Principles of Access Control

CYBR371: System and Network Security, (2024/T1)

Arman Khouzani (course coordinator), Mohammad Nekooei
Slides modified from "Masood Mansoori"

04 March, 2024

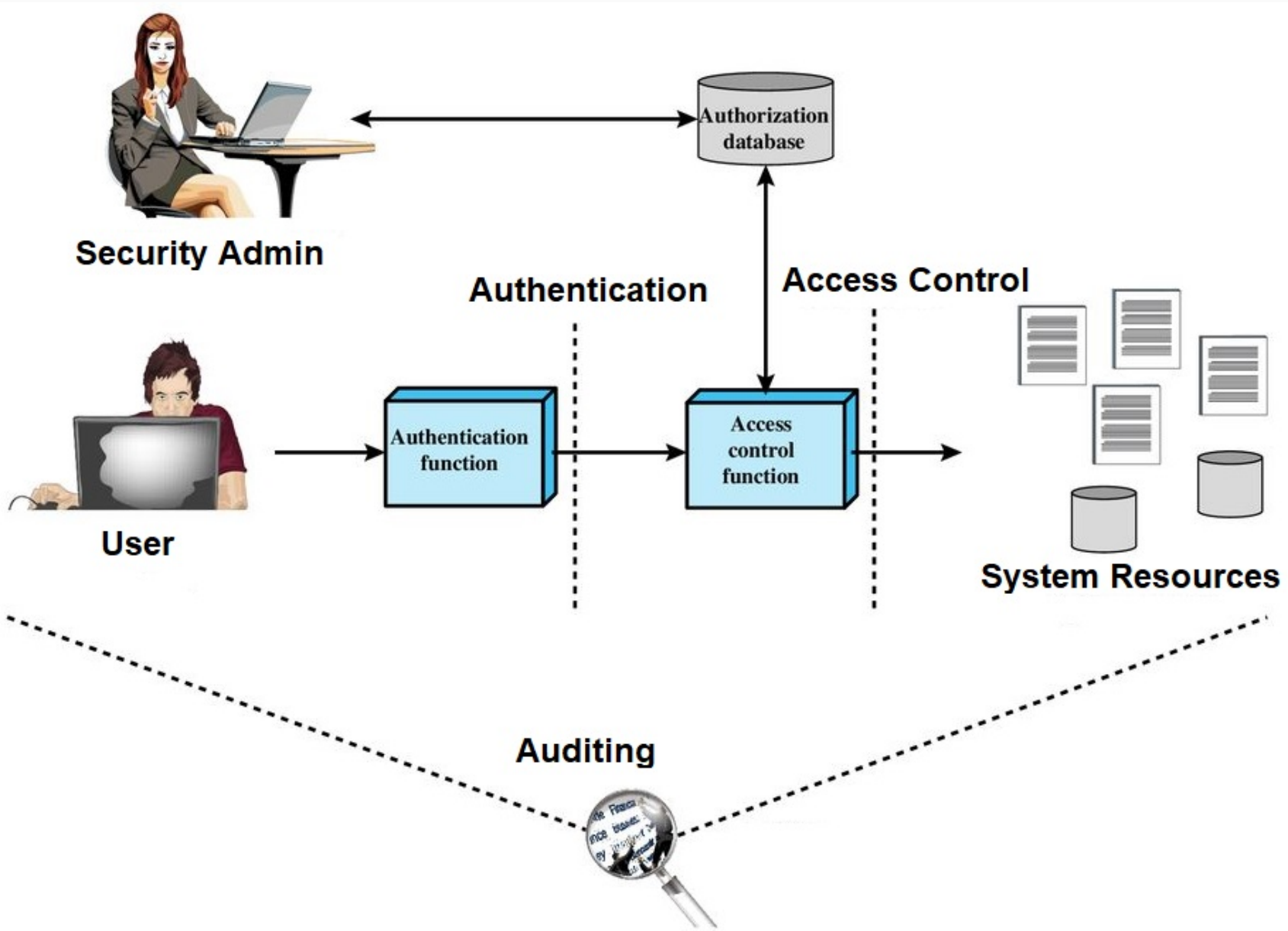
Victoria University of Wellington – School of Engineering and Computer Science

Access Control

The purpose is to limit the operations or actions that a legitimate user of a computer system can perform.

- Tries to prevent activities that could lead to a breach of security.
- Access control decision is actually an *authorisation* decision.
- if *o* is an object, authorisation answers the question “Who is trusted to access *o* and what can they do with it?”

