC:

1. You cannot control if someone you share a file with will not further share the data contained in it
   - **Cannot control "information flow"**

2. In many organizations, **a user does not get to decide how certain type of data can be shared**
   - Typically the employer may mandate how to share various types of sensitive data

- Mandatory Access Control (MAC) helps address these problems

Sometimes you create the file but you do not have the full power to control who can access or modify the file. Some other regulations limit the freedom you have according to the sensitivity of the file content.

Mandatory Access Control (MAC) Models

User works in a company and the **company decides how data should be shared**

- Hospital owns patient records and limits their sharing
  - **Regulatory requirements may limit sharing**
  - HIPAA for health information

Users can create and hold the files, but the company decide how you should share the information.

Mandatory Access Control (MAC) Models

Military and intelligence agencies:

Data has **associated classification level** and users are cleared at various levels

- Top secret, secret, confidential etc.
- Limits on **who can access data at a certain level**
  - User cleared only at secret level should not be able to access top secret data
- Also called **multilevel security (MLS)**

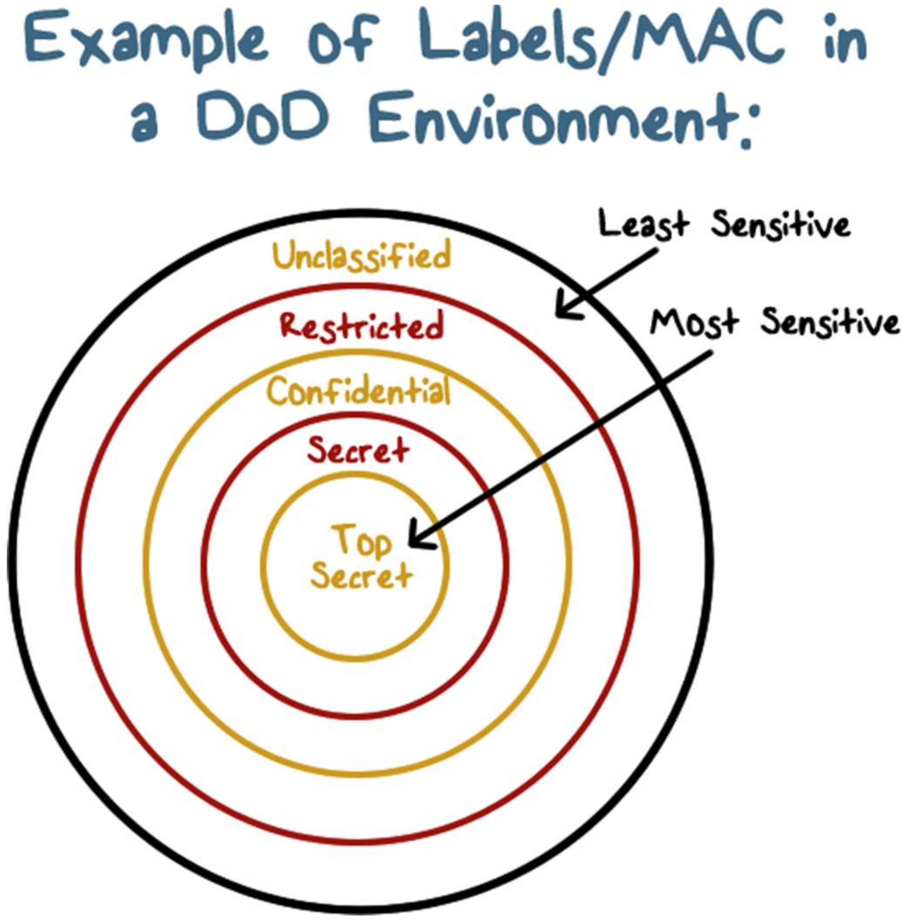
Most top secret information is time sensitive.

Implementing MAC

Labels: A Key Requirement for Implementing MAC

- indicate sensitivity/category of data or clearance/need-to-know requirements of users
- TCB associates **labels with each user and object and checks them when access requests are made**
  - Need to relate labels to be able to compare them
- Exact nature of labels **depends on what kind of model/policy is implemented**
  - DoD models include classification/clearance level and a compartment in the label
  - Commercial policies are different but use labels to deal with conflict-of-interest, separation-of-duty etc.

