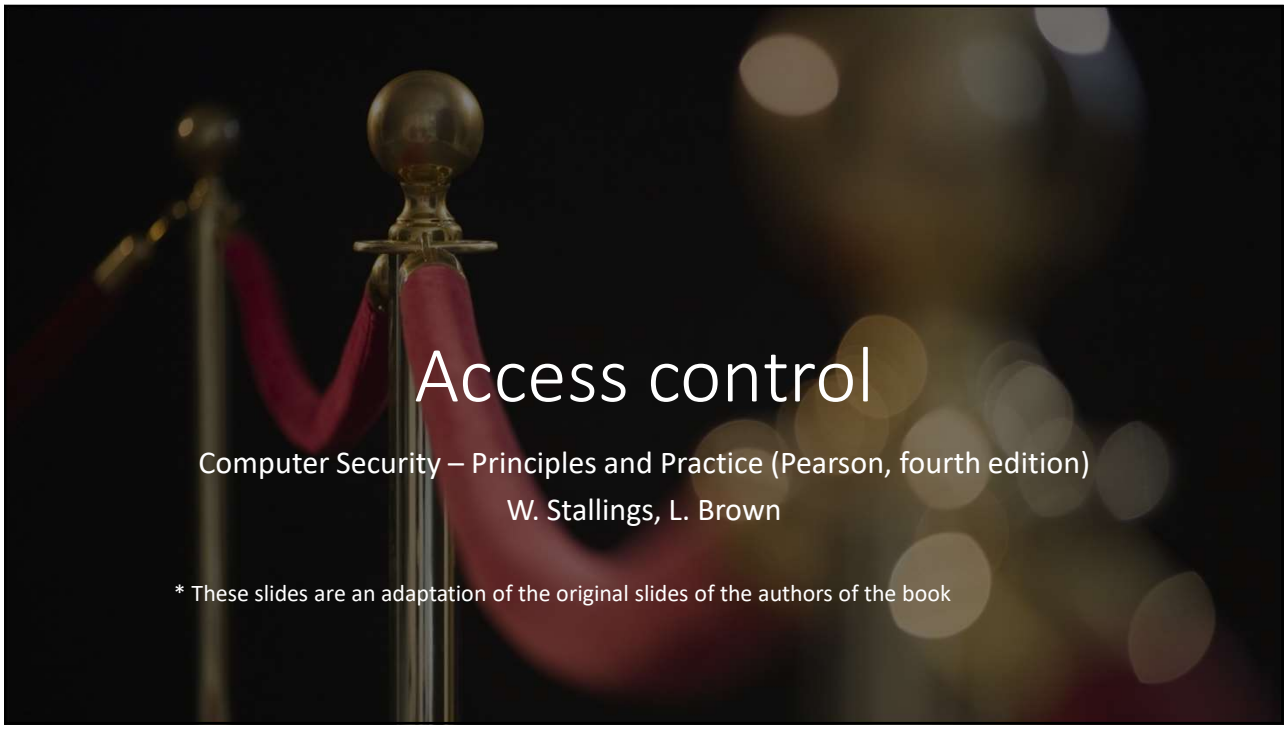


Access control

Computer Security – Principles and Practice (Pearson, fourth edition)

W. Stallings, L. Brown

* These slides are an adaptation of the original slides of the authors of the book

1

Learning objectives

- Explain how access control fits into the broader context that includes authentication, authorization, and audit.
- Define the three major categories of access control policies.
- Distinguish among subjects, objects, and access rights.
- Describe the UNIX file access control model.
- Discuss the principal concepts of role-based access control.
- Summarize the role-based access control model.
- Discuss the principal concepts of attribute-based access control.
- Explain the identity, credential, and access management model.
- Understand the concept of identity federation and its relationship to a trust framework.

2

Access Control Definitions 1/2

NISTIR 7298 defines access control as:

"the process of granting or denying specific requests to:
(1) obtain and use information and related information processing services; and
(2) enter specific physical facilities"

NIST

NOT SPECIFIC TO COMPUTER SECURITY

3

Access Control Definitions 2/2

RFC 4949 defines access control as:

"a process by which use of system resources is regulated according to a security policy and is permitted only by authorized entities (users, programs, processes*, or other systems) according to that policy"

Authorization here: we verifying identity of a user and we authorize them to do something

* Proxies of the system

4

Access Control Security Requirements (SP 800-171)

CUI : controlled unclassified information

Basic Security Requirements

1. Limit information system access to authorized users, processes acting on behalf of authorized users, or devices (including other information systems).
2. Limit information system access to the types of transactions and functions that authorized users are permitted to execute

Derived Security Requirements

3. **Control the flow of CUI** (controlled unclassified information) in accordance with approved authorizations.
4. **Separate the duties of individuals** to reduce the risk of malevolent activity without collusion.
5. Employ the principle of **least privilege**, including for specific security functions and privileged accounts.
6. **Use non-privileged accounts or roles** when accessing non-security functions.
7. Prevent non-privileged users from executing privileged functions and audit the execution of such functions.
8. Limit unsuccessful logon attempts.
9. Provide privacy and security notices consistent with applicable CUI rules.
10. Use session lock with pattern-hiding displays to prevent access and viewing of data after period of inactivity.

5

Access Control Security Requirements (SP 800-171)

CUI : controlled unclassified information

11. Terminate (automatically) a user session after a defined condition.
12. Monitor and control remote access sessions.
13. Employ cryptographic mechanisms to protect the confidentiality of remote access sessions.
14. Route remote access via managed access control points.
15. Authorize remote execution of privileged commands and remote access to security-relevant information.
16. Authorize **wireless access** prior to allowing such connections.
17. Protect wireless access using authentication and encryption.
18. Control connection of mobile devices.
19. **Encrypt CUI** on mobile devices.
20. Verify and control/limit connections to and use of external information systems.
21. Limit use of organizational **portable storage devices** on external information systems.
22. Control CUI posted or processed on publicly accessible information systems.

6

Access Control Principles

- In a broad sense, all of computer security is concerned with access control
- RFC 4949 defines computer security as:

"measures that implement and assure security services in a computer system, particularly those that assure access control service"

7

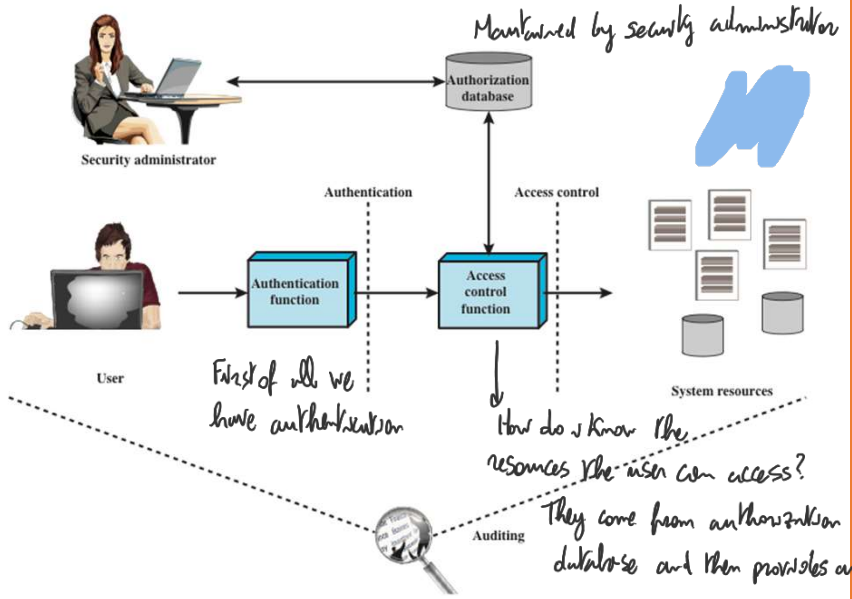
Access control & computer security

In our case we consider:

- We address a specific concept of access control:
  - implements a security policy that
  - specifies who (or what) may have access to each specific system resource,
  - and the type of access that is permitted in each instance

8

Access control and other security functions



9

Access control and other security functions

In addition to access control, this context involves the following entities and functions:

- **Authentication**: Verification that the credentials of a user or other system entity are valid.
- **Authorization:** The granting of a right or permission to a system entity to access a system resource. This function determines who is trusted for a given purpose.
- **Audit:** An independent review and examination:
  - checks the adequacy of system controls,
  - ensures compliance with established policy and operational procedures,
  - detects breaches in security,
  - recommend any indicated changes in control, policy and procedures.

10

Access control policies

Classify them in 4 classes

MOST IMPORTANT

- Discretionary access control (DAC)
  - Controls access based on the **identity** of the requestor and on **access rules** (authorizations) stating what requestors are (or are not) allowed to do
- Mandatory access control (MAC)
  - Controls access based on comparing security labels with security clearances

Inspired by military areas. Used in combination with other policies

- Role-based access control (RBAC)
  - Controls access based on the **roles** that users have within the system and on **rules** stating what accesses are allowed to users in given roles
- Attribute-based access control (ABAC)
  - Controls access based on **attributes** of the user, the **resource** to be accessed, and current **environmental** conditions

11

① Based on user id. Give the id it checks the privileges and accesses the results and decides. Very simple and efficient. Used in OS.

② Based on roles. Authoriz. not based on id but on roles the user is impersonating anytime. There is association between user and roles. After authentication the system knows which role the user is acting.

Subjects, objects, and access rights

Subject

Active component

An entity capable of accessing objects

Three classes
- Owner
- Group
- World

Unix

Object

subject to access control

A resource to which access is controlled

Entity used to contain and/or receive information

→ Granularity (bytes, files, I/O)

↓

What are the smallest objects subject to access control (in a OS might be files, in a database single lines of a table)

Access right

Describes the way in which a subject may access an object

Could include:
- Read, Write, Execute
- Delete, Create
- Search

12

Granularity is a decision that the access control manager needs to take.

② Very popular in databases.

③ Most flexible one. [NOT MORE COMPLEX!] Makes use of functions but take decisions based on different values: attributes of the user, resource and environmental conditions. The execution of this function may be more complex than just looking up. But gives us a lot of flexibility that allow us to implement complex policies easily, that would otherwise be very complex with simpler approaches. So drawback: if the policy is simple you can do it in a more efficient way with DAC and RBAC.

④ Users are associated to clearances and objects are associated to security labels. If you have a certain level of clearance you cannot access objects with a certain sec. label.

Authorization phase: catching with the mouth. Authentication and access control happen at the same time.

Subjects, objects and access rights

Subject:
- Generally, the concept of subject equates with that of process.
  - A user/application accesses objects by means of a process that represents it.
  - The process takes on the attributes of the user/application, such as access rights.
- It is held accountable for its actions,
- an audit trail may be used to record the association of a subject with security relevant actions performed on an object by the subject.

Three classes of subjects:
- **Owner**: The creator of a resource (a system administrator or a project leader…)
- **Group:** Membership in the group lets to exercise some access rights. A user may belong to multiple groups.
- **World:** Users that are able to access the system anyway. Grants the least amount of access.

Object:
- Contains and/or receives information:
  - records, blocks, pages, files, portions of files, directories, messages, programs etc.
  - may also be: bits, bytes, words, processors, ports, clocks, etc.
- Number and types depends on:
  - the context in which access control operates
  - the tradeoff between security vs complexity, processing burden, and ease of use.

Access right:
- **Read:** view information in a system resource. Includes the ability to copy or print.
- **Write**: add, modify, or delete data in system resource. Includes read access.
- **Execute**: execute specified programs.
- **Delete:** delete certain system resources, such as files or records.
- **Create:** create new files, fields, etc.
- **Search:** list the files in a directory or otherwise search the directory.

13

Most popular access control method

Discretionary access control (DAC)

14

Discretionary Access Control (DAC)

Possibility for a subject to grant a permission over an object. Of course you must own it, and you can do it anytime without any restriction.

- Scheme in which an entity may be granted access rights that permit the entity, by its own volition, to enable another entity to access some resource
- Often provided using an access matrix *
  - One dimension consists of identified subjects that may attempt data access to the resources
  - The other dimension lists the objects that may be accessed
- Each entry in the matrix indicates the access rights of a particular subject for a particular object

* If empty, no permission at all.