Access Control



Access Control: Phases

Phases of Access Control Process

Phase	Purpose	Example
Identification	Who are you?	USER ID, IP Address
Authentication	Prove who you claim to be.	Password, badge, fingerprint
Authorisation	Which resources can you access? What are you allowed to do with those resources?	User A accesses Resource B in read and write mode
Accounting	What have you done?	User A has modified Resource B on `Date:Time`

Access Control: Identification & Authentication

- **Identification** should be *unique*, and *non-descriptive*, and have a *secure issuance*.

Authentication Methods

AuthN Method	Description	Examples
by Knowledge	Something only the user knows.	Password, PIN
by Ownership	Something only the user has.	Smart Card, badge, token
by Characteristic	Something only the user is/-does	Fingerprint, hand geometry, voice, keystroke dynamic, iris

Access Control: AuthN

Authentication Methods (usually Authentication by Characteristic, e.g. through biometrics) may not be completely accurate and may be susceptible to errors:

- Type-I error (false rejection): when a known legitimate authorised user is rejected as unknown/unauthorised user.
- Type-II error (false acceptance): when an unknown/unauthorised user is authenticated as a known/authorised user.

Access Control: AuthN

Multi-factor Authentication: requiring multiple *independent* evidences to establish identity. Examples:

- Bank Card+PIN
- use of “one time passwords” (OTP) generated by a security token or smartphone (either in combination with password, or requiring the user to authenticate to the device)
- sending a challenge through an independent channel (e.g. to a previously established email, or phone).

- but be careful not to mix identification with authentication!
- also don’t mix-up *2-factor authentication (2FA)* with *mutual authentication* (what are they?!)

Access Control: AuthN

Authentication methods in applications:

- Local: the application independently takes care of authentication (e.g. maintains a database of users and their salted password hashes)
- Claim based: where the user authenticates not directly with the app, but with a Trusted Third Party (authentication server, identity provider), and is issued a *claim* (or token/ticket). Ex.: `Kerberos`
- Federated: a.k.a. Single-Sing-On (SSO), where an app in a security realm can use resources in a different security realm on behalf of an authenticated entity: the authentication to one system is extended to others (if they trust the authN). Ex.: `SAML`, `CAS`

Subjects, Objects, and Access Rights

Subject

An entity capable of accessing objects.

- e.g., user, process.

Object

A resource to which access is controlled.

- e.g., file, hardware, process.

Access Rights

Describes the way in which a subject may access an object.

- e.g., read, write, execute.

▶ Subjects initiate actions or operations on objects.

Subjects and objects provide a different focus of control.

- What is the subject allowed to do?
- What may be done with an object?

Subjects, Objects, and Access Rights

Example of access rights/modes:

- For files, the typical access rights are:
  - read, write, execute

The OS implements them.

- For bank accounts, the typical access rights are:
  - inquiry, credit, debit

The application programs implement them.

Ownership

Ownership is an aspect often considered in access control rules.

When a new object is created, in many operating systems the subject creating the object becomes its owner.

Access Control Matrix

A conceptual model that specifies the rights that each subject possesses for each object.

- subjects in rows, objects in columns.

	File 1	File 2	File 3	File 4	Account 1	Account 2
John	Own R W		Own R W		Inquiry Credit	
Alice	R	Own R W	W	R	Inquiry Debit	Inquiry Credit
Bob	R W	R		Own R W		Inquiry Debit

Access Control Matrix: Implementation Approaches

Access matrix is usually sparse and hence not implemented as a matrix.

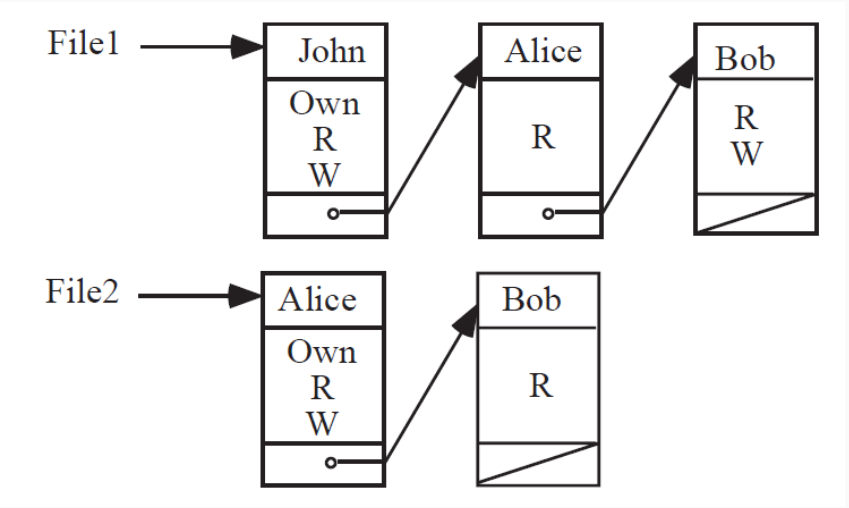
Some common approaches to implementing the access matrix practice:

