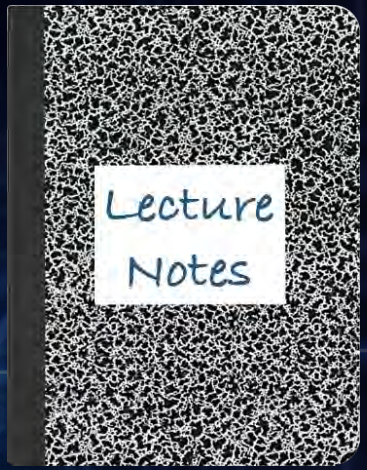
CS 419: Computer Security

Week 2: Part 3
Mandatory Access Control



Paul Krzyzanowski

What's wrong with ACLs?

- **Users are in control**

```
chmod o+rw secret.docx
```

  - Now everyone can read and modify `secret.docx`
- **Doesn't work well in environments where management needs to define access permissions**
- **No ability to give time-based or location-based permissions**
- **Access is associated with objects**
  - **Hard to turn off access for a subject** - except by locking the user
  - Otherwise have to go through each object and remove user from the ACL

    … but you're still stuck with default access permissions and wondering how other users will set access rights in the future

Access Control Models: MAC vs. DAC

DAC: Discretionary Access Control

- A subject (domain) can pass information onto *any* other subject
- In some cases, access rights may be transferred
  e.g., *chown*
- ***Users are in charge of access permissions***
- *Most systems use this*

MAC: Mandatory Access Control

- Policy is centrally controlled
- Users cannot override the policy
- ***Administrators are in charge of access permissions***

MLS: Multilevel Security Systems

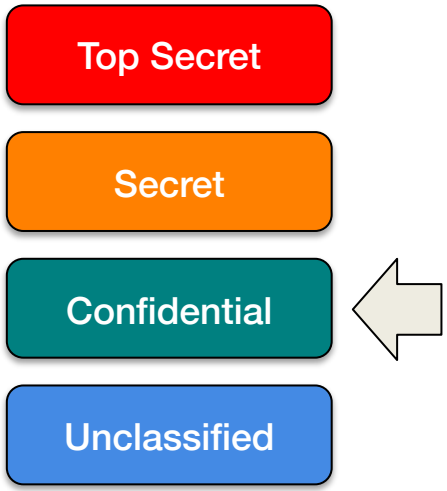
Designed to address security concerns in the Air Force

Handle multiple levels of classified data in one system

Bell-LaPadula Model

- Designed for the military
- Based on U.S. military classification levels

Motivation:
Preserve confidentiality. If one program gets hacked, it will not be able to access data at higher levels of classification

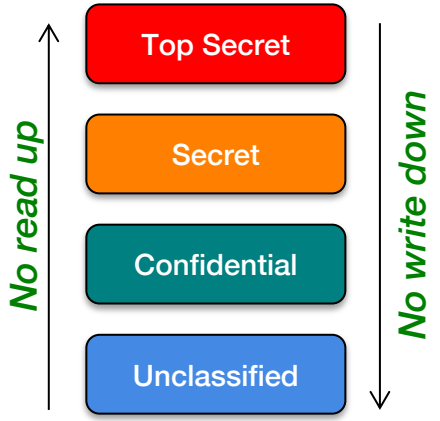


If you have confidential clearance:

- You can **access** confidential & unclassified data
- You can **create** confidential, secret, and top-secret data

Bell-LaPadula (BLP) Access Model

- **Objects are classified into a hierarchy of sensitivity levels**
  - Unclassified, Confidential, Secret, Top Secret
- **Each user is assigned a clearance**
- **"*No read up; no write down*"**
  - Cannot read from a higher clearance level
  - Cannot write to a lower clearance level
- **Assumes vulnerabilities exist and staff may be careless**
- **Need a "trusted subject" to declassify files**



Confidential cannot read Secret
Confidential cannot write Unclassified

Bell-LaPadula (BLP) Model Properties

Every subject & object gets a security label (e.g., confidential, secret)

1. **The Simple Security Property** – ***mandatory rules for reading***
   – No Read Up (NRU)
   A subject cannot read from a higher security level

2. ***-Property** (star-property) – ***mandatory rules for writing***
   – No Write Down (NWD)
   A subject cannot write to a lower security level

3. **The Discretionary Security Property**
   – Access control matrix can be used for **DAC** *after* MAC is enforced

Type Enforcement Model (TE)