15 We use a module that inspects the data structure to make a decision.
*In theory

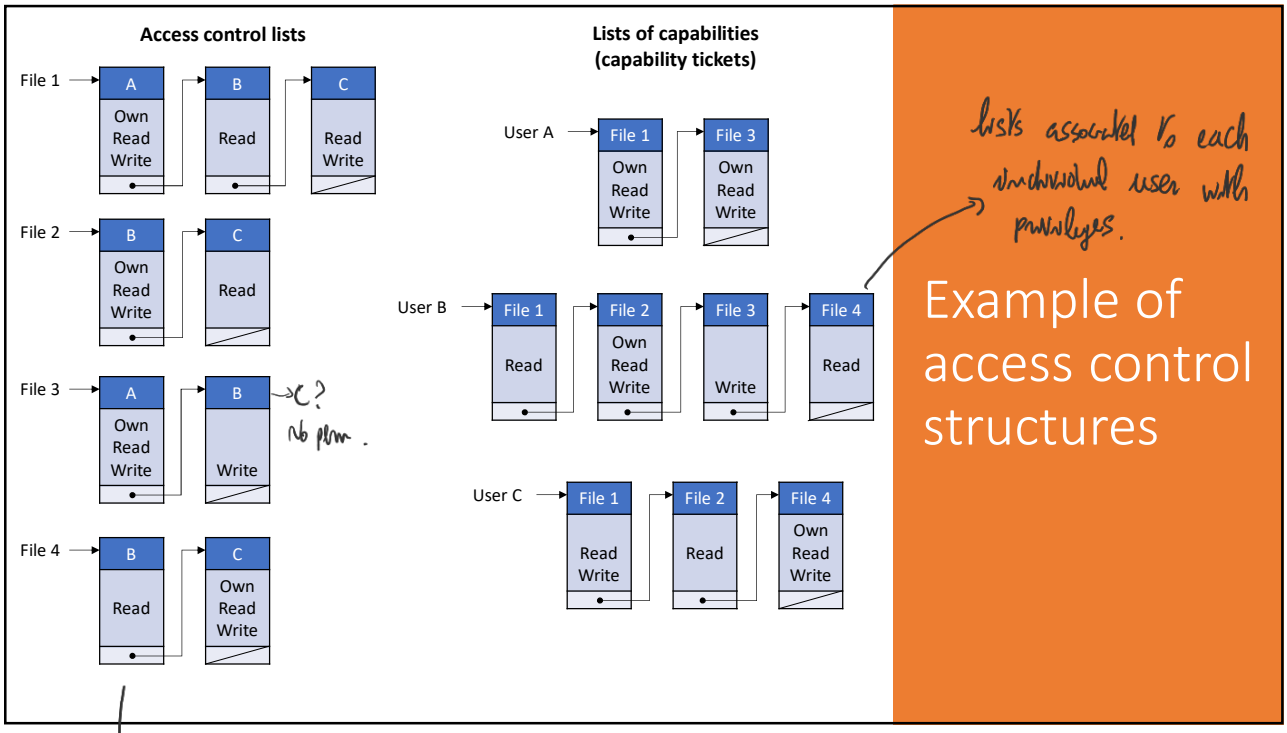
Access matrix

Objects

Subjects	File 1	File 2	File 3	File 4
USER A	Own Read Write		Own Read Write	
USER B	Read	Own Read Write	Write	Read
USER C	Read Write	Read		Own Read Write

16 Very easy to use and easy for lookup. It should be efficient. Decision very fast. Matrix is indeed efficient, but not memory efficient. In general it is huge and most of the subjects do not have permissions. So wasting a lot of time and memory

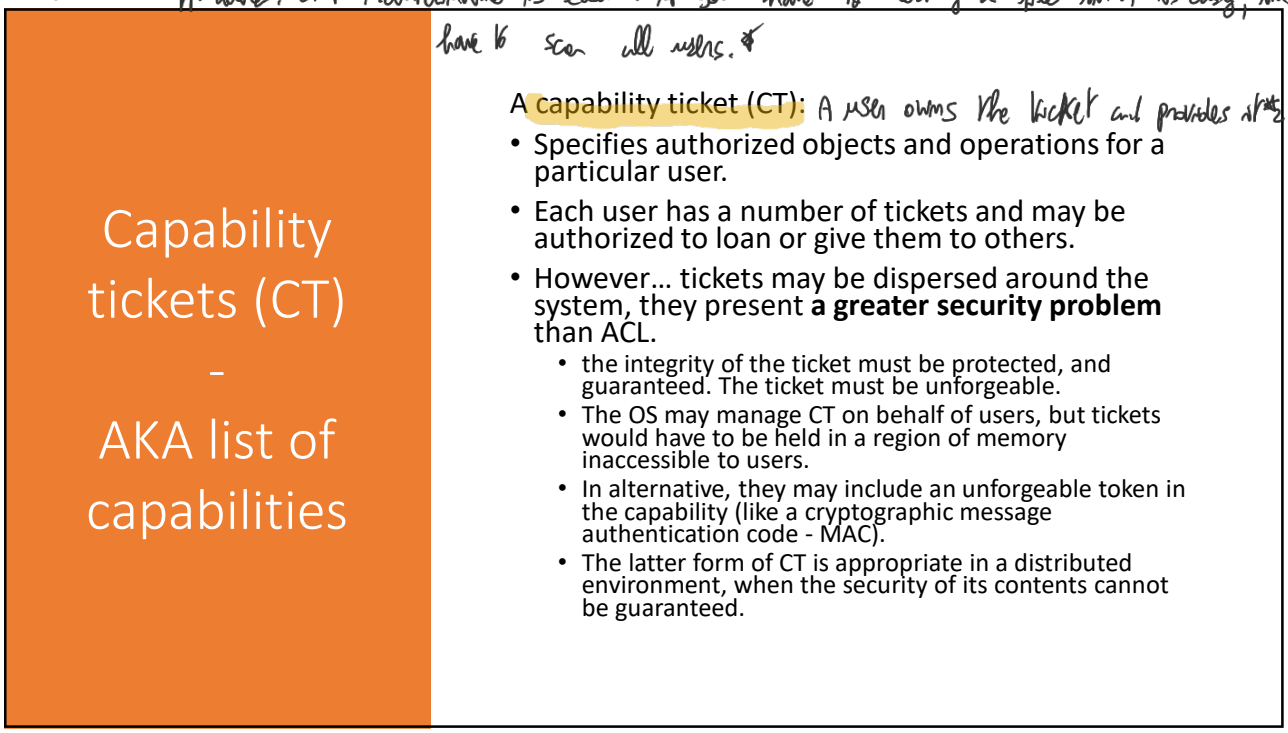
Alternative structures: sets of list that contain equivalent info in a more compact form.



17

We store a list of subjects who have access to what object and with which permissions.

Difference: from the access control POV, not much difference: input of subject, object and permission. Takes a bit more time but more affordable. But Maintenance is easier: if you have to destroy a file in A is easy, in B you have to scan all users.



18

* This also if you want to edit all permissions on a file. So most OS adopt A, while B are restricted to only some tasks.

CONCERNS FOR THE 3: Where are these Data S. stored? They are critical. They must be protected and its access will be mediated by the access control system itself. They are stored within the OS and access is mediated with system calls.

*2 They may be used in a network communication and there it become a problem. You have to make sure of confident. and integrity. So they could be protected by a MAC.
NOTE: in the context of DAC, every user can give permissions here and there and cap. tickets are sent here and there, so revoking permission can be hard (with A you control the data structure and it is stored).

Authorization Table

Authorization table: more convenient than either ACLs or CT

- Sorted by subject gives CT
- Sorted by object gives ACL
- Easy to implement in a relational database

Subject	Access Mode	Object
A	Own File	File 1
A	Read File	File 1
A	Write File	File 1
A	Own File	File 3
A	Read File	File 3
A	Write File	File 3
B	Read File	File 1
B	Own File	File 2
B	Read File	File 2
B	Write File	File 2
B	Write File	File 3
B	Read File	File 4
C	Read File	File 1
C	Write File	File 1
C	Read File	File 2
C	Own File	File 4
C	Read File	File 4
C	Write File	File 4

19 Represent a table with subject and Object, with permission info.

Access control models for DAC

- A model for DAC developed by Lampson, Graham, Denning (1971/72)
- Protection state of a system: We need to represent a
  - the set of information, at a given time, that specifies the access rights for each subject with respect to each object

Valid at a given time

- Requirements:
  - representing the protection state
  - enforcing access rights
  - allowing subjects to alter the protection state in certain ways

↳ for this they subjected to modify the concept of object

With DAC

20 Problem not only implementing AC by itself. You have to also maintain, update or change the data structure. Model adopted was this:

DAC: representing the protection state

- Extends the universe of objects:
  - **Processes**: Access rights include the ability to delete a process, stop (block), and wake up a process.
  - **Devices**: Access rights include the ability to read/write the device, to control its operation (e.g., a disk seek), and to block/unblock the device for use.
  - **Memory locations or regions**: Access rights include the ability to read/write certain regions of memory that are protected such that the default is to disallow access.
  - **Subjects**: Access rights with respect to a subject have to do with the ability to grant or delete access rights of that subject to other objects, as explained subsequently

21

Matrix is now extended

Objects

Subjects	S_1	S_2	S_3	F_1	F_2	P_1	P_2	D_1	D_2
	subjects			files		processes		disk drives	
S_1	control	owner	owner control	read *	read owner	wakeup	wakeup	seek	owner
S_2		control		write *	execute			owner	seek *
S_3			control		write	stop			

* - copy flag set

A subject may create a new subject and/or modify your permissions.

The subject can also modify its own permissions.

S_1 created S_2.

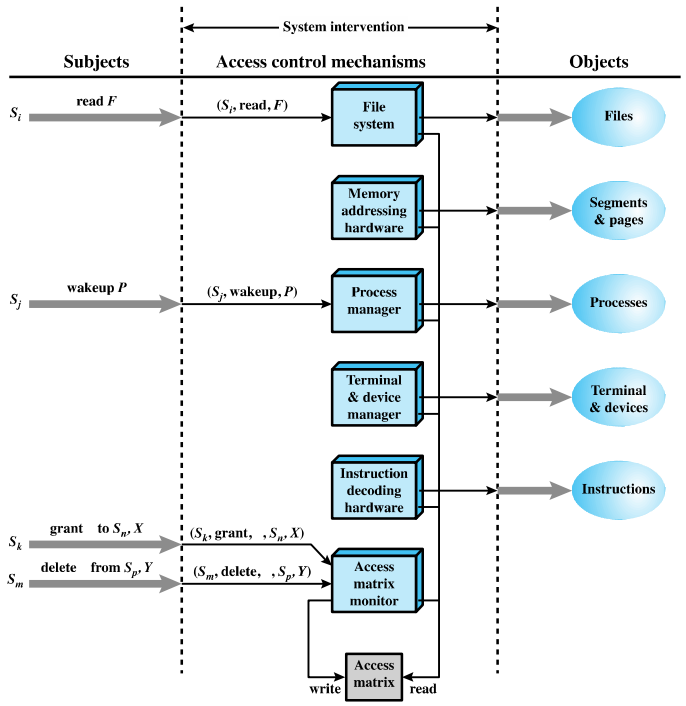
DAC: representing the protection state – extended access control matrix

22

DAC: enforcing access rights

- This model assumes a "separate" access control module for each object
- An access triggers the following steps:
  1. A subject S_0 issues a request of type α for object X.
  2. The request causes the system to generate a message of the form (S_0, α, X) to the controller for X.
  3. The controller interrogates the access matrix A to determine if α is in $A[S_0, X]$: If so, the access is allowed; if not, the access is denied.

23



DAC: an organization of the access control function

24

NOTE: Some requests are explicit, like S_i, S_j. Not all operations that imply access to an object are explicit. Imagine a process reading a value in the memory. There is still an AC action, at a hardware level for this example.

2. There are specific AC modules to modify the access matrix, they go to the ACmonitor, it checks if

Operation is permitted and changes the values.