- Access Control Lists (ACLs)
- Capabilities
- Authorisation Relations

Access Control Lists (ACL)

- Each object (e.g. File) is associated with an ACL.
- ACL has an entry of each subject if it has some kind of access to that object
- corresponds to storing the access matrix by columns.

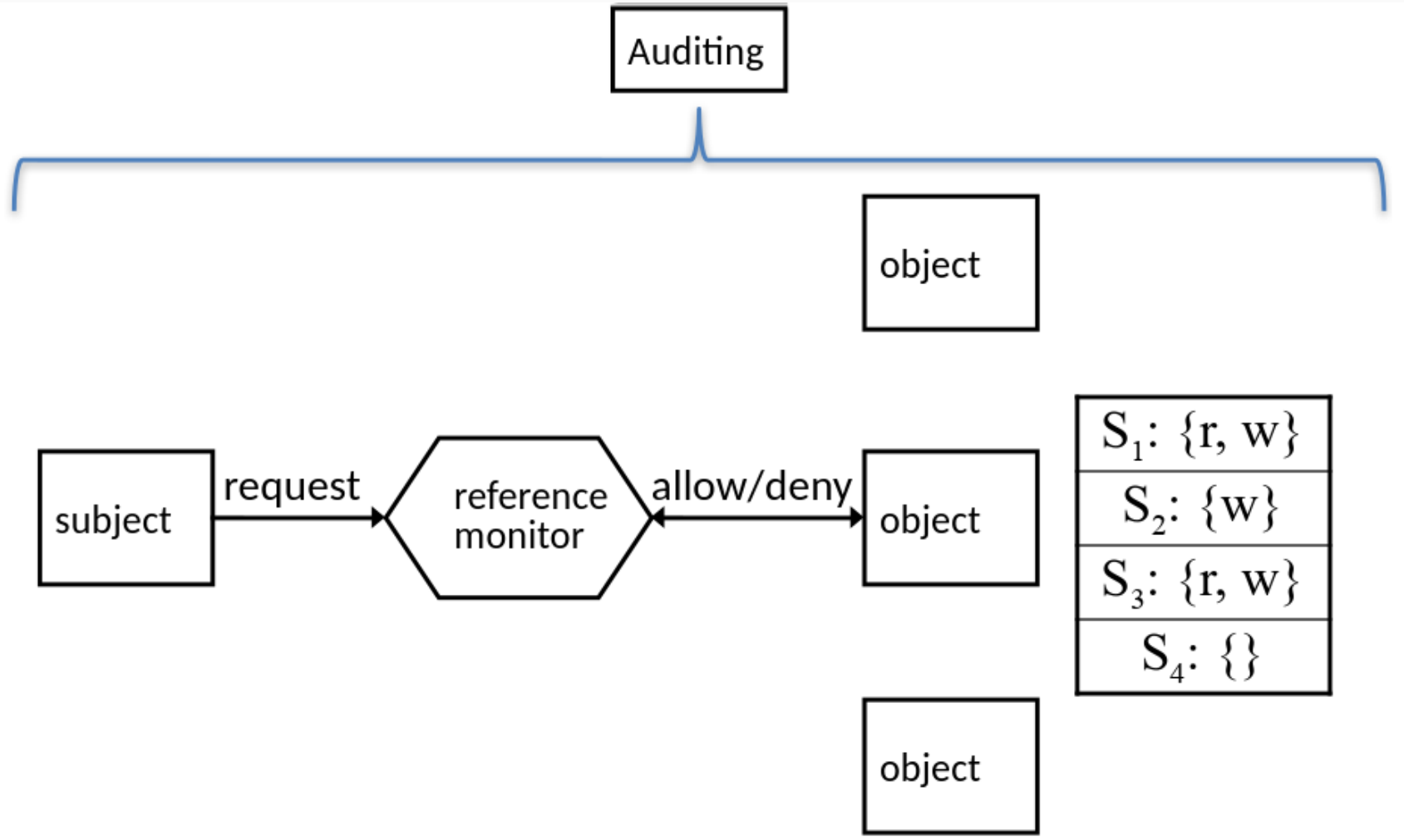
	File 1	File 2	File 3	File 4
John	Own R W		Own R W	
Alice	R	Own R W	W	R
Bob	R W	R		Own R W



To reduce the list length, the usual practice is to use groups instead of (or in addition to) individual subject identifiers.