Secondary Access Control Matrix that gives MAC priority over DAC

- **Domains and Types**
  - Assigns subjects to **domains**
  - Assigns objects to **types**
  - Matrix defines permitted **domain-domain** and **domain-type** interactions

Role-Based Access Control (RBAC)

- **More general than Bell-LaPadula**
- **Designed to allow enforcement of both MAC & DAC properties**
- **Access decisions do not depend on user IDs but on roles**
  - Administrators define roles for various job functions
  - Each role contains permissions to perform certain operations
  - Users are assigned one or more roles
- **Roles are *job functions*, not *permissions***
  - "update customer information" is a role
  - "write to the database" is not a role
- **Enables fine-grained access**
  - Roles may be defined in application specific ways (e.g., "move funds")

RBAC Rules

- **Role assignment**
  - A subject can execute an operation only if the subject has been assigned a role
- **Role authorization**
  - A subject's active role must be authorized for that subject
  - Ensures that users can only take on roles for which they have been authorized
- **Transaction authorization**
  - A subject can execute a transaction only if the transaction is authorized through the subject's role membership

RBAC is essential to database security

Aren’t roles == groups?

- **Group = collection of users**
  - Does not enable management of user-permission relationships
- **Role = collection of permissions**
  - Permissions can be associated with users and groups
- **Roles have a session**
  - Users can activate a role

RBAC Benefits

- **RBAC is hugely popular in large companies**
  - Driven by regulations such as HIPAA and Sarbanes-Oxley
- **Makes it easy to manage movement of employees**
- **Makes it easy to manage "separation of duty" requirements**
- **Can manage complex relationships**
  - Doctor X wants to view records of Patient Y
  - Doctor needs roles of "Doctor" and "attending doctor with respect to Y"
  - Roles allow specification of *only if*, *not if* or *if and only if* relations
- **RBAC can simulate MAC and DAC**

See http://csrc.nist.gov/groups/SNS/rbac/faq.html

SELinux (Security Enhanced Linux)

SELinux = Security-Enhanced Linux

Originally a kernel patch created by the NSA to add MAC to Linux

Supports three MAC models:

1. **Type Enforcement** (**TE**)
2. **Role-Based Access Controls** (**RBAC**)
3. **Multilevel Security** (**MLS**) – the Bell-LaPadula Model
   - **Multi-Category Security** (**MCS**)
     - Extension of MLS to define categories within a security level

There other security models and implementations available in other distributions

Type Enforcement (TE) on SELinux

- **Subjects are grouped into domains**
  - Processes are subjects – they run with the privileges of a user
  - Each subject is assigned a label identifies its domain
- **Objects are grouped into types**
  - A label assigned to an object (file) identifies its **type**
- **Domains & types are managed in the same way**
  - Each has a security context, represented by a **security ID** (**SID**)
- **An Access Control Matrix defines subject-object permissions**
- **Each process has a security ID (SID), user ID, and group ID**

Type Enforcement (TE) on SELinux

Access control rules

The security administrator defines
what access a **domain** (subject) can perform on a **type** (object)

```
allow userdomain bin_t:file: execute;
allow user2domain bin_t:file: read;
```

- **Allows users with the label "`userdomain`" execute rights for files with the label "`bin_t`"**
- **Allows users with the label "`user2domain`" read rights for those files**

RBAC in SELinux

- **RBAC is built on top of TE (type enforcement)**
  - Users mapped to roles at login time
  - Roles are authorized for domains
  - Domains are given permissions to access object types
- **Role-based access is specified in terms of TE**
  - Role = { groups, users, file operatons }
  - Goal is to simplify labeling

Note:
This does not allow fine-grained roles, such as "*access employee names*" or "*transfer funds*"

Biba Integrity Model

- **Bell-LaPadula was designed to address confidentiality**
- **Biba is designed to ensure data integrity**

Confidentiality = constraints on who can *read* data

Integrity = constraints on who can *write* data

Motivation:
Preserve data integrity.
If one program gets hacked, it will not be able to modify data at higher levels of integrity

Biba model properties

- **Simple Security Property** = A subject cannot read an object from a lower integrity level
  Subjects may not be corrupted by objects from a lower level
  No read down
- **Star property** = A subject cannot write to an object at a higher integrity level
  Subjects may not corrupt objects at a higher level than the subject
  No write up
- A process cannot request higher access