DAC: enforcing access rights

- The model also includes rules for the modification of the access matrix
- Need for "owner" and "control" access rights:
  - To change the rights and the structure of the access matrix
- Need for "copy flag" concept
  - To give a right to another subject

To grant a permission, you need to have a "copy flag"

25

To allow transfer, the user needs the transferring permission + copy flag

Access control system commands

Subject V_0 owns R_1/R_3

- Transferring, granting and deleting access rights
- Reading the access matrix
- Manipulating objects and subjects

Rule	Command (by S_0)	Authorization	Operation
R1	**transfer** $\begin{Bmatrix}\alpha^* \\ \alpha\end{Bmatrix}$ **to** S, X	'α^*' in $A[S_0, X]$	store $\begin{Bmatrix}\alpha^* \\ \alpha\end{Bmatrix}$ in $A[S, X]$
R2	**grant** $\begin{Bmatrix}\alpha^* \\ \alpha\end{Bmatrix}$ **to** S, X	'owner' in $A[S_0, X]$	store $\begin{Bmatrix}\alpha^* \\ \alpha\end{Bmatrix}$ in $A[S, X]$
R3	**delete** α **from** S, X	'control' in $A[S_0, S]$ or 'owner' in $A[S_0, X]$	delete α in $A[S, X]$
R4	$w \leftarrow$ **read** S, X	'control' in $A[S_0, S]$ or 'owner' in $A[S_0, X]$	copy $A[S, X]$ into w
R5	**create object** X	none	add column for X to A; store 'owner' in $A[S_0, X]$
R6	**destroy object** X	'owner' in $A[S_0, X]$	delete column for X from A
R7	**create subject** S	none	add row for S to A; execute **create object** S; store 'owner' in $A[S_0, S]$; store 'control' in $A[S, S]$
R8	**destroy subject** S	'owner' in $A[S_0, S]$	delete row for S from A; execute **destroy object** S

26

- To transfer you have to own the permission. If you own a given object you can also not have a permission on that object but still grant it.
- Delete permission (* makes no sense) only if we control or own that subject. A s

R7: If I am S and want to create S, I do not need auth. It means adding a row to the matrix and adding an object S. Automatically store "control" and "owner".
If you are the owner of a subject you can give permission to control a subject.

R4: S can read the matrix to see the permissions of S over X

R8: you are not the owner of yourself so you cannot destroy yourself.

Extra with PD:

Protection domains

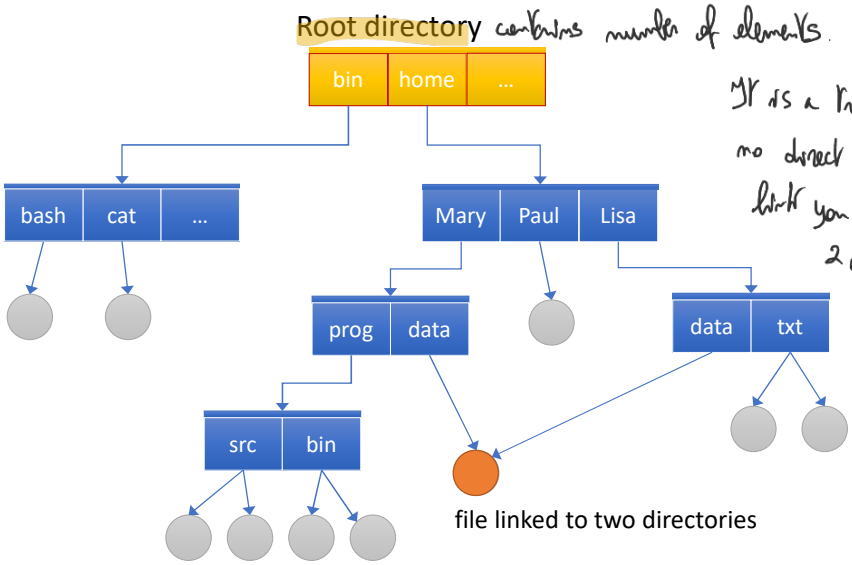
- So far, a subject is a user, and a row in the access control matrix represents its capabilities
- More flexibility when associating capabilities with protection domains:
  - Set of objects together with access rights to those objects
  - In terms of the access matrix, a row defines a protection domain
- Flexibility:
  - User can spawn processes with a subset of the access rights of the user
  - Association between a process and a domain can be static or dynamic
- An example in Unix:
  - In user mode certain areas of memory are protected from use and certain instructions may not be executed
  - In kernel mode privileged instructions may be executed and protected areas of memory may be accessed

27

Short recap from Operating Systems class…

28

File System: tree structure



29

Some directories in Unix

/: it's the root directory

/bin: binary executable files. These are utilities of the system

/boot: everything required to boot the system

/dev: contains the device files («everything is a file in Unix»…)

- These are not regular files; accessing these files means accessing the respective system devices (drives, serial lines, etc…)

/etc: contains system administration and configuration files (the file of passwords is here)

/home: contains the users' home directories

/lib: contains kernel modules and the shared libraries

/mnt: generic point under which to mount a file system or a device

/opt: contains add-on packages not part of the defauls installation

/proc: is very special in that it is also a virtual filesystem (aka process information pseudo-file system):

- It doesn't contain 'real' files but runtime system information (e.g. system memory, devices mounted, hardware configuration, etc).
- it can be regarded as a control and information centre for the kernel.
- In fact, quite a lot of system utilities are simply calls to files in this directory.
- By altering files located in this directory you can even read/change kernel parameters (sysctl) while the system is running.

/tmp: contains temporary files;

/usr: contains «everything user-related», for example programs for users (not system programs).

- **/usr/bin**: programs for the user (included games)
- **/usr/lib**: libraries for user programs

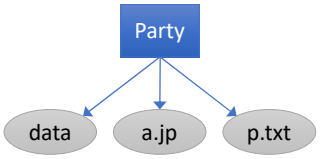
30

Data structures: Directories

- Each file (and directory) belongs to a directory;
- Each directory is a data structure that links file names to file attributes
  - Example of attributes: file size, address on disk, access rights, time of last access, time of creation, …

A possible implementation (FAT file systems):

- The directory is a table, it associates each file name to its file descriptor (which includes all attributes of the file)



name	descriptor
data	Attribute: type, address, etc.
a.jp	Attribute: type, address, etc.
p.txt	Attribute: type, address, etc.

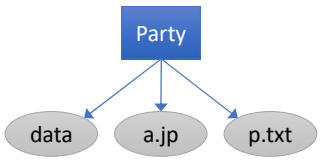
Implementation of directory Party

31

Data structures: Directories

Implementation of directories in Unix:

- The directory is a table that includes the references to the file descriptors (i-nodes), which are stored in a separate data structure in the disk



name	pointer
data	
a.jp	
p.txt	

Implementation of directory Party

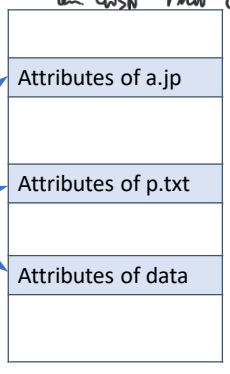


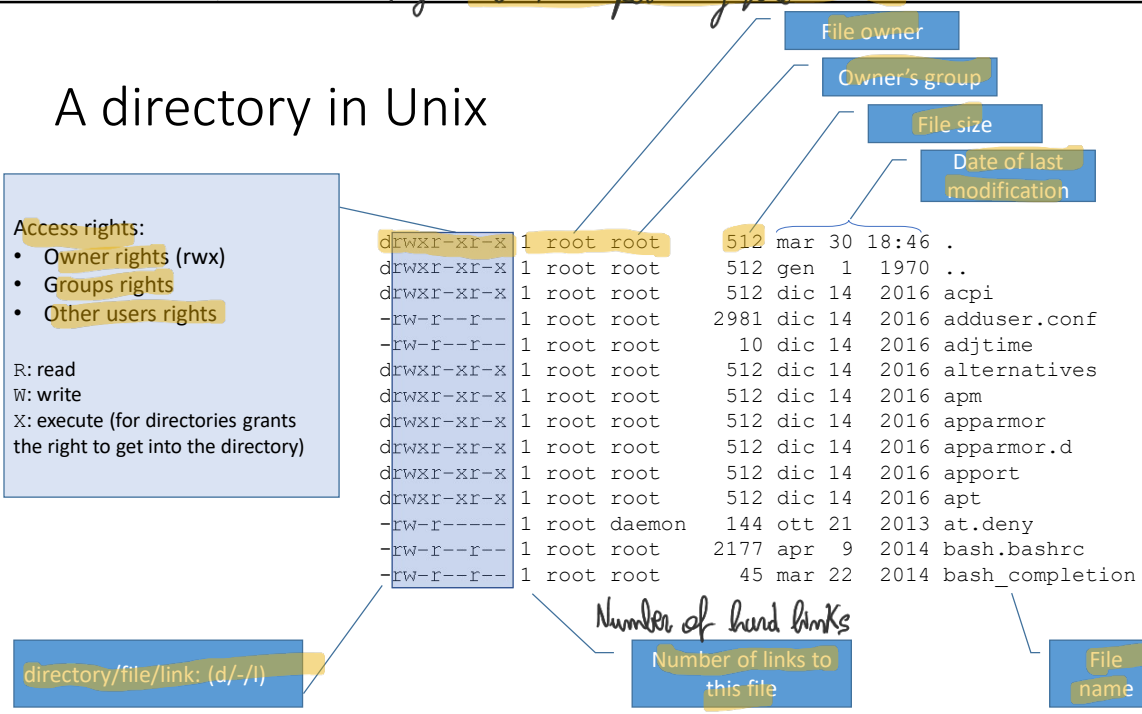
Table (i-list) of file descriptors (i-nodes)

Pointer to a data structure stored on disk that contains the descriptor of a file.

32 All the data structures used by ACCESS CONTROL must be protected. The access to those data must be mediated by access control. I-nodes are not visible by any user. Accessible with some system calls.

List a directory: one line for every file or dir

A directory in Unix



Access control list for a file uses 9 bits for 3 classes of users: owner, group, world.

Methods to create or destroy users and change privileges.

Access rights

creation of users and groups (and deletion) reserved to the `root` user, by means of:

- `useradd / userdel`
- `groupadd / groupdel`

to change permissions to files and directories:

- `chmod [a,u,g,o] [+,-] [r,w,x] file_name`: adding or revoking a certain privilege
- give (+) or revokes (−) rights `[r,w,x]` to all (`a`) owner (`u`), group (`g`), others (`o`) to the file `file_name`

To change the owner or the group of a file:

- `chown new_owner file_name` (change owner)
- `chgrp new_group file_name` (change group)

Berkeley UNIX FFS (Fast File System)

inode table

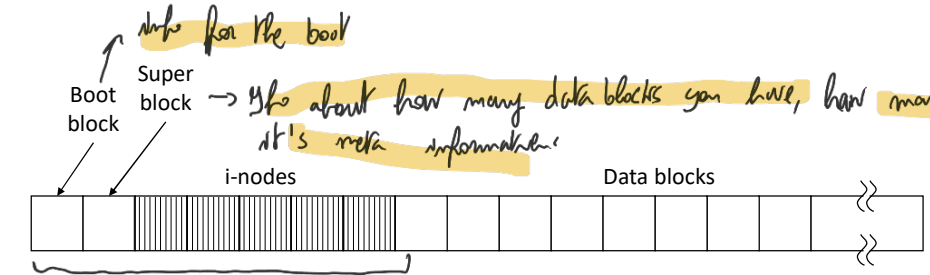
- A table in the file system that contains all i-nodes

inode

- One per file
- Metadata:
  - File owner, access permissions, access times, …
- Set of pointers to data blocks
  - A asymmetric tree structure (see OS class…)

35

Physical disk organization in UNIX



36

Unix file access control

SUMMARY

UNIX files are administered using inodes (index nodes)

- Control structures with key information needed for a particular file
- Several file names may be associated with a single inode
- An active inode is associated with exactly one file
- File attributes, permissions and control information are sorted in the inode
- On the disk there is an inode table, or inode list, that contains the inodes of all the files in the file system
- When a file is opened its inode is brought into main memory and stored in a memory resident inode table

Directories are structured in a hierarchical tree

- May contain files and/or other directories
- Contains file names plus pointers to associated inodes

37

Traditional Unix file access control

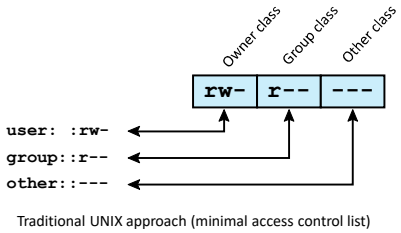
What are the perms?