Modern OS typically take the ACL-based approach.

Access Control Lists (ACL)



Access Control Lists (ACL)

Example: UNIX **`getfacl`** and **`setfacl`** allows to create ACLs on files and folders.

```
osboxes@osboxes:~$ ls -l
total 32
drwxrwxr-x 2 osboxes osboxes 4096 Mar  5 15:00 Desktop
drwxr-xr-x 2 osboxes osboxes 4096 Mar 26  2022 Documents
drwxr-xr-x 2 osboxes osboxes 4096 Mar 26  2022 Downloads
-rw-rw-r-- 1 osboxes osboxes    0 Mar  5 15:01 file1
-rw-rw-r-- 1 osboxes osboxes    0 Mar  5 15:01 file2
```

```
setfacl -m u:avahi:rwx file1
```

```
osboxes@osboxes:~$ ls -l
total 32
drwxrwxr-x  2 osboxes osboxes 4096 Mar  5 15:00 Desktop
drwxr-xr-x  2 osboxes osboxes 4096 Mar 26  2022 Documents
drwxr-xr-x  2 osboxes osboxes 4096 Mar 26  2022 Downloads
-rw-rwxr--+ 1 osboxes osboxes    0 Mar  5 15:01 file1
-rw-rw-r--  1 osboxes osboxes    0 Mar  5 15:01 file2
```

```
osboxes@osboxes:~$ getfacl file1
# file: file1
# owner: osboxes
# group: osboxes
user::rw-
user:avahi:rwx
group::rw-
mask::rwx
```

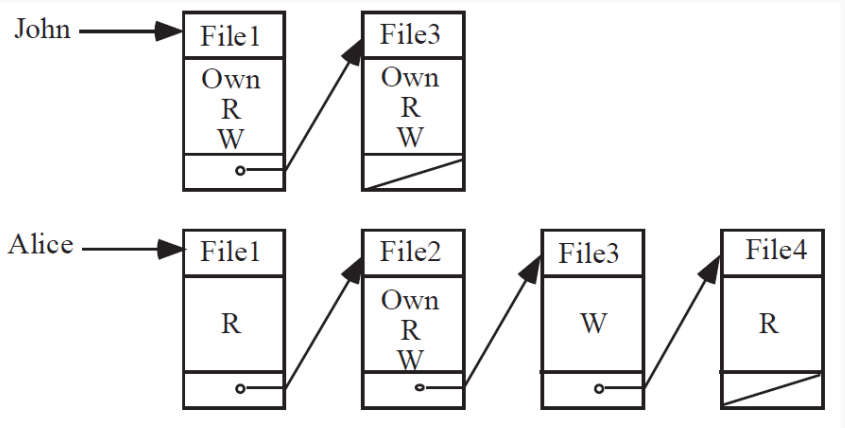
Capability Lists

- A dual approach to ACLs

Each subject is associated with a list (known as capability list)

- A capability list of a subject has a list of objects for which subject has some kind of access (like a ticket)
- corresponds to storing the access matrix by rows.

	File 1	File 2	File 3	File 4
John	Own R W		Own R W	
Alice	R	Own R W	W	R
Bob	R W	R		Own R W



Capabilities: two approaches

- Ticket is held by the OS, which returns to the subject a pointer to the ticket
- Ticket is held by the user, but protected from forgery by cryptographic mechanisms
  - Ticket can then be verified by the OS, or by the object itself

Authorisation Relations

Authorisation Relations: Each row or tuple of the authorisation relation specifies one access right of a subject to an object.

- e.g., `John`'s accesses to `File 1` require 3 rows.

- If the table is sorted by subjects, it reflects **Capabilities**.
- If the table is sorted by objects, it reflects **ACLs**.

The relation is not "normalised".