Implementing MAC



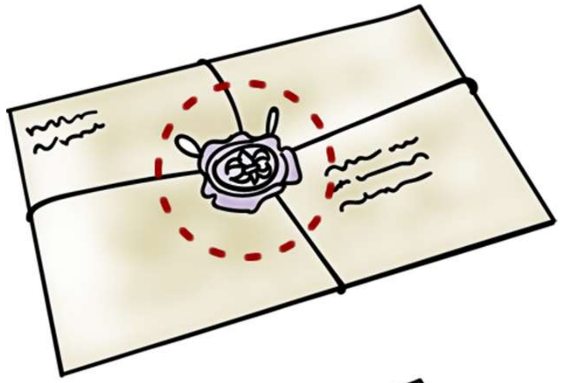
1. **Label** = (sensitivity level, compartment)

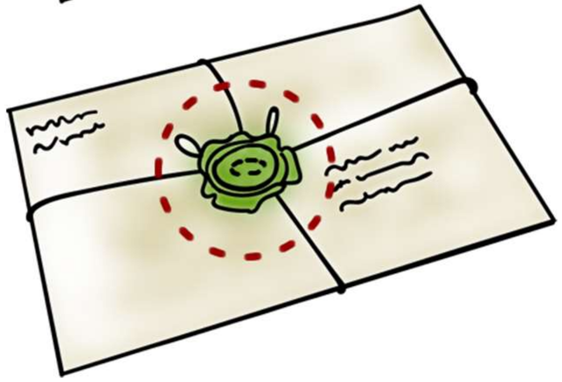
- Let us consider highly sensitive documents that have information about various arms stockpiles.

  L1 = (TS, {nuclear, chemical})
  L2 = (S, {nuclear, conventional})

- Providing confidential access to documents (Bell and La Padula or BLP Model)

Comparing Labels





- Assume **sensitivity levels are totally ordered**
  (TS > S > C > U)

- Compartments are sets which can only be partially ordered

- **How do we order labels?**

Comparing Labels

$L_1 = (X_1, \text{Comp}_1),\ L_2 = (X_2, \text{Comp}_2)$

L_1 dominates L_2 :	$l_1 > l_2$ and $\text{Comp}_1 \geq \text{Comp}_2$
or L_1 is dominated by L_2 :	$l_1 < l_2$ and $\text{Comp}_1 \leq \text{Comp}_2$
or $L_1 = L_2$:	$l_1 = l_2$ and $\text{Comp}_1 = \text{Comp}_2$
or L_1 and L_2 are not comparable :	$L_1 \not> L_2$ and $L_1 \not< L_2$ and $L_1 \neq L_2$

Ordering Among Labels

Ordering among labels defines a structure called a lattice:

Example:

Partial Order

$L_1 = (TS, \{A,B,C\})$ $L_1 > L_2$? Yes

$L_2 = (S, \{A,B\})$ $L_2 < L_1$? Yes

$L_3 = (S, \{B,C,D\})$ L_1 and L_3 are not compared

Using Labels for MAC: Confidentiality

- **Bell and La Padua or BLP Model** (Developed by DoD)
  - Assumes classification of data (TS, S, C, U) and clearances for subjects
  - Access modes can be one of two types

1. Simple security "no read-up"
   - Prohibits a subject of lower clearance from reading an object of higher classification, but allows a subject with a higher clearance level to read an object at a lower level (read down)
2. The * (star) property "no write down"
   - The * property (the write property) prohibits a high-level subject from sending messages to a lower-level object

   You can't take top secret information and write it to a disk which is protected at only secret level.

Biba Integrity Model

- Assigns integrity levels to subjects and objects using two properties
  - –The simple integrity (read) property
    - ●Permits a subject to have read access to an object only if the security level of the subject is equal to or lower than the level of the object
  - –The integrity * (write) property
    - ●Permits a subject to have write access to an object only if the security level of the subject is equal to or higher than that of the object
- Ensures no information from a subject can be passed on to an object in a higher security level
  - –Prevents contaminating data of higher integrity with data of lower integrity

Policies for Commercial Environments

- **User clearance is not common**
- Other requirements exist
    - **Data only be accessed by certain application** (e.g., payroll)
    - **Separation-of-duty and conflict-of-interest requirements**

eg: you don't want someone who is trading with stocks to has access to internal information of the stock market

Clark-Wilson Model

The three main rules of integrity models:

1. Prevent unauthorized users from making modifications
2. Prevent authorized users from making improper modifications (separation of duties)
   information access to be accomplished through a policy controlled process
3. Maintain internal/external consistency (well-formed transaction)

Clark-Wilson model addresses each of these goals. Biba model only addresses the first goal.

Clark-Wilson Integrity Model

The model uses the following elements:

- Users: Active agents.
- Transformation Procedures (TPs): Programed abstract operations, such as read, write and modify.
- Constrained Data Item (CDI): A data item whose integrity is to be preserved. Can only be manipulated by TPs.

Clark-Wilson Integrity Model

The model uses the following elements:

- Unconstrained Data Item (UDI): Data items outside of the control area of the modeled environment such as input information. Can be manipulated by users via primitive read and write operations.
- Integrity Verification Procedure (IVP): Check the consistency of CDIs with external reality.

Confirm the transformation procedures are doing what they are supposed to do. A third party check to ensure that the TP is not changing itself.