A user: Subject is represented by UID

- Unique user identification number (user ID)
- Member of a primary group identified by a group ID
- May also belong to several other groups

A file (an object):

- Has an owner
- 12 protection bits
  - Specify read, write, and execute permission for the owner of the file, members of the group and all other users
- The owner ID, group ID, and protection bits are part of the file's inode



Traditional UNIX approach (minimal access control list)

3 triples associated here.

38

Traditional Unix file access control

Interpretation of these bits: over files okay, what does it mean to write or execute a directory?

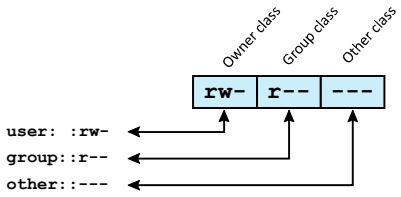
A hierarchy:

owner – group – all others

Concerning directories:

- Read grants the list capability
- Write grants the create / rename / delete files capability
- Execute grants the ability to descend into subdirectories

↓ enter into a subdir



Traditional UNIX approach (minimal access control list)

39

Traditional UNIX file access control

- "Set user ID"(SetUID) and "Set group ID"(SetGID)
  - System temporarily uses rights of the file owner/group in addition to the real user's rights when making access control decisions
  - Enables privileged programs to access files/resources not generally accessible
- Sticky bit (Applies to dirs)
  - When applied to a directory it specifies that only the owner of any file in the directory can rename, move, or delete that file

Important for the shared /tmp dir: can only put files inside.

- Superuser
  - A specific User ID
  - Is exempt from usual access control restrictions
  - Has system-wide access

≅ root

40

You don't want user to access the PW file! But how can you change your own password? There's a program that lets you change the PW. To do that you need to access the file.

When you execute a program, sys creates a process which your id associated. The exe has the SetUID flag: the process will take the credentials of the owner of the file to be executed. This is secured because it is the only way to interact with the PW file. In some cases to do something you need to have higher privileges. You can set the setUID to 1 only if you are the owner of the file or if you have permission.

SetGroupID, the program will run with the same groupID of the owner of the file

That was the original UNIX. Now it evolved. We now have full ACL lists, which are normally not visible because they are not needed.

Access control lists (ACLs) in Unix

Modern UNIX systems support ACLs

- FreeBSD, OpenBSD, Linux, Solaris

FreeBSD Admin can create an ACL and can be

- The administrator can assign a list of UNIX user IDs and groups by means of the Setfacl command
- Any number of users and groups can be associated with a file
  - Each with read, write, execute protection bits
- A file does not need to have an ACL:
  - Includes an additional protection bit that indicates whether the file has an extended ACL

When a process requests access to a file system object two steps are performed:

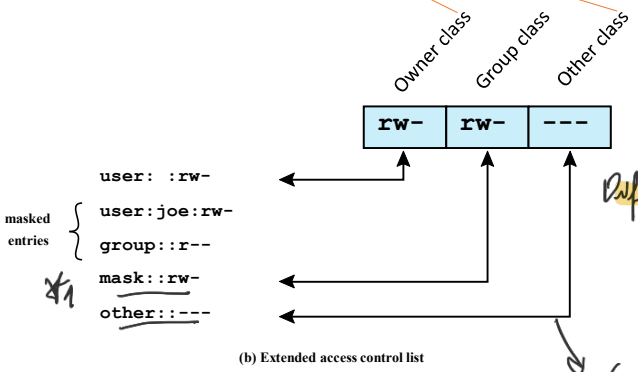
- Step 1 selects the most appropriate ACL (entry)
- Step 2 checks if the matching entry contains sufficient permissions

If you are using an ACL the system needs

41 starts with inspecting the ACL of the owner, then user then group.

Access control lists (ACLs) in Unix

Same as in traditional UNIX



Different meaning

*

Group represents a mask.

Sequence to grant access to a file:

1. Select the ACL that closely match the requesting process
   - In this order: owner, named users, groups, others
2. Check if the matching entry contains sufficient permissions
   - A process can be in more than one group, just one is sufficient…

42 * if ACL has not been enabled this repr. has the classic meaning. Otherwise, *1

*1 other users and all the groups as an ACL.

Mask is used to have a fast and efficient way to grant or revoke.

Mask can revoke privileges to everybody. Access control is and and between ACL and if you are allowed because of the mask.

- For the DAC model an alternative representation of the protection state is a directed graph.
- Each subject and each object in the protection state is represented by a node (a single node is used for an entity that is both subject and object).
- A directed line from a subject to an object indicates an access right, and the label on the link defines the access right.

DAC – exercise

43

Rather than representing a matrix in this way, you get:

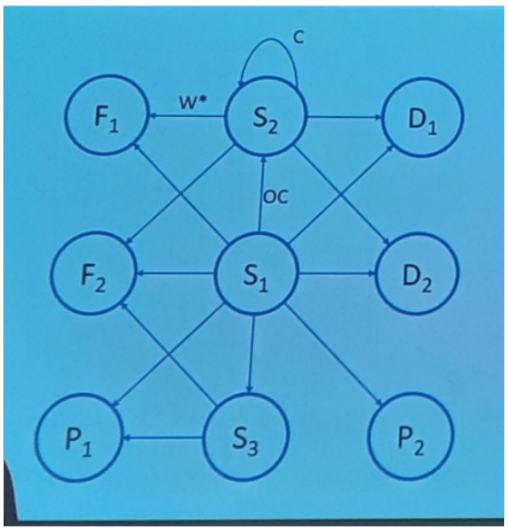
a. Draw a directed graph that corresponds to the access matrix of the figure

Objects

Subjects	File 1	File 2	File 3	File 4
User A	Own Read Write		Own Read Write	
User B	Read	Own Read Write	Write	Read
User C	Read Write	Read		Own Read Write

DAC – exercise

44



	File 1	File 2	File 3	File 4
User A	Own Read Write		Own Read Write	
User B	Read	Own Read Write	Write	Read
User C	Read Write	Read		Own Read Write

DAC – Solution (a)

45

b. Draw a directed graph that corresponds to the access matrix of the table:

Objects

Subjects	subjects			files		processes		disk drives	
	S_1	S_2	S_3	F_1	F_2	P_1	P_2	D_1	D_2
S_1	control	owner	owner control	read *	read owner	wakeup	wakeup	seek	owner
S_2		control		write *	execute			owner	seek *
S_3			control		write	stop			

c. Is there a one-to-one correspondence between the directed graph representation and the access matrix representation? Explain.

DAC – exercise

47

	S_1	S_2	S_3	F_1	F_2	P_1	P_2	D_1	D_2
S_1	control	owner	owner control	read *	read owner	wakeup	wakeup	seek	owner
S_2		control		write *	execute			owner	seek *
S_3			control		write	stop			

DAC
–
Solution (b)

48

Role based access control (RBAC)

Granularity in OS is Files. So not many requirements. DAC is good enough, but not in modern OSes. Those that push for more security they complement DAC with MAC

50 Role based represent the roles of an org.

The users do not disappear.

- From users' identity to roles of users:
  - Defines a job function in an organization
  - Rights assigned to roles (Permissions were ass. to users)
  - A user may change roles and thus change rights

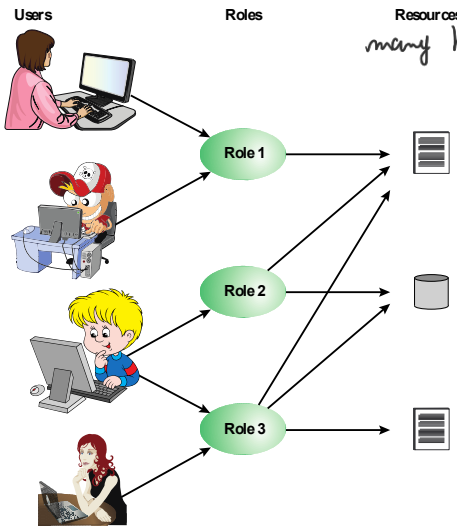
A user is associated to one or more roles and they decide which one to impersonate.

- Widespread commercial use
- NIST standard in 2009

Role based access control

51

Relationship between Users and roles is many to many.



Users, roles and resources

52 With roles it's possible to implement the least privilege approach. In an org. it makes a lot of sense.

Access control matrix representation of RBAC

Matrix for the Relation users-roles

	R_1	R_2	...	R_n
U_1	X			
U_2	X			
U_3		X		X
U_4				X
U_5				X
U_6				X
...				
U_m	X			

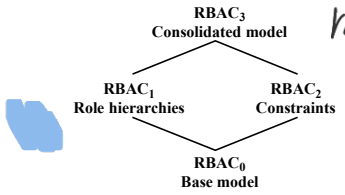
Subjects are roles rather than users, Access control matrix (roles/objects) and roles are also represented as objects because we need to create them

ROLES	OBJECTS								
	R_1	R_2	R_n	F_1	F_1	P_1	P_2	D_1	D_2
R_1	control	owner	owner control	read *	read owner	wakeup	wakeup	seek	owner
R_2		control		write *	execute			owner	seek *
...									
R_n			control		write	stop			

53

A family of role-based access control models

Formal hierarchies: roles that inherit privileges of a role and add something else. Another possibility is adding constraints: "if user gets a role it can't get another.



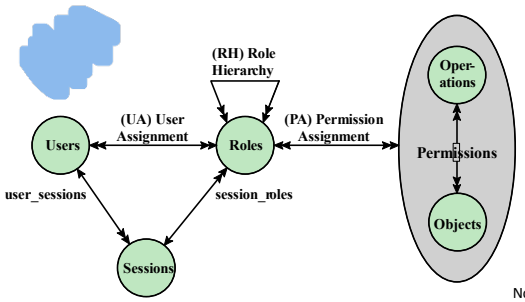
Four models related to each other

Models	Hierarchies	Constraints
$RBAC_0$	No	No
$RBAC_1$	Yes	No
$RBAC_2$	No	Yes
$RBAC_3$	Yes	Yes

Scope of RBAC models

54 You can implement RBAC by implementing constraints, hierarchies or you can combine them. You have to think what you need to implement.

Users are in relationship many to many. User create a session and takes a role.



RBAC$_0$ model

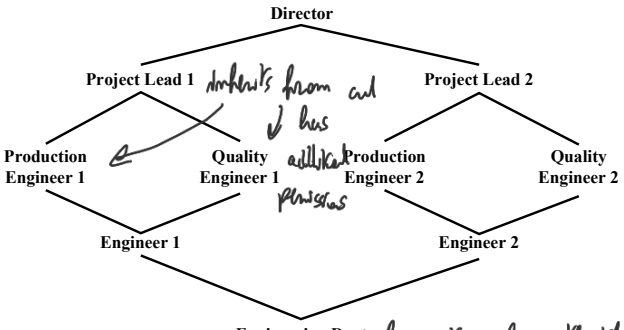
Note:
single arrow means one,
double arrow means many.

- RBAC$_0$ does not have role hierarchy and constraints
- It contains the four types of entities:
  - **User**: An individual (with user ID) that has access to this computer system.
  - **Role:** A named job function within the organization that controls this computer system (includes a description of the authority and responsibility conferred on this role).
  - **Permission:** An approval of a particular mode of access to one or more objects (i.e. *access right, privilege, and authorization*).
  - **Session**: A mapping between a user and an activated subset of the set of roles to which the user is assigned.

A family of role-based access control models

55

When you connect to a system you invoke a session where you take a role after can authentication (check also constraints). You can take multiple roles too. Simple case: you only connect to take another role. General model.



inherits from and has all the permissions

lowest role with the lowest permissions.