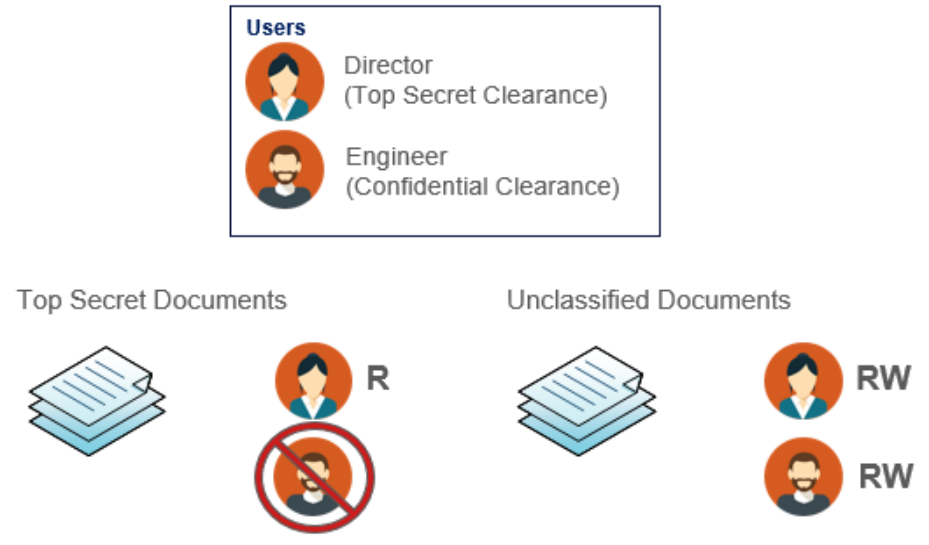
Subject	Access model	Object
John	Own	File 1
John	R	File 1
John	W	File 1
John	Own	File 3
John	R	File 3
John	W	File 3
Alice	R	File 1
Alice	Own	File 1
Alice	R	File 1
Alice	W	File 3
Alice	W	File 3
Alice	R	File 3
Bob	R	File 1
Bob	W	File 1
Bob	R	File 1
Bob	Own	File 3
Bob	R	File 3
Bob	W	File 3

Access Control Policies/Methodologies:

- Mandatory Access Control (MAC)
- Discretionary access Control (DAC)
- Role-Based Access Control (RBAC)
- Attribute-Based Access Control (ABAC)

Mandatory Access Control Policy

Access is based on the security level assigned to objects and subjects.



- The security level associated with object reflects the sensitivity of the information contained in the object.
- The security level associated with a subject (also called *clearance*) reflects the user's trustworthiness not to disclose sensitive information to uncleared users.

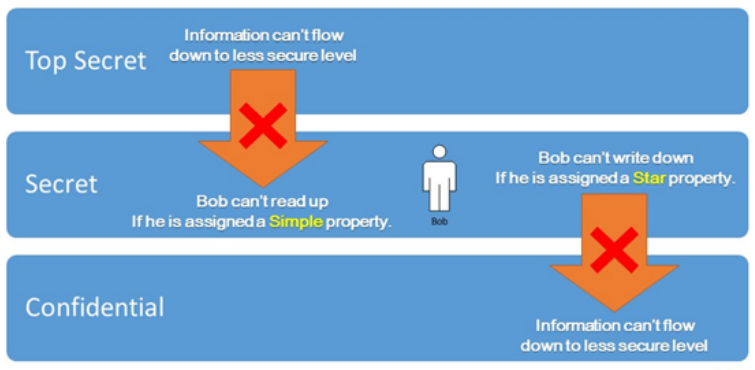
MAC Confidentiality Policies: Bell-LaPadula

Bell-LaPadula Confidentiality Model (DoD multilevel military security policy):

a subject's (usually a user's) access to an object (usually a file) is allowed or disallowed by comparing the object's security classification with the subject's security clearance.

The three basic rules are as follows:

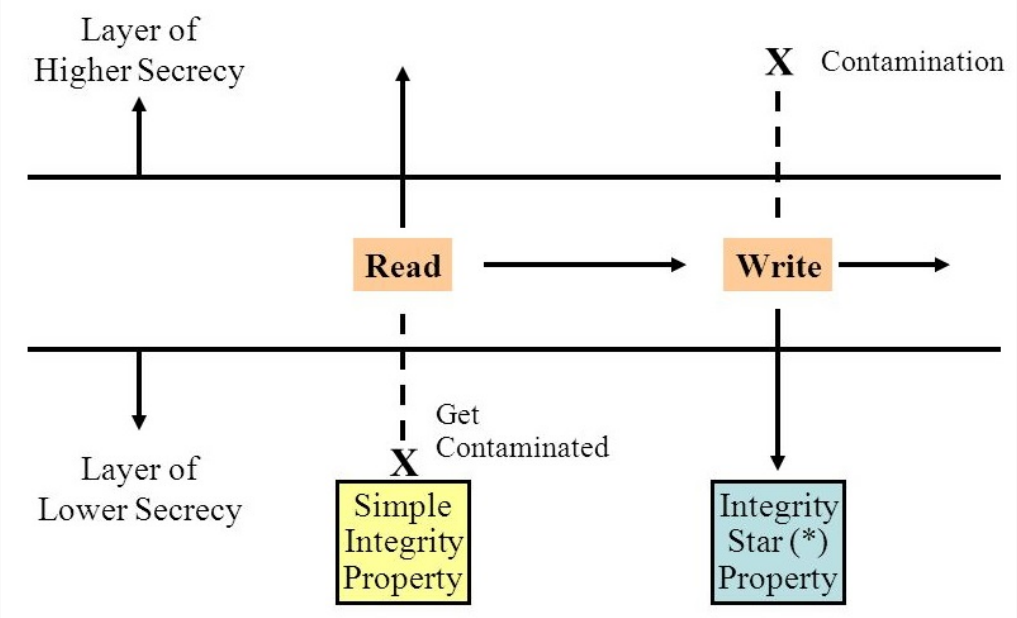
- The simple security condition: **No Read UP**
- The *-property (star property): **No Write DOWN**
- The tranquillity property: No changes while processing



MAC Integrity Policies: Biba

Biba Integrity Model, sometimes called the Bell-LaPadula upside down model:

- The simple integrity property: **No read DOWN**
- The integrity *-property: **No write UP**



Example of usage?

Issues with MAC

- Information tends to becomes over classified.
- No protection against violations that produce illegal information flow through indirect means.
  - **Inference Channels:** A user at a low security class uses the low data to infer information about high security class. Ex. Sudden assignment of low level soldiers to region could be an indication of a top secret mission.
  - **Covert channels:** Require two active agents, one at a low level and the other at a high level and an encoding scheme.

Implementation of MAC in Linux


SELinux — Security Enhancement to Linux


TOMOYO Linux — Lightweight and easy-use Mandatory Access Control


AppArmor — Linux Security Module for name-based access controls


grsecurity — Innovative set of patches for the Linux kernel

RSBAC — Patch adding several mandatory access models to the Linux kernel


Smack — The Simplified Mandatory Access Control Kernel

Discretionary Access Control (DAC) Policies

Access control is under the discretion of the Owner.

- Flexible
- Closed or open

Does not provide real assurance on the flow of information in the system:

- I create a file
- I give read permission to Mary because Mary is cool
- Mary likes Ben
- Mary creates a copy of the file and lets Ben read it
- But I don’t like Ben! :(

DAC: Example in Oracle Database

`WITH GRANT OPTION`:

Indicates that the principal (user/role) will also be given the ability to grant the specified permission to other principals.

```
GRANT [ALL {PRIVILEGES} | SELECT | INSERT |
UPDATE | DELETE] ON object TO [user | role |
PUBLIC] {WITH GRANT OPTION}
```

- $A \longrightarrow B \longrightarrow C$

A cannot revoke privileges from C.

Only the user who granted the privilege can revoke.

However, revoked privileges can “cascade”. A can revoke the privilege from B, that automatically revokes them from C.

DAC and Grant/Admin Options

`WITH ADMIN OPTION`:

We can give the system privileges only with admin option (CREATE TABLE, CREATE INDEX, CREATE SESSION, etc)

- $A \longrightarrow B \longrightarrow C$

```
GRANT CREATE INDEX TO Bob WITH ADMIN OPTION;
```

In admin option, it is possible for A to revoke the privileges from both B and C individually.

- Revoke the privileges from B ≠ Revoke the privileges of C.

Security Hole: "**`WITH ADMIN OPTION`**" are considered equal and can grant and revoke that privilege from anyone, including the person who granted it to them in the first place.

Role Based Access Control

Neither DAC nor MAC approaches satisfy the needs of most commercial enterprises.

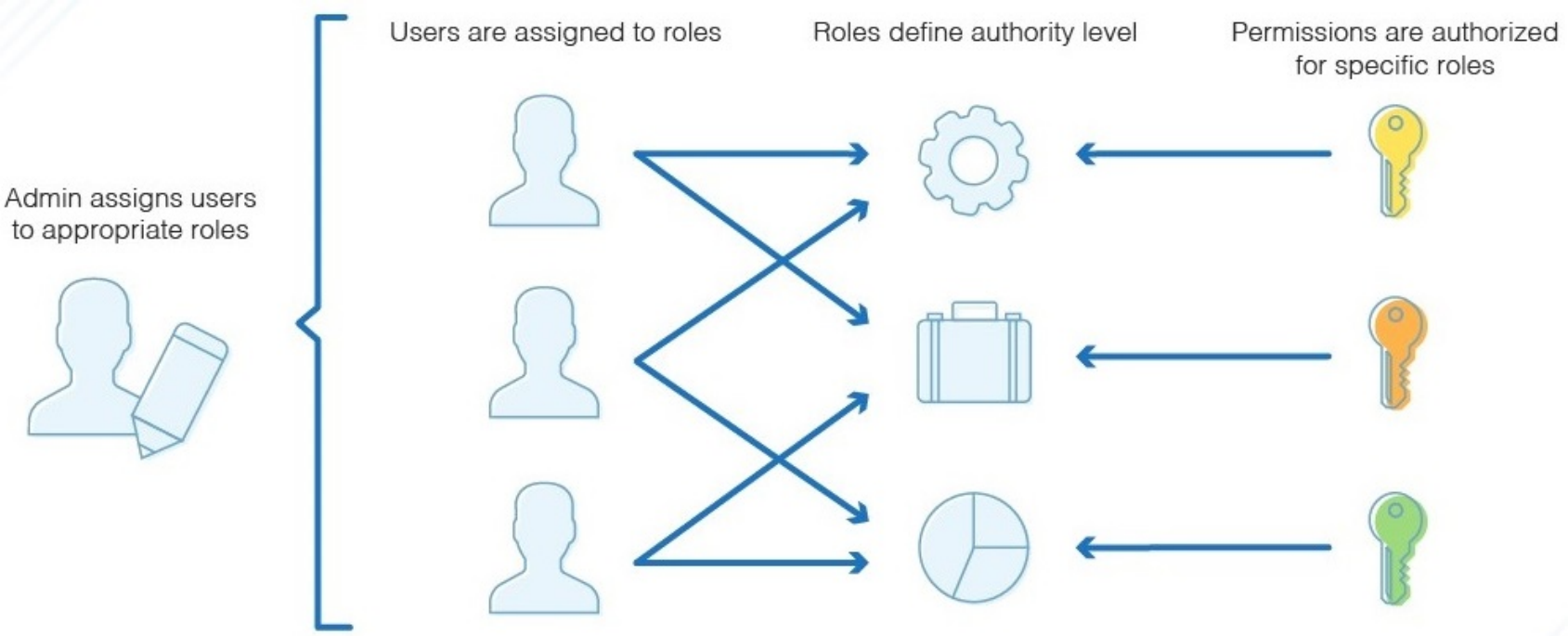
Mandatory policies are suitable for rigid environments such as military.

Discretionary policies come from cooperative yet autonomous environments, such as academia.

One alternative is Role-base Access Policies (RBAC).

Role-based Policies

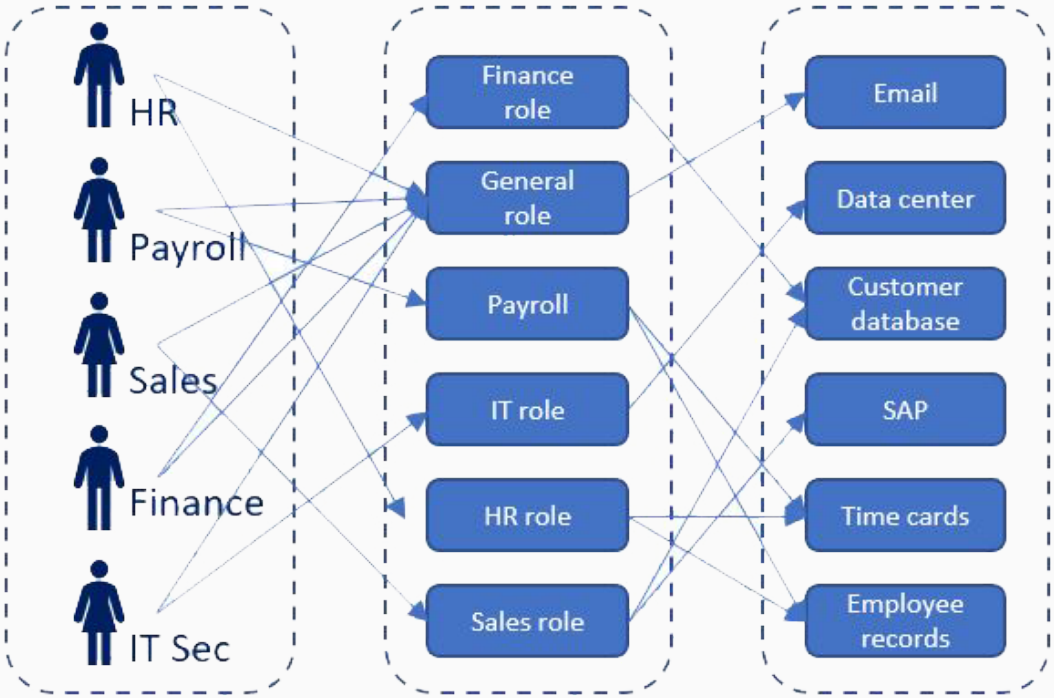
- A role is a set of actions, privileges and responsibilities associated with a particular working activity
- Instead of specifying all the accesses each user is allowed to execute, access authorisations are specified for roles