Clark-Wilson Model

- Developed by Clark-Wilson in 1987, the model addresses the integrity requirements of applications.
- Clark-Wilson model enforces the three goals of integrity by using access triple (subject, software TP, and object), separation of duties, and auditing. It enforces integrity by using well-formed transactions (through access triple) and separation of user duties.

Clark-Wilson Integrity Model

- Built upon principles of change control rather than integrity levels
- Designed for the commercial environment
- Its change control principles
  - No changes by unauthorized subjects
  - No unauthorized changes by authorized subjects
  - The maintenance of internal and external consistency

Clark-Wilson Integrity Model (cont'd.)

- Establishes a system of subject-program-object relationships
  - Such that the subject has no direct access to the object
  - The subject is required to access the object using a well-formed transaction using a validated program
- Provides an environment where security can be proven through separated activities, each of which is provably secure
- Same user cannot execute two programs that require separation-of-duty

Revisiting Assurance

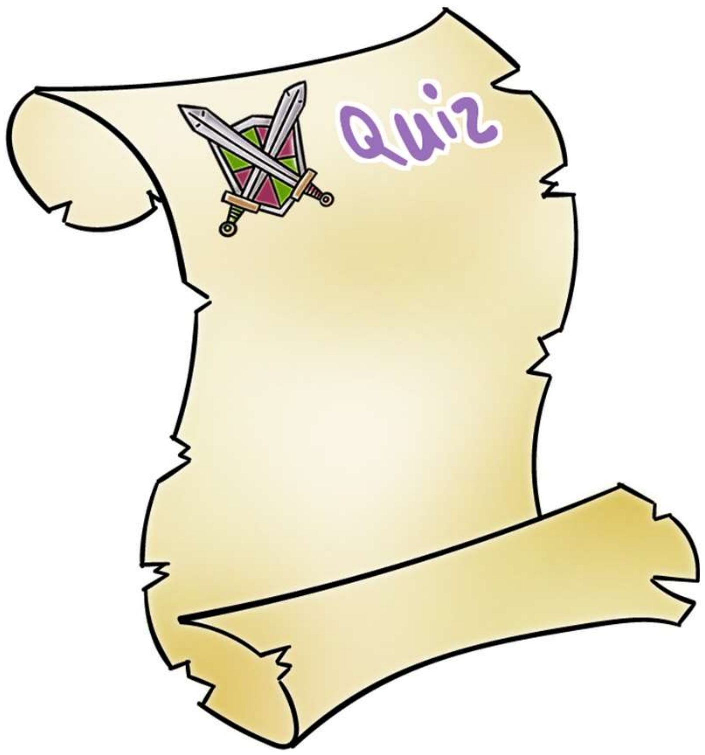
Assurance: Ways of convincing ourselves
that a model, design, & implementation are correct
ensure all the holes are closed

Methods of assurance validation:

- Testing /Penetration testing
- Formal verification Validation
- Checking that developers have implemented all requirements
- Requirements checking, design & code reviews, system testing

Very very lucrative field. Everybody needs this but few of us know how to do this.

Revisiting Assurance



Testing:

- Demonstrate existence of problem
- Cannot demonstrate absence of problem
- **Regression testing**: ensure that alterations do not break existing functionality / performance (regression: "going backwards")

make sure the new application does not mess up the existing ones.

Revisiting Assurance

Challenges:

- Test case generation
- Code coverage
- Exponential number of different executions
- Different execution environments

Penetration testing:

- Ethical hackers attempt to defeat security measures
- Cannot demonstrate absence of problem

Revisiting Assurance

Formal verification: Checking a mathematical specification of program to ensure that security assertions hold.

- **Model checking**, automated theorem proving
- State variables w/ initial assignment, program specification describing how state changes, boolean predicates over state variables
- **Difficulty:** exponential time & space worst case complexity
- Model checking pioneers won the 2007 Turing Award

Security Evaluations

Government Security Evaluations

- **U.S. Orange Book (late 1970's)**
- D < C1 < C2 < B1 < B2 < B3 < A1
  - **D:** no protection
  - **C:** discretionary protection
  - **B:** mandatory protection
  - **A:** Verified protection
- C1, C2, B1: security features common to commercial OSes
- B2: Proof of security of underlying model, narrative spec of TCB
- B3, A1: Formal design & proof of TCB

Government Security Evaluations

Common Criteria (2005) international standard replaced orange book

- Originated out of European, Canadian, and US standards
- **Idea:** users specify system needs, vendors implement solution and make claims about security properties, evaluators determine whether vendors actually met claims
- **Evaluation assurance level** (EAL) rates systems
  - EAL1 most basic, EAL7 most rigorous

Mandatory Access Control
Lesson Summary

- Provides enterprises **an ability to control how sharing** of sensitive information can be controlled
- Can address both **confidentiality and integrity** but require added functionality with labels
- **High level of assurance** for trusted systems is challenging