Role hierarchies provide a means of reflecting the hierarchical structure of roles in an organization

- greater responsibility implies greater authority to access resources
- Inheritance of access rights from the bottom to the top

You do not necessarily need a hierarchy but it is more efficient.

RBAC$_1$: Example of ROI hierarchy

56

The use of inheritance can reduce a lot of complexity. With DAC, the users organize the work with transferring privileges. It can be dangerous in an org. with several systems. Losing control over the access right of resources can be dangerous.

$RBAC_2$: constraints

- Provide a means of adapting RBAC to the specifics of administrative and security policies of an organization
- A constraint is a relationship among roles or a condition related to roles
- Types of constraints:

Mutually exclusive roles	Cardinality	Prerequisite roles
• A user can only be assigned to one role in the set (either during a session or statically) • Any permission (access right) can be granted to only one role in the set	• Setting a maximum number of users in a roles • May also limit the number of roles per user, or the number of roles a user can have per session	• Dictates that a user can only be assigned to a particular role if it is already assigned to some other specified role

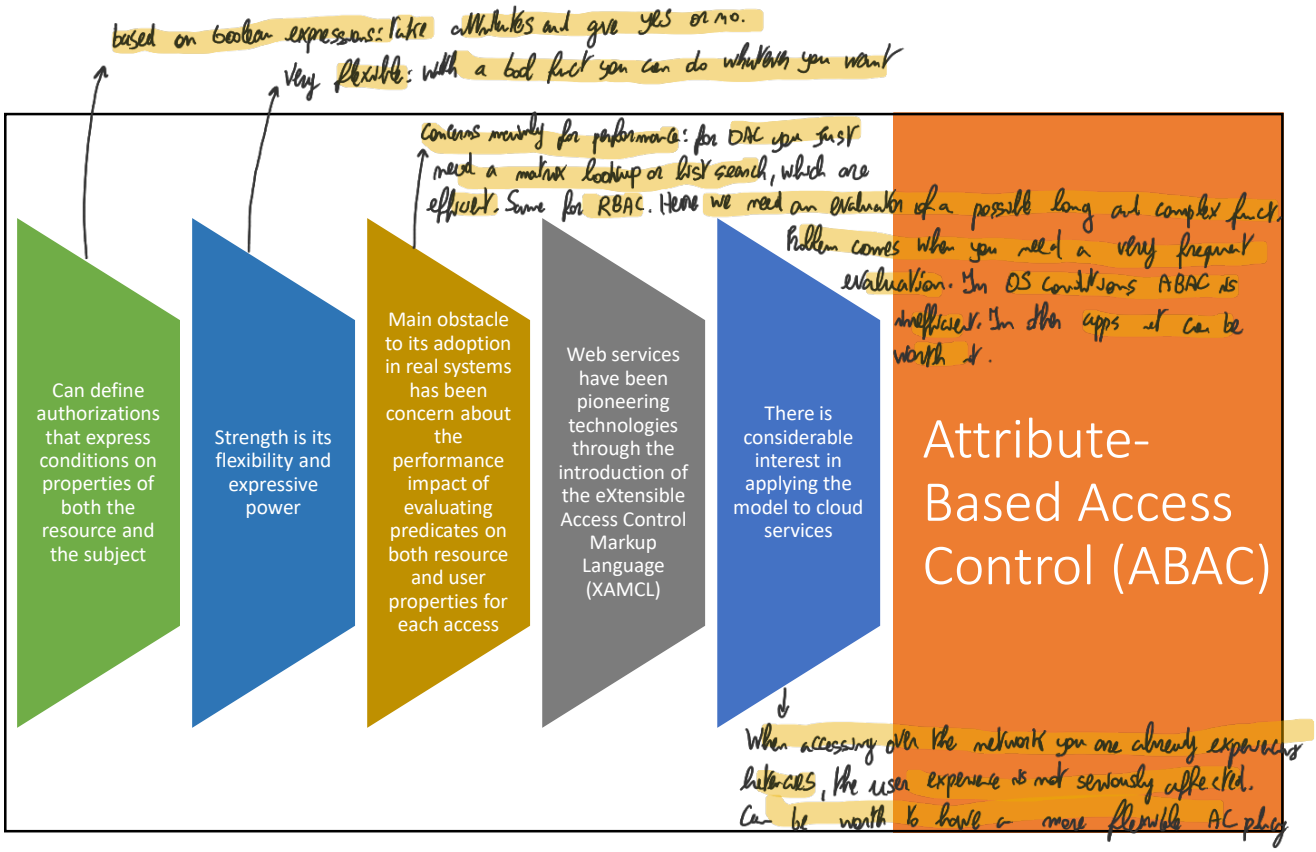
You cannot assign a specific role if the user already has a specific role: can enforce PRINCIPLE of SEP. OF PRIVILEGES

57

Attribute-based access control (ABAC)

We take an ac decision based on attributes that can refer to the subj., obj. and environment.

58



59

- Three key elements of an ABAC system:
  - attributes – defined for entities in a configuration;
  - a policy model – defines the ABAC policies;
  - the architecture model – applies to policies that enforce access control

Attribute-Based Access Control (ABAC)

60

Attribute-Based Access Control (ABAC) model: attributes

Subject attributes	Object attributes	Environment attributes
• A subject is an active entity that causes information to flow among objects or changes the system state	• An object (or resource) is a passive information system-related entity containing or receiving information	• Describe the operational, technical, and even situational environment or context in which the information access occurs
• Attributes define the identity and characteristics of the subject (id, name, job title, etc.)	• Objects have attributes that can be leverages to make access control decisions	• These attributes have so far been largely ignored in most access control policies
	• Attribute depend on the object… (owner, author, date, QoS,…)	• For example: current time or virus/hacker activity, network security level,…

Any active entity, can act on an object

Whatever you may associate to a subj.

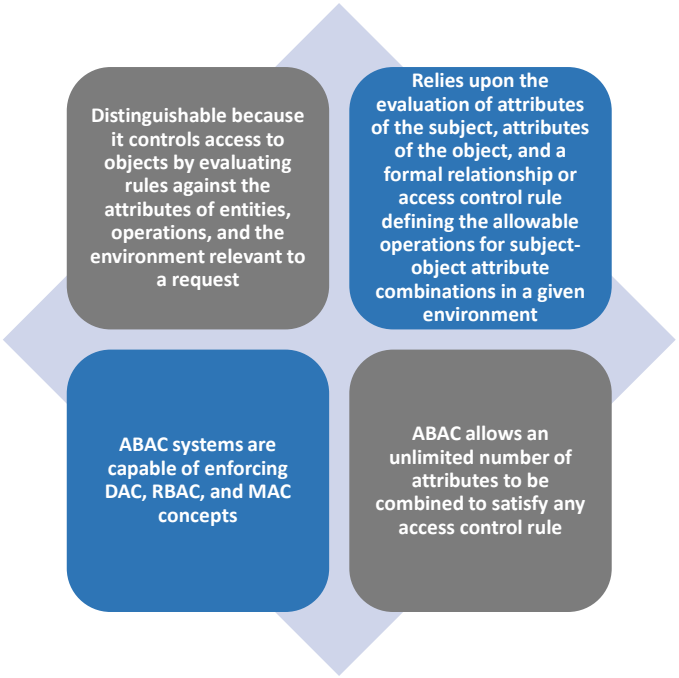
RBAC is implementable by assigning role as attribute

ex: owner, group in unix.

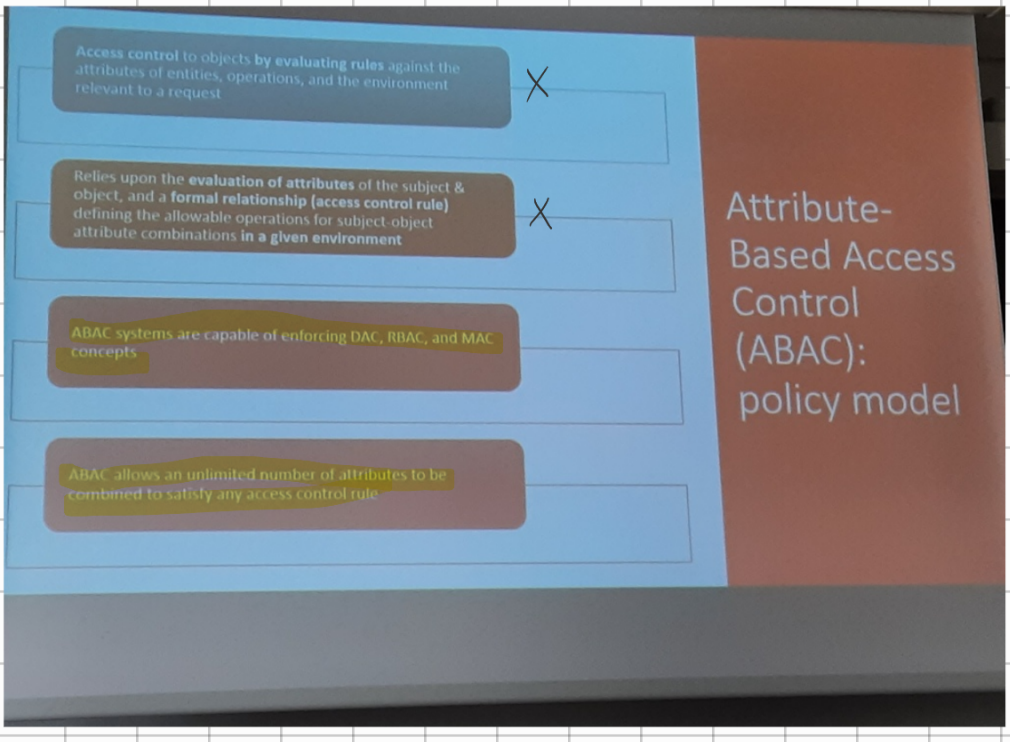
61

* Represents everything else. May also not be present but useful. Assume AC for elevator, which should not be used in case of fire. So to model the fact that in case of fire users are forbidden to use the elevator.

Attribute-Based Access Control (ABAC)

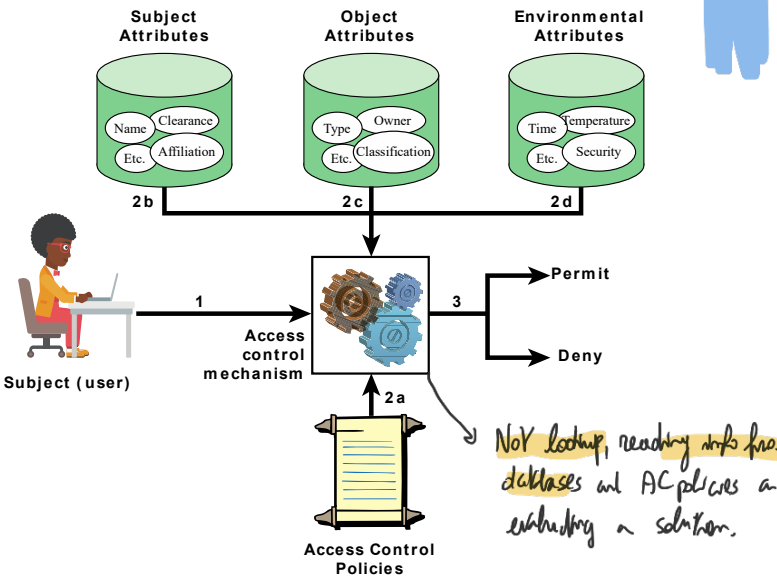


62



When you define the policy you write the bool function. You can write as many rules as you wish. Their eval gives the AC answer.

Attribute-based access control scenario

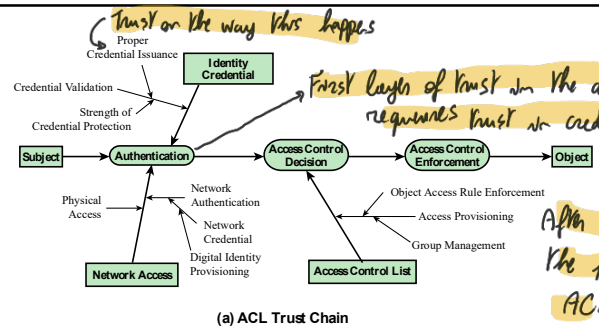


NoV lookup, reading info from databases and AC policies and evaluating a solution.