ref

Role-based Policies

- Users are given authorisation to adopt roles.
- A user playing a role is allowed to execute all accesses for which the role is authorised.
- User may or may not be allowed to play multiple roles at the same time.
- A user may take on different roles on different occasions.



RBAC Example – Oracle Database

- Doctor Role
  - CREATE ROLE **doctor**;
  - GRANT SELECT ON patient_info TO doctor;
  - GRANT SELECT ON med_history_rec TO doctor;
  - GRANT INSERT ON med_history_rec TO doctor;
  - GRANT UPDATE ON med_history_rec TO doctor;
  - GRANT **CONNECT** TO **doctor**;
  - GRANT **doctor** TO **Emily**, **Sam**;
- Nurse Role
  - CREATE ROLE nurse;
  - GRANT SELECT ON med_history_rec TO nurse;
  - GRANT UPDATE dosage ON med_history_rec TO nurse;
  - GRANT CONNECT TO nurse;
  - **GRANT nurse** TO **Masood**;

CONNECT is a role in Oracle and consists of system privilege "**CREATE SESSION**"

RBAC Example - Redhat Linux

```
[root@server ~]# kinit admin
[root@server ~]# ipa role-add --desc="User Administrator" useradmin
  ------------------------
  Added role "useradmin"
  ------------------------
  Role name: useradmin
  Description: User Administrator
```

Add the new role

```
[root@server ~]# ipa role-add-privilege --privileges="User Administrators"
useradmin
  Role name: useradmin
  Description: User Administrator
  Privileges: user administrators
  ----------------------------
  Number of privileges added 1
----------------------------
```

Add the required privileges to the role

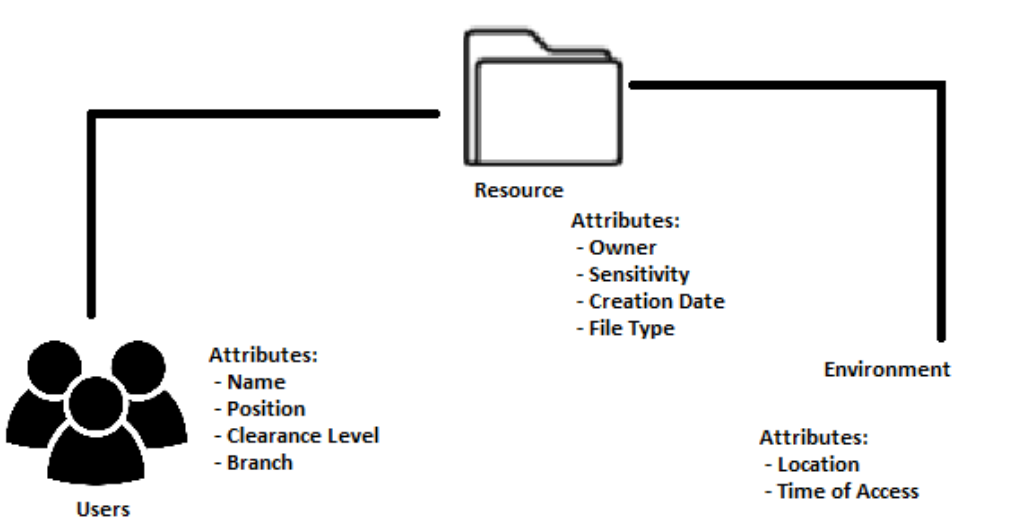
```
[root@server ~]# ipa role-add-member --groups=useradmins useradmin
  Role name: useradmin
  Description: User Administrator
  Member groups: useradmins
  Privileges: user administrators
  ------------------------
  Number of members added 1
------------------------
```

Add the required groups/users to the role

Attribute-Based Access Control (ABAC)

Access is granted/denied based on the corresponding attributes of the objects and subjects

- Subject attributes examples:
  - job title, security clearance, employment date, ...
- Object attributes examples:
  - location, access time, file type, creation date, ...



Pros & cons?

Next: Linux/UNIX File Access Control