Access Control Mechanisms Fundamentals

By Daniel Ellebæk

What is an Access Control Matrix?

“In computer science, an Access Control Matrix or Access Matrix is an abstract, formal security model of protection state in computer systems, that characterizes the rights of each subject with respect to every object in the system.”

by Butler W. Lampson in 1971

Which models?

- Security models
  - Bell La Padula
    - Code example
  - Biba
  - RBAC (Role Based Access Control)

Access Control Models

Security: Basic concepts

- Identification and authorization
  - Prove to the system who/what you are
    - Something you are (biometrics)
    - Something you have (key, token)
    - Something you know (PIN, password)
- Access control
  - Define what a user can do with a system-object
  - Access operations
  - Access rights

Exercise #1

What is needed to define access control?

5-10 minutes in groups

Access Control: Basic Concepts

- Terminology: who accesses what and how
  - Who? Subjects: users, processes, ...
  - What? Objects: files, memory, external devices, ...
  - How? Actions/operations (actions): read, write, copy, ...
- Things to consider
  - Can a program be a subject?
  - Can a program be an object?
  - Can a program be an action?
  - How would you define (non-) authorized access?
  - How would you implement/enforce access control?
  - Who define authorized access?

Access Control Matrix (DAC)

DAC (Discretionary Access Control)

Access Matrix

Eksempel

$S = \{\text{Alice}, \text{Bob}, \text{Charlie}\}$
$O = \{\text{a}, \text{b}, \text{c}\}$
$A = \{\texttt{read}, \texttt{write}, \texttt{exec}, \texttt{append}\}$

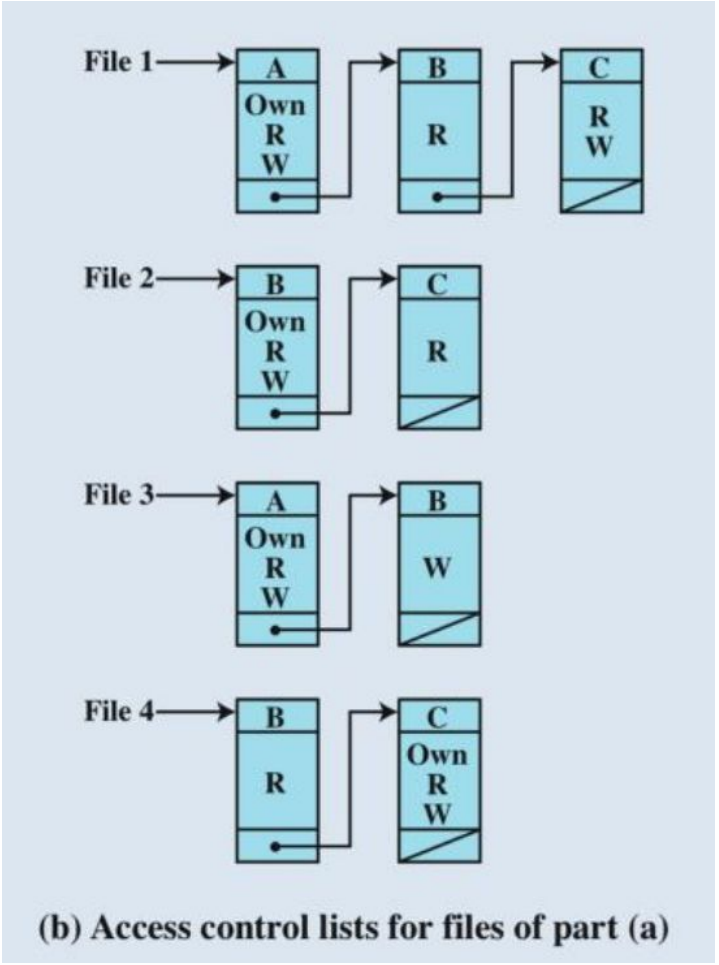
	a	b	c
Alice	`read, write`	$\emptyset$	`append`
Bob	$\emptyset$	`exec`	`read`
Charlie	`read`	`exec`	`append`

- Columns: Access control lists
- Rows: Capabilities

Linux uses DAC.

Access control lists

In computer security, an access-control list (ACL) is a list of permissions associated with a system resource (object).