63

Behind every AC Policy there's a trust model. There's an inherent trust in every ACP. The entity in which you put trust varies over ACPs. Take DAC with ACL. The Chain is the one below.

ACL and ABAC relationships



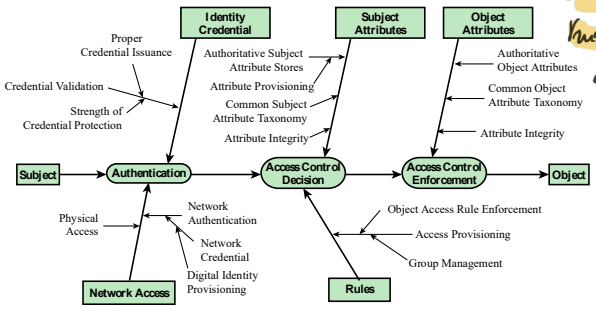
(a) ACL Trust Chain

Trust on the way this happens

First layer of trust in the authentication, which requires trust in credential and network (way in which they are exchanged.

After auth we have AC decision: 2 elements of relevance: the fact that the user has been auth and the ACL, so trusting the way it is managed.

In DAC, the root of the trust itself is the user and the subject, who decides who gets specific rights.

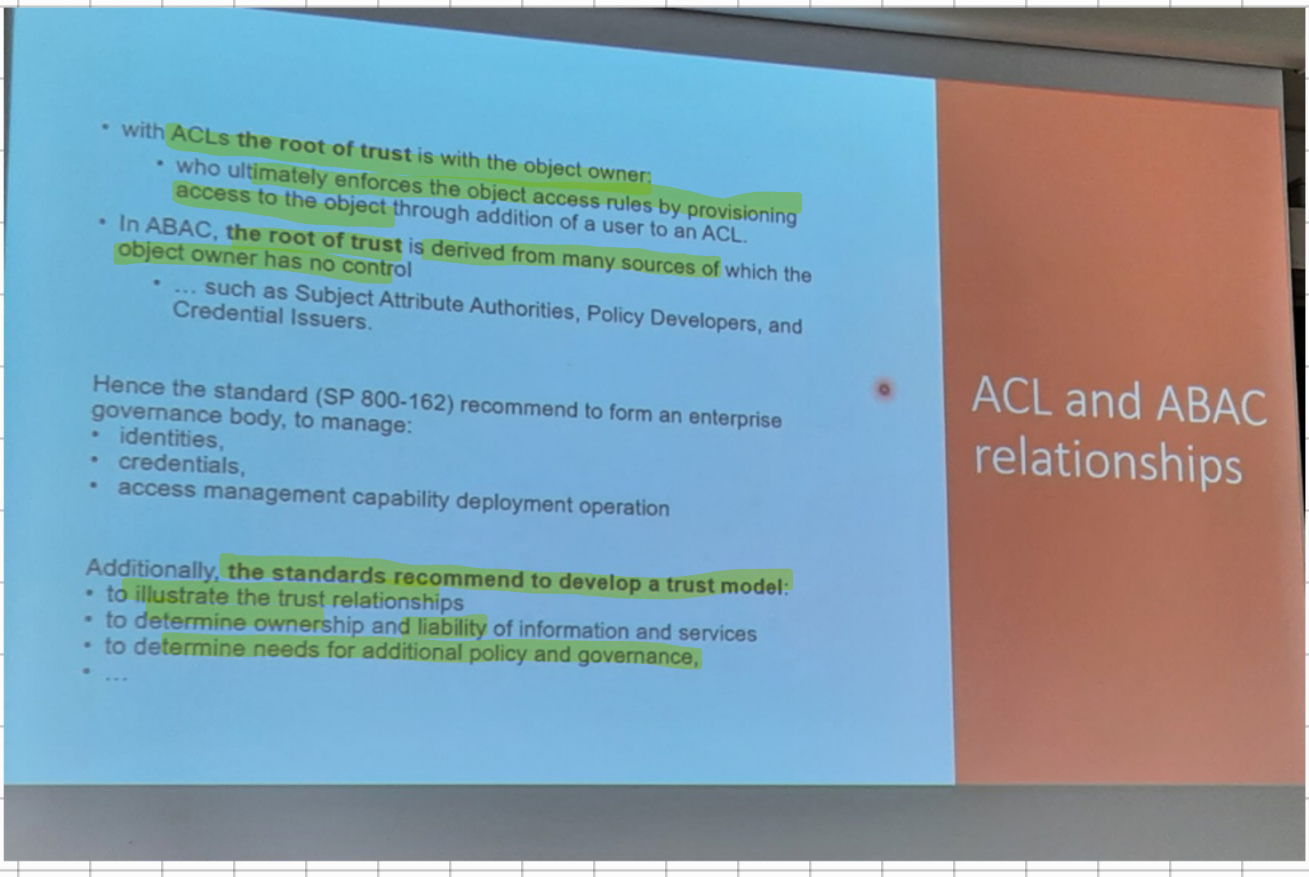


(b) ABAC Trust Chain

64

For ABAC the first part is the same, but the decision on AC depends on a set of databases regarding Subject attributes, Object attributes and rules. We rely on who define the attributes. & environments

We rely on longer # of entities and the subject doesn't have the same much power has before.


with ACLs the root of trust is with the object owner:
who ultimately enforces the object access rules by provisioning access to the object through addition of a user to an ACL.
In ABAC, the root of trust is derived from many sources of which the object owner has no control
… such as Subject Attribute Authorities, Policy Developers, and Credential Issuers.
Hence the standard (SP 800-162) recommend to form an enterprise governance body, to manage:
identities,
credentials,
access management capability deployment operation
Additionally, the standards recommend to develop a trust model:
to illustrate the trust relationships
to determine ownership and liability of information and services
to determine needs for additional policy and governance,
…
ACL and ABAC relationships

A trust model is a tool that is suggested to be used down to reason the relationships of trust in org.

A policy is a set of rules and relationships that govern allowable behavior within an organization, based on the privileges of subjects and how resources or objects are to be protected under which environment conditions

Typically written from the perspective of the object that needs protection and the privileges available to subjects

Privileges represent the authorized behavior of a subject and are defined by an authority and embodied in a policy

Other terms commonly used instead of privileges are: rights, authorizations, and entitlements

ABAC policies

65

1. S, O, and E are (sets of) subjects, objects, and environments, respectively;
2. SA_k ($1 \le k \le K$): predefined attributes for subjects; (We define the attributes and define the values)
   OA_m ($1 \le m \le M$): predefined attributes for objects;
   EA_n ($1 \le n \le N$): predefined attributes for environments;

ABAC policies notation

66

3. ATTR(*s*), ATTR(*o*), and ATTR(*e*) are attribute assignment relations for subject *s*, object *o*, and environment *e*, respectively:
   - ATTR(*s*) $\subseteq SA_1 \times SA_2 \times \ldots \times SA_K$
   - ATTR(*r*) $\subseteq OA_1 \times OA_2 \times \ldots \times OA_M$
   - ATTR(*o*) $\subseteq EA_1 \times EA_2 \times \ldots \times EA_N$

   Examples of attributes:
   - Role(*s*) = "Service Consumer"
   - ServiceOwner(*o*) = "XYZ, Inc."
   - CurrentDate(*e*) = "01-23-2005"

ABAC policies notation

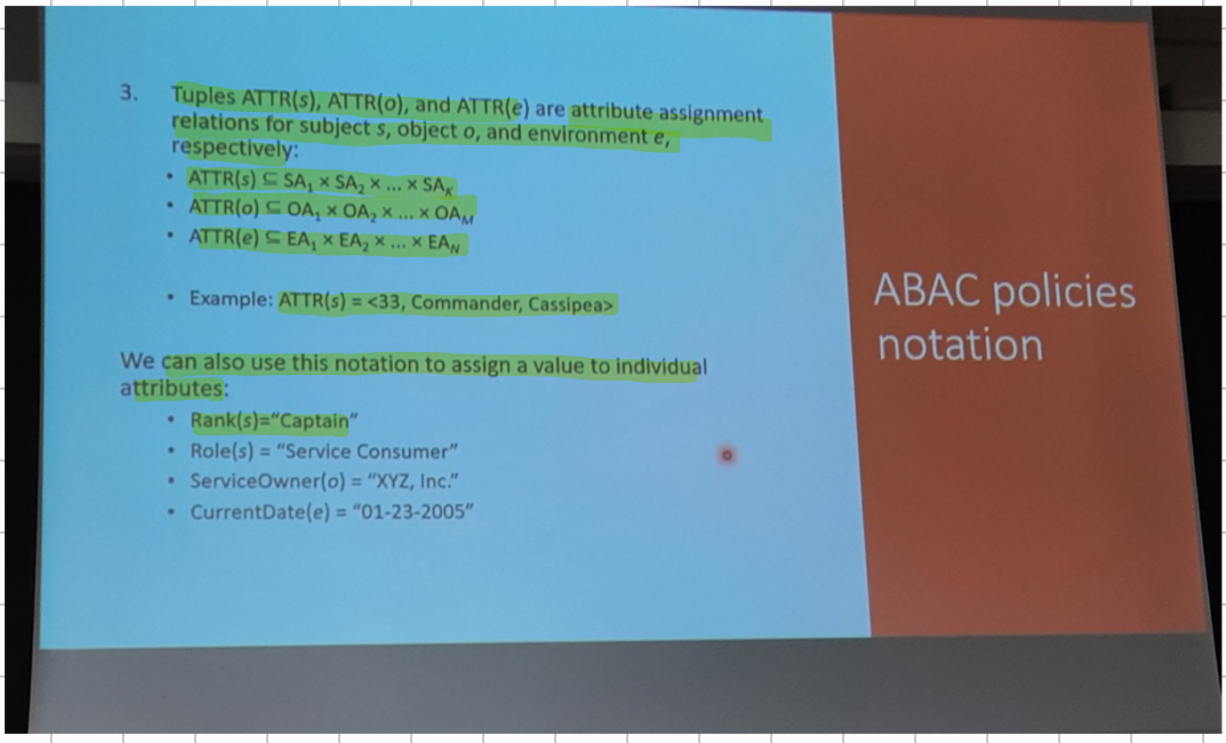
67

4) a Policy Rule:
   - decides on whether a subject *s* can access an object *o* in a particular environment *e*;
   - it is a Boolean function of the attributes of *s*, *o*, and *e*:
   - Rule: can_access(*s*, *o*, *e*) ← *f*(ATTR(*s*), ATTR(*o*), ATTR(*e*))

5) A Policy Rule Base (or policy store):
   - may consist of a number of policy rules
   - Each rule covers many subjects and objects within a security domain.
   - The access control decision process in essence amounts to the evaluation of applicable policy rules in the policy store.

ABAC policies notation

68


3. Tuples ATTR(s), ATTR(o), and ATTR(e) are attribute assignment relations for subject s, object o, and environment e, respectively:
• ATTR(s) ⊆ SA1 × SA2 × ... × SAK
• ATTR(o) ⊆ OA1 × OA2 × ... × OAM
• ATTR(e) ⊆ EA1 × EA2 × ... × EAN
• Example: ATTR(s) = <33, Commander, Cassipea>
We can also use this notation to assign a value to individual attributes:
• Rank(s)="Captain"
• Role(s) = "Service Consumer"
• ServiceOwner(o) = "XYZ, Inc."
• CurrentDate(e) = "01-23-2005"
ABAC policies notation

- Now consider the example of an online entertainment store that streams movies to users for a flat monthly fee.
- The store must enforce the following access control policy based on the user's age and the movie's content rating:

Movie Rating	Users Allowed Access
R	Age 17 and older
PG-13	Age 13 and older
G	Everyone

Model the access control policies with RBAC and with ABAC

RBAC and ABAC – exercise

69

Movie Rating	Users Allowed Access
R	Age 17 and older
PG-13	Age 13 and older
G	Everyone

RBAC To do:

1. Define the roles (?)
2. Define the permissions (?)
3. Define which permissions are given to each role (?)

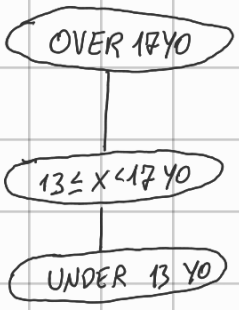
ABAC to do:

1. Define the roles (?)
2. Define attributes (?)
3. Define policy rules (?)

Solution – Hint

70

4



PERMS: 1. Access G Rating Movies. 2. Access PG-13 movies. 3. Access R Rated movies.