An example of where Biba is useful

The Biba model fits certain real-world applications

- **ECG device**
  - Runs a calibration process, which stores a calibration file = *high integrity*
  - Runs user processes, that run ECG tests = *lower integrity*
- **Normal users cannot write the calibration file but can read it**
  - Can read data at higher levels (calibration = higher data level)
    - User process can read calibration data – but cannot modify it
- **Calibration process can write data to lower levels**
    - Calibration process can write to the user process – but cannot read user data
- **Works well when you need to get data from a trusted device**

Biba Problems

Like Bell-LaPadula, it doesn't always fit the real world

- **Microsoft offers Mandatory Integrity Control (Biba model)**
  - User's access token gets assigned an integrity level
  - File objects have an Access Control Entry (ACE) to hold an integrity level:
    - ***System***: Critical files
    - ***Medium***: Regular users and objects
    - ***High***: Elevated users
    - ***Low***: Internet Explorer, Adobe Reader, etc.
    - New process gets the *minimum* of the user integrity level and the file integrity level
  - Default policy = *NoWriteUp*
    - Goal: Apps downloaded with IE can read files but cannot write them – limit damage done by malware
    - Trusted subjects would have to overwrite the security model
      - Users get used to the pop-up dialog boxes asking for permission!
  - Microsoft dropped the *NoReadDown r*estriction
    - Did not end up protecting the system from users

MAC vs DAC Summary

- **DAC = Discretionary Access Control**
  - The user is in charge of setting file permissions
  - If you own a file, you can set any access permissions you want on it … and even give it away
  - The root user (user ID 0) has the power to change any permissions
- **MAC = Mandatory Access Control**
  - System owner (administrator) defines security policies
  - Users cannot override them, regardless of their privilege level

MAC takes priority over DAC

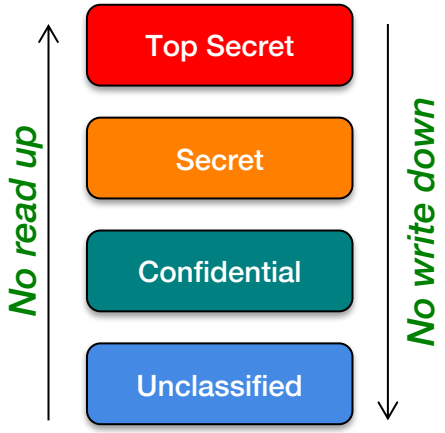
Access Models: Summary

- **Discretionary Access Control**
  - Works great when it's ok to put the user is in charge
- **Mandatory Access Control**
  - Needed when an organization needs to define policies
  - Bell-LaPadula (BLP)
    - Oldest & most widely studied model – synonymous with MLS
    - Designed to protect confidentiality
    - Doesn't work well outside of the DoD … and is clunky within the DoD
  - Type Enforcement (TE)
    - Simple MAC model to override DAC
  - Role-Based Access Control (RBAC)
    - Identifies roles and assigns users to roles
    - Made popular by business needs
    - Most actively used MAC model
  - Biba Model
    - Opposite of Bell-LaPadula: concerned with integrity, not confidentiality

Multilateral Security

Multilevel Security

- Subjects and objects have assigned classification labels
- Rules control what you can read or write