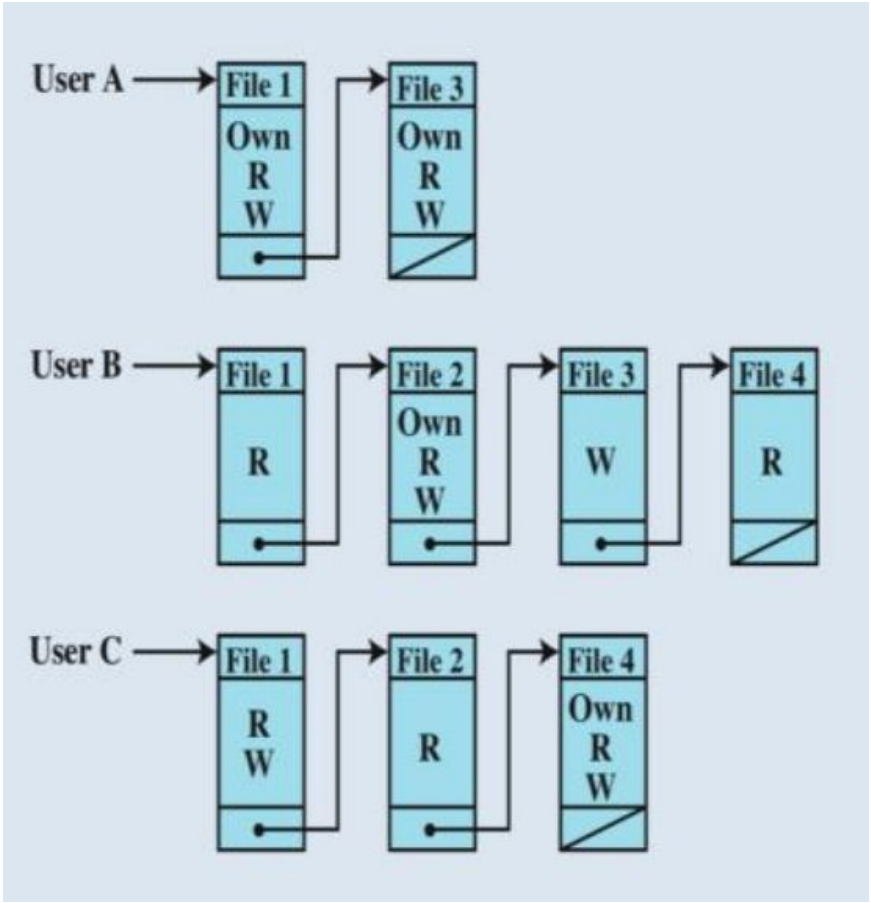


(b) Access control lists for files of part (a)

Capabilities

Capability-based security is a concept in the design of secure computing systems, one of the existing security models.

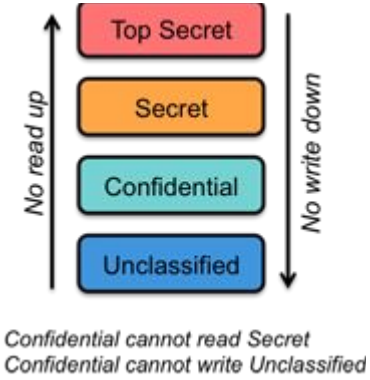
A capability (known in some systems as a key) is a communicable, unforgeable token of authority



Bell-LaPadula Confidentiality Model (Military use)

MAC (Mandatory Access Control)

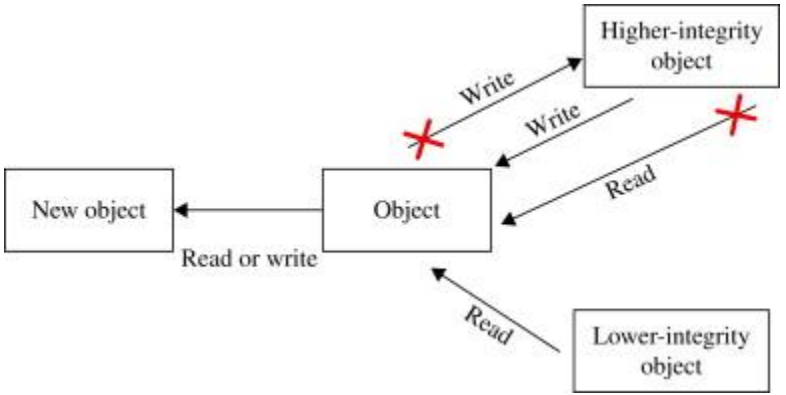
Confidentiality based security model



Is there any problems with this model?

Biba Integrity Model (military use)

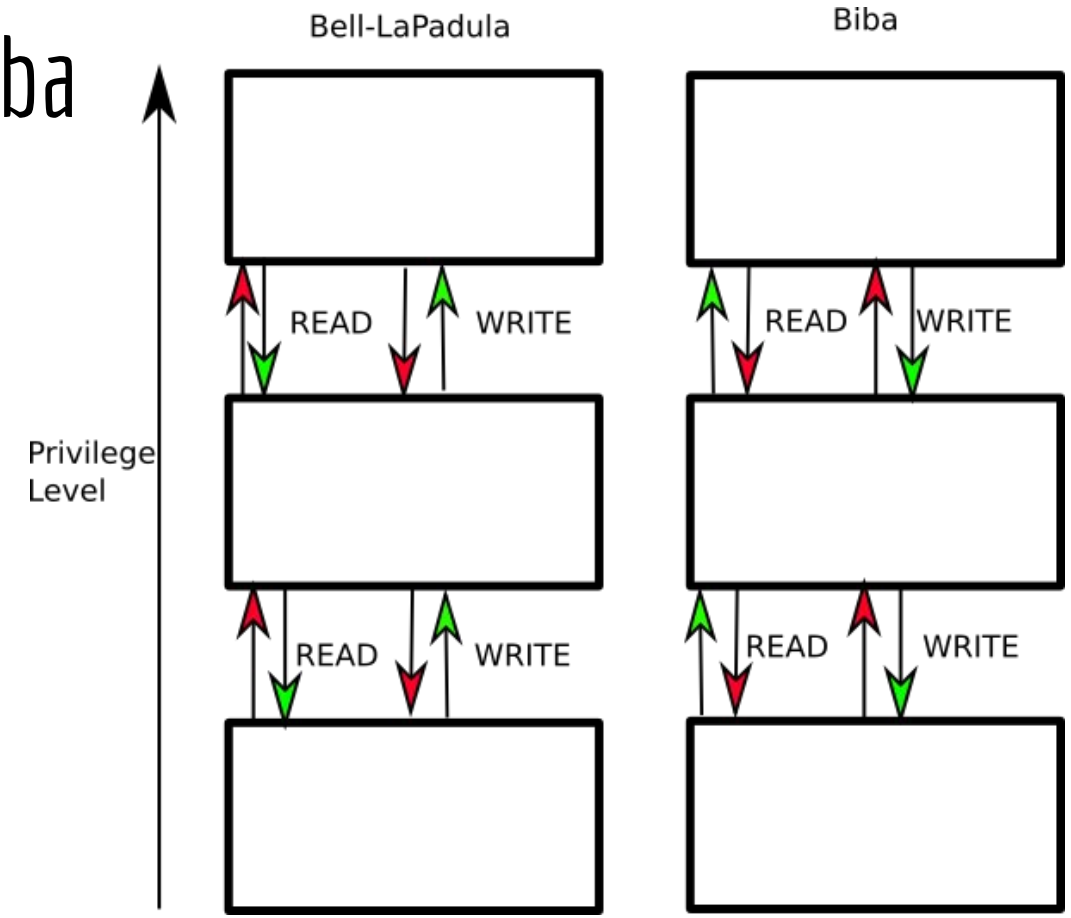
MAC (Mandatory Access Control)



Is there any problems with this model?