UNDER 13 YO only has permission 1. 13 ≤ X < 17 YO also has permission 2. OVER 17 YO also has permission 3.

ABAC:

Attributes of subjects: $SA_1 = \{$ Under 13, Over 13 and under 17, Over 17 $\}$

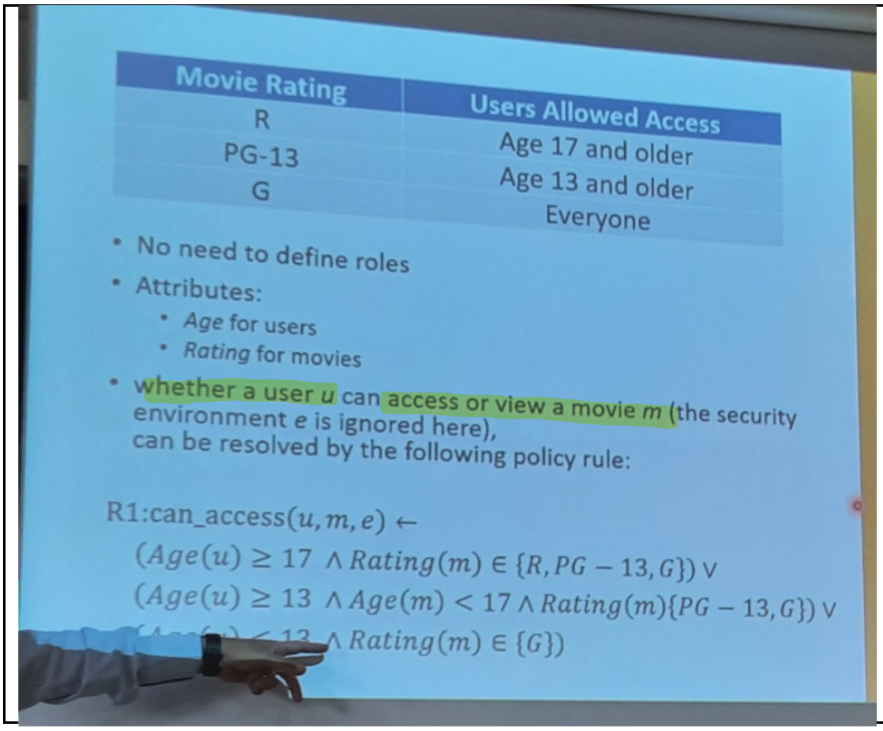
Attributes of objects: $OA_1 = \{$ G rated, PG-13 rated, R rated $\}$

$$can\_access(s, o) = Over17 + (Over13\ and\ under\ 17) \cdot (\overline{R\ rated}) + (Under\ 13) \cdot (G\ rated)$$

Movie Rating	Users Allowed Access
R	Age 17 and older
PG-13	Age 13 and older
G	Everyone

Solution – RBAC

71



Solution – ABAC

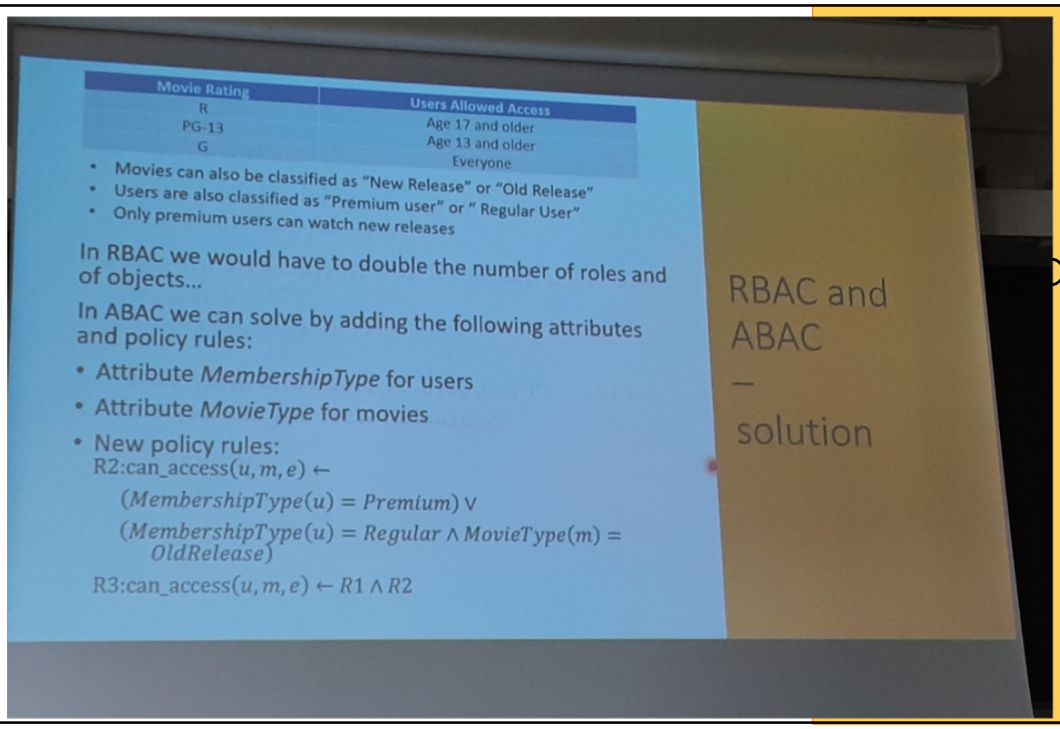
73

Assume now that:

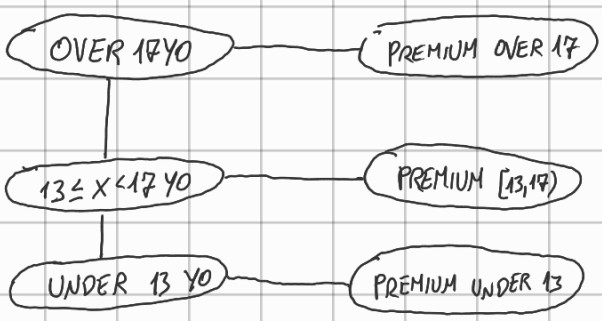
- movies can also be classified as “New Release” or “Old Release”
- Users are also classified as “Premium user” or “ Regular User”
- Model RBAC and ABAC so that only Premium Users can see the New Releases.
- Model ABAC

RBAC and ABAC – exercise

75



76



PERMS: 1. Access to G rated movies; 2. Access to new release G rated movies;
3. Access to PG-13 rated movies; 4. Access to new release PG-13 rated movies;
5. Access to R rated movies; 6. Access to new release R rated movies;

Under 13 yo only has 1. [13,17) YO has 1 and 3. Over 17 YO has 1, 3, 5.
Premium Under 13 yo has 1, 2. Premium [13,17) YO has 1, 2, 3, 4. Premium Over 17 YO has all.

ABAC: Sub. Attribute 1 = age; Sub. Attribute 2 = Type; Obj. attribute 1 = Rating; Obj. Attribute 2 = relTime

Attributes of subjects: $SA_1 = \{$ Under 13, Over 13 and under 17, Over 17 $\}$

$SA_2 = \{$ Regular, Premium $\}$

Attributes of objects: $OA_1 = \{$ G rated, PG-13 rated, R rated $\}$

$OA_2 = \{$ Old Release, New Release $\}$

Con_access(s, o) =

$$= \Big\{ (age(s) = \text{Over 17}) \text{ OR } \big[(age(s) = \text{Over 13 and under 17}) \text{ AND } (Rating(o) \neq \text{R rated}) \big] \text{ OR } \big[(age(s) = \text{Under 13}) \text{ AND } (Rating = \text{G RATED}) \big] \Big\}$$

$$\text{AND } \Big\{ (Type(s) = \text{Premium}) \text{ OR } (relTime(o) = \text{Old Release}) \Big\}$$

MATRIX:

	Old G	New G	Old PG-13	New PG-13	Old R	New R
Under 13 YO	Access					
[13, 17) YO	Access		Access			
Over 17 YO	Access		Access		Access	
Premium Under 13 YO	Access	Access				
Premium [13, 17) YO	Access	Access	Access	Access		
Premium 17 YO	Access	Access	Access	Access	Access	Access

Excluding extremes

Assume in RBAC, if there are *K* subject attributes and *M* object attributes: Equivalent role based models

- the number of required roles are:

$$\prod_{k=1}^{K} range(SA_k)$$

- the number of required permissions are:

$$\prod_{m=1}^{M} range(SA_m)$$

where $range(\cdot)$ denotes the number of possible values of each attribute

In contrast ABAC is more efficient, it is just sufficient to add an attribute.

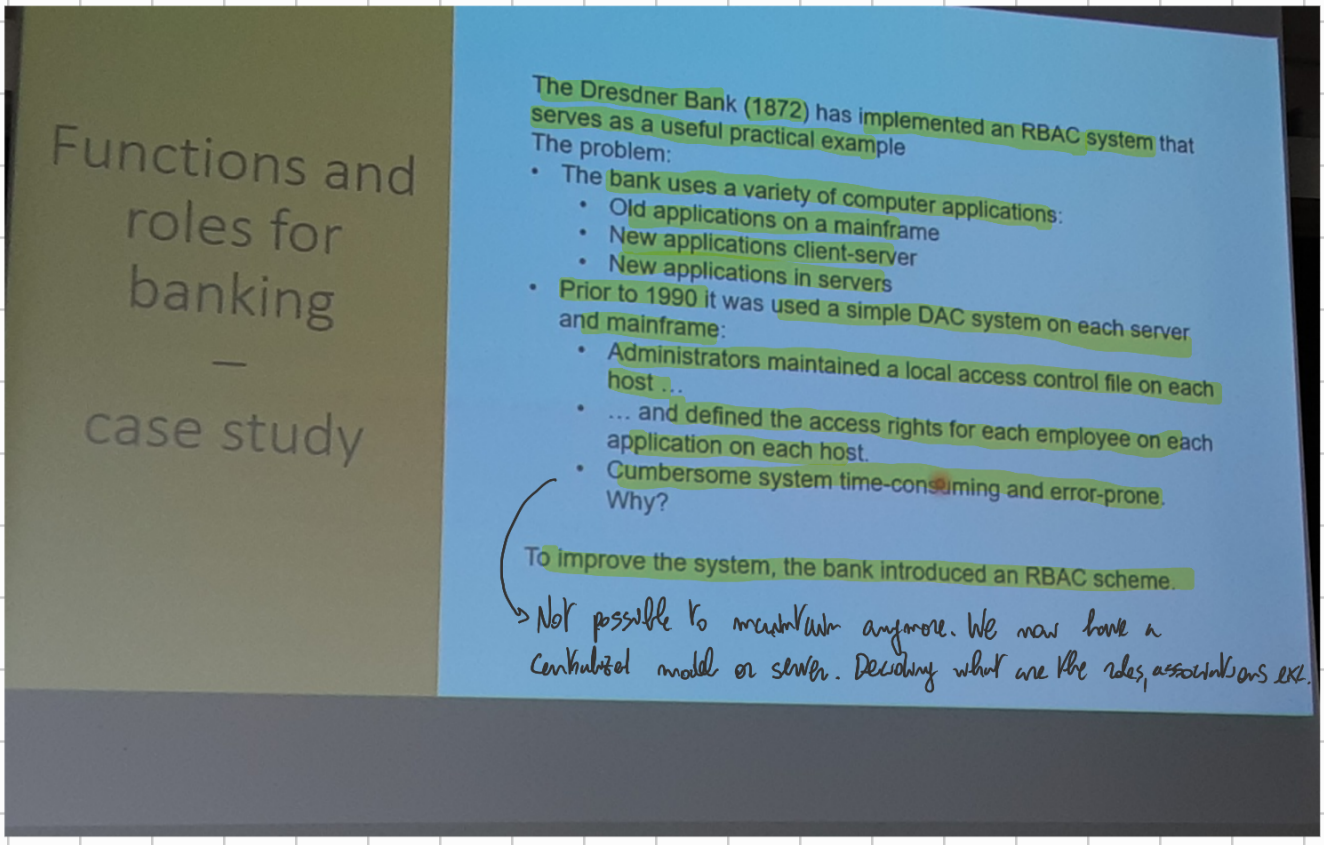
You can express complex situations with simple boolean functions.