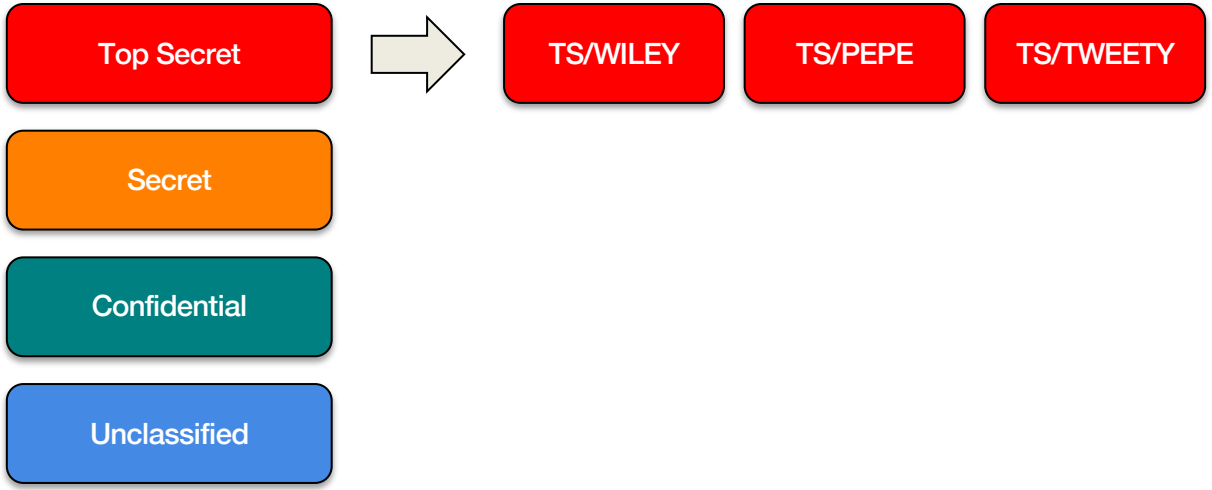


Bell-LaPadula

Multilateral Security

Each security level may be divided into compartments

- Usually applied to the top-secret level
- TS/SCI = Top-Secret / Special Compartmentalized Intelligence
- You will be granted access to specific compartments
  - Formalized description of “need to know”

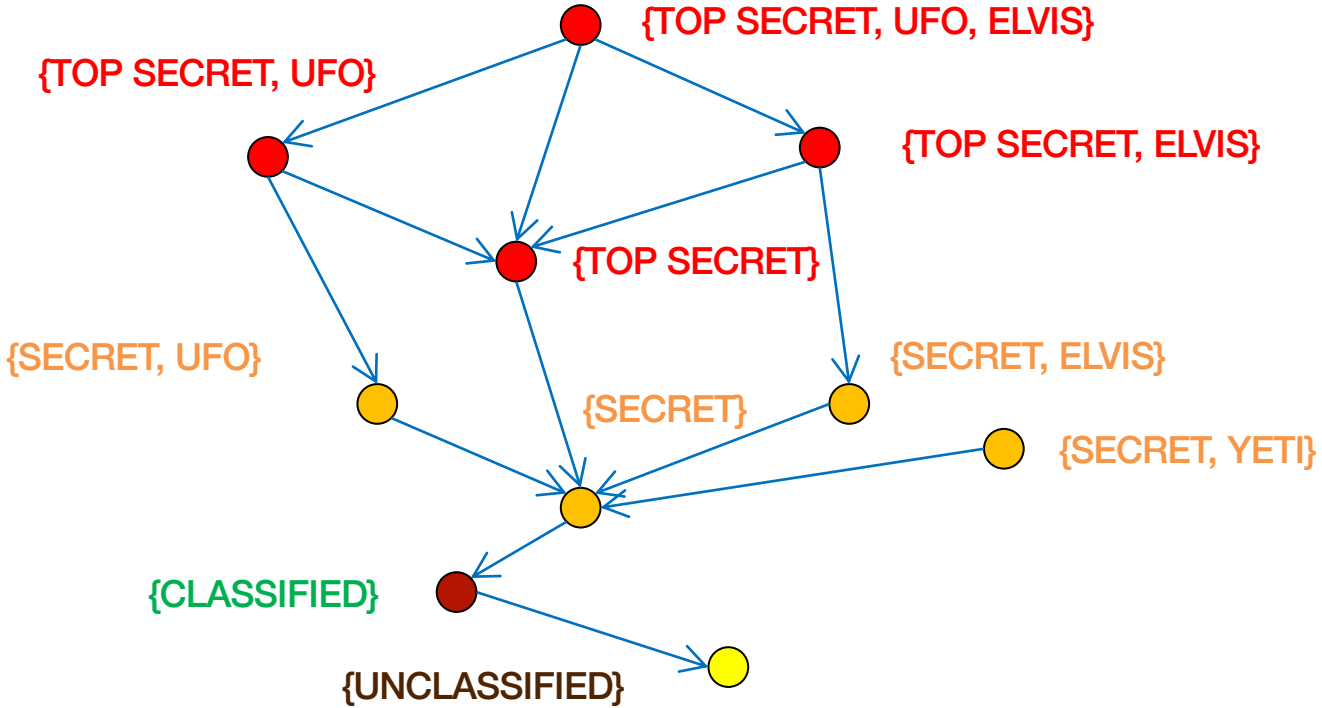


Compartmentalization

- **Subjects & objects get security labels (compartments) in addition to security classification labels**
- **If you do not have clearance for the label, you cannot access the data**
  - {TOP SECRET, UFO} cannot be read by someone with only {TOP SECRET} clearance
  - Neither can {SECRET, UFO}

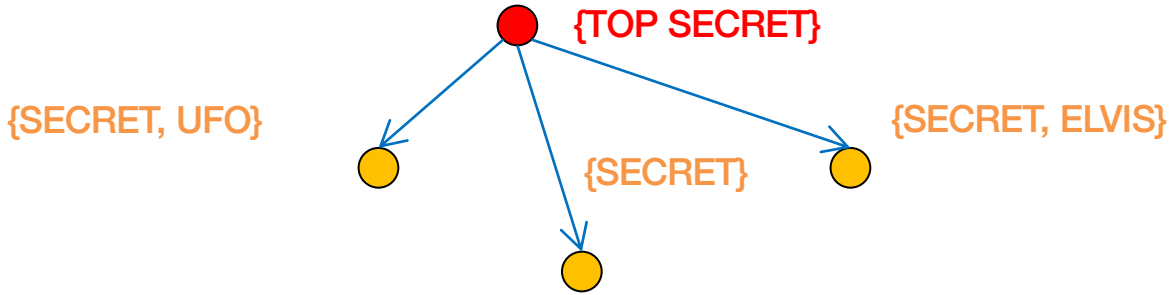
Lattice Model

Graph representing access rights of different labels & levels



Multilateral Security

- **Data from two compartments ⇒ third compartment**
  - Creates more isolation
  - Does not help with sharing
- **One option**
  - Allow multiple compartments at a lower level to be readable by a higher level



Multilevel & Multilateral Security Models

- **Do not help downgrading data**
  - Need special roles to re-label or declassify data
- **Handing searches across compartments is difficult**
  - No single entity will likely have rights to everything

Chinese Wall model

Chinese wall = rules designed to prevent conflicts of interest

- Common in financial industry
  - E.g., separate corporate advisory & brokerage groups
- Also in law firms and advertising agencies

- **Separation of duty**
  - A user can perform transaction *A* or *B* but not both

- **Three layers of abstraction**
  - **Objects**: files that contain resources about some company
  - **Company groups** = set of files relating to one company
  - **Conflict classes**: groups of competing company groups:

    Class 1 = `{Coca-Cola, PepsiCo, Keurig Dr. Pepper}`

    Class 2 = `{Alaska Airlines, American Airlines, United, Delta, JetBlue }`

Chinese Wall model

Basic rule

A subject can access objects from a company **only** if it never accessed objects from competing companies.

Simple Security property

- A subject ***s*** can be granted access to an object ***o*** only if the object
  - Is in the same company group as objects already accessed by ***s***

or

  - ***o*** belongs to a different conflict class

*-property

- Write access is allowed ***only*** if
  - Access is permitted by the simple security property

and

  - No object was read which is in a different company dataset than the one for which write access is requested *and* contains **unsanitized** information
    - ***Sanitization*** = disguising a company's identify
    - This means that you could read data across the wall ***only*** if it's anonymized

MAC can reduce the need for root

- **Traditionally the *root* user has supreme power**
  - You need supreme power to do *any* administrative task
  - Example: a network administrator can read – and modify – any files on the system
- **Models such as TE and RBAC allow you to define classes of users that can perform only certain operations and access certain files**
  - E.g., you can define a **network administrator** who can modify network configuration files and run network commands ... but not create user accounts or reboot the system

Security Risks

- **Even if the mechanisms work perfectly, policies may fail**
  - DAC: you're trusting the users or a sysadmin to set everything up correctly
  - MAC
    - User or role assignment may be incorrect
    - Collaboration needs to be considered
    - Models like Bell-LaPadula and Biba require overrides to function well
- **Corruption**
  - Attacks may change the definition of roles or the mapping of users to roles
  - This is an attack on the Trusted Computing Base
- **Users**
  - Most malware is installed willingly
  - Users thus give it privileges of – at least – normal applications
  - As far as the operating system is concerned, it is enforcing defined policy

Security Risks

- **Even administrators should not be able to read all files**
  - Many security systems enforce this
  - Edward Snowden should not have been able to copy sensitive documents onto a thumb drive … even if NSA policy banned thumb drives
- **General assumption has been that programs are trusted and run with the user's privileges**
- **Worked well for system programs**
- **Do you trust the game you installed on your phone?**
- **Need to consider better application isolation**
  - Android turned Linux into a single-user system
  - User IDs are used on a per-application bases

Program-Based Control

- **A lot of access decisions must be handled by programs, not the OS**
  - Database users and the access each user has within the database
  - Microsoft Exchange & Active Directory administrators
  - Mail readers
  - Web services: users are unlikely to have accounts on the system
  - Movement of data over a network
    - How do you send access permissions to another system?
    - Digital rights management = requires trusted players

- **Programs may implement RBAC (e.g., Exchange) or other mechanisms**
  - But the OS does not help

The End