Prevents unauthorized users from making modifications (1st goal of integrity)

Bell-LaPadula vs Biba



MAC vs DAC

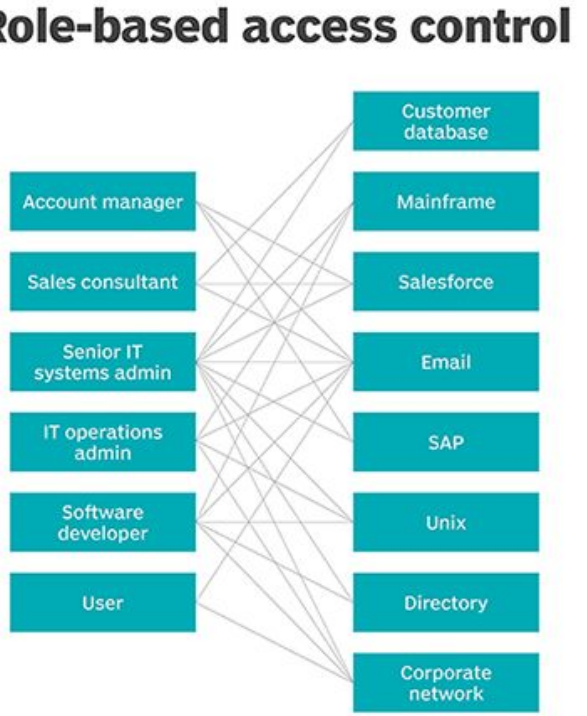
- MAC is more secure while DAC is more flexible
- DAC is easier to implement than MAC

Conclusion

The main difference between DAC and MAC is that the DAC is an access control method in which the owner of the resource determines the access while the MAC is an access control method that provides access to the resource depending on the clearance level of the user. In brief, DAC is suitable for systems that require general security while MAC is more suitable for systems that contain highly sensitive data.

Bell-LaPadula Java implementation example

RBAC (Role Based Access Control)



- Used a lot on web applications
- Easier to maintain
- Easier to give access on demand
- Giving administrators increased visibility
- Decreasing risk of breaches and data leakage
- Security: Access permissions are defined exclusively via the role model which prevents you from giving more permissions than needed to individual employees. This is in line with the Principle of Least Privilege (PoLP).
- Temporary assignments: If a user only needs extended access permissions temporarily, it is easier to forget about them when using RBAC than when assigning permissions individually
- Labor-intensive setup: Translating organizational structures into the RBAC model requires a lot of work.

RBAC on Web Applications

- Full access aka. Admin
- Normal user access

```
if user.is_admin:
   # allow full access
else:
   # allow user access
```

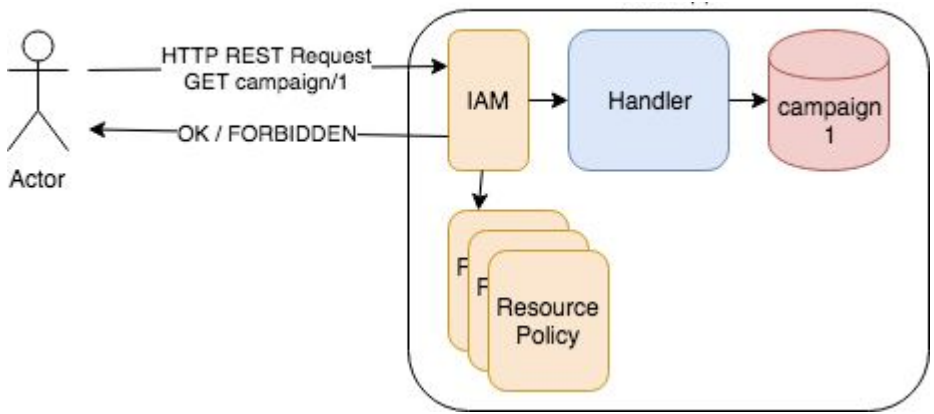
- We need a new profile: Read and Create
  - Hardcode it?
  - Smarter way?

```
if user.is_admin:
   # allow full access
elif user.has_read_access:
   # allow read access, deny create/update/delete
else:
   # allow user access
```

IAM (Identity Access Management)

Terminology

- **User:** An authenticated person or service logged in to the web application
- **Resource:** These are the things that we want to secure.
- **Permission**: A specific authorization to do something. Example: “edit”, “view” etc.
- **Resource Policy:** Define who has what permissions for what resources. Example: “Peggy” and “Peter” have “read” permission for “campaigns” or a specific campaign “1”, while “Joan” has “write permission.




```
{
 "read": [
   "Peggy",
   "Peter"
 ],
 "write": [
   "Joan"
 ]
}
```

Conclusion

- There are many access control mechanisms
- Choose wisely
- There are more wrong ways than correct ways