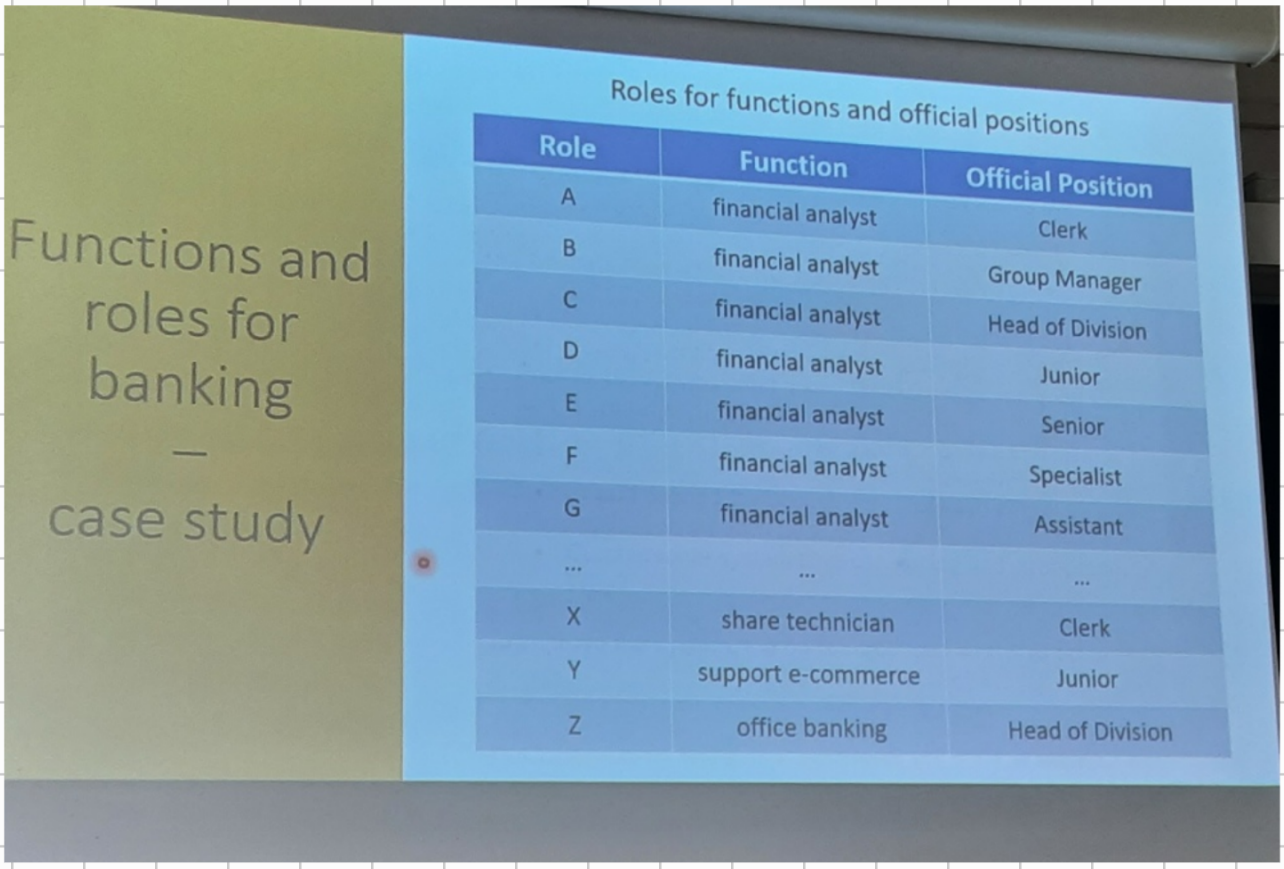
RBAC vs ABAC

78

Identity, credential, and access management (ICAM) and trust frameworks

79





Associated to the function we have the official position. So Job functions and positions were used to assign roles.

Financial analyst – clerk

Financial analyst – group manager

Functions and roles for banking – case study

Permission assignment

Role	Application	Access Right
A	money market instruments	1, 2, 3, 4
	Derivatives trading	1, 2, 3, 7, 10, 12
	Interest instruments	1, 4, 8, 12, 14, 16
B	money market instruments	1, 2, 3, 4, 7
	derivatives trading	1, 2, 3, 7, 10, 12, 14
	interest instruments	1, 4, 8, 12, 14, 16
	private consumer instruments	1, 2, 4, 7
...	...	...

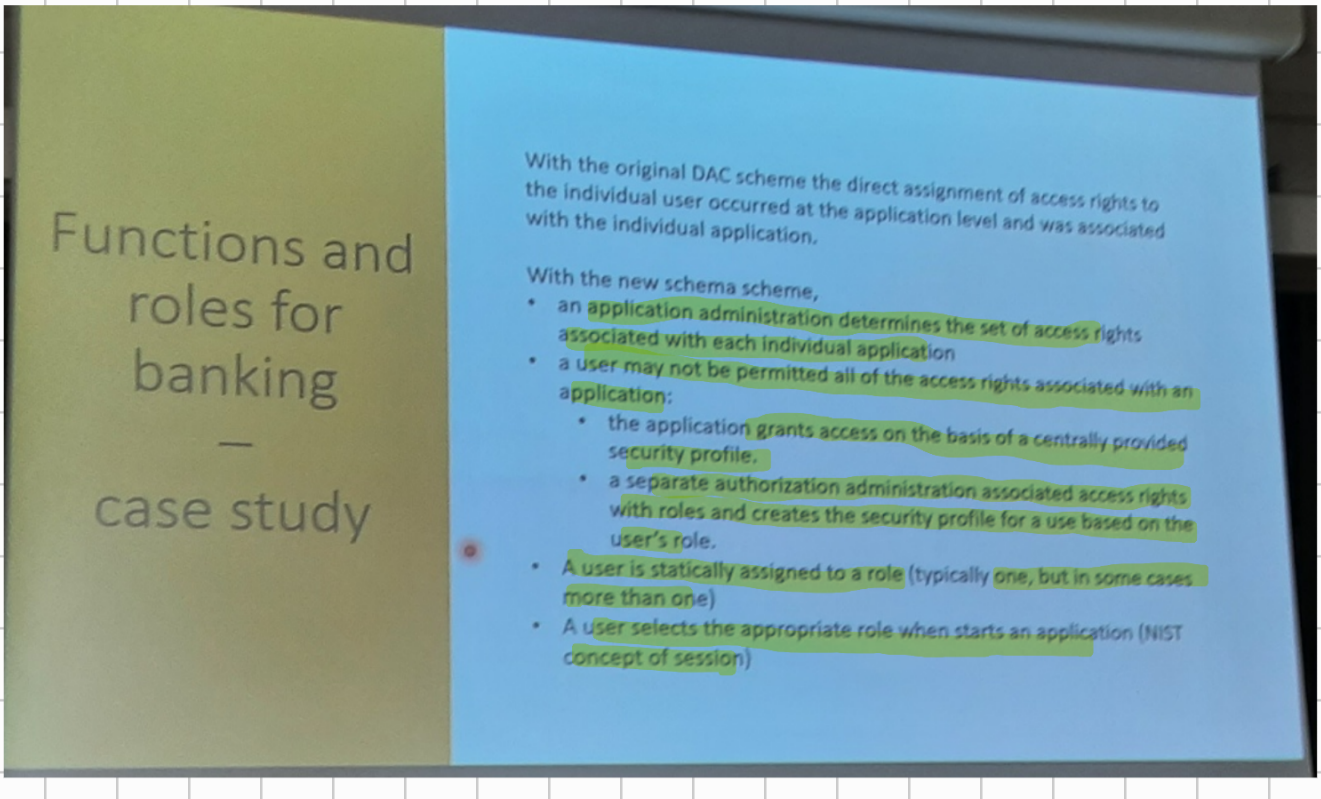
Perm. assignment with inheritance

Role	Application	Access Right
A	money market instruments	1, 2, 3, 4
	derivatives trading	1, 2, 3, 7, 10, 12
	interest instruments	1, 4, 8, 12, 14, 16
B	money market instruments	7
	derivatives trading	14
	private consumer instruments	1, 2, 4, 7
...	...	...

1. Partial ordering of roles w.r.t. permissions. For example, permissions of A ≤ permissions of B
   - However, there's no hierarchical relationship between Office banking/Group Manager and financial analyst/Clerk because they work in different functional areas
2. Inheritance of permissions: B inherits permissions of A

A bit redundant without hierarchy.

There is a partial order. Others are not in a strict relationship among them.

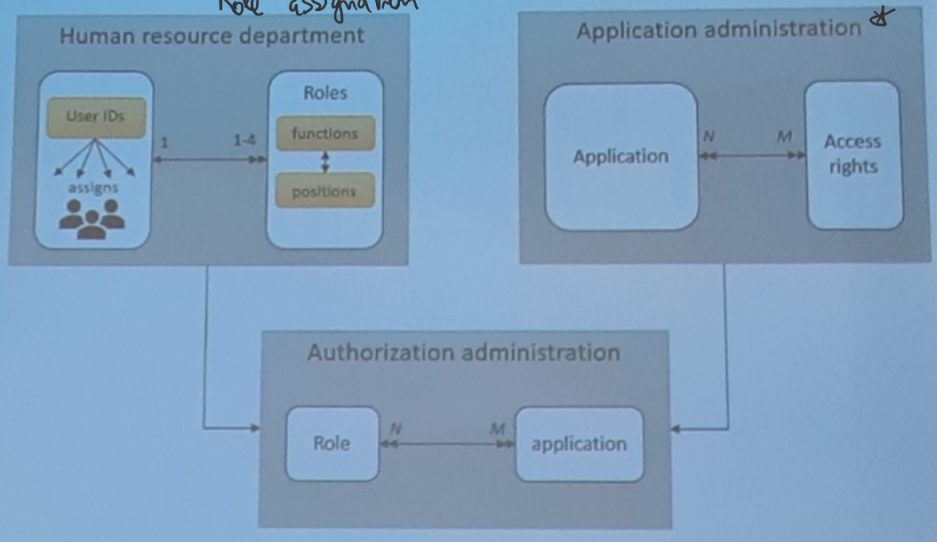


First part is defining the access matrix. Then assign the roles, which can be dynamically done.

Functions and roles for banking – case study

Examples of access control administration

Role assignation



* Composed by technical people that develop application, identifying the functionalities within applications and define the access right. Those 2 entity provide info for the Auth admin. that provides relation. Now me with a role should be allowed to each. So association between roles and privileges.

Functions and roles for banking – case study

Some figures about this system are of interest:

- Within the bank, there are 65 official positions
  - ranging from a Clerk in a branch, through the Branch Manager, to a Member of the Board.
- These positions are combined with 368 different job functions provided by the human resources database.
- Potentially, there are 23,920 different roles
- The number of roles in current use is about 1,300.

Not all roles are useful for the bank.

Nowadays any org. selling something has a digital marketer. You will have users registered on the e-commerce platform. You are selling something but you are tasked with a critical management of critical info (wallets, credentials ...). This happens for everything that is in digital world.

Identity, credential, and access management (ICAM)

- ICAM is a comprehensive approach to managing and implementing digital identities, credentials, and access control
- Developed by the U.S. government
- Designed to:
  - Create trusted digital identity representations of individuals and nonperson entities (NPEs)
  - Bind those identities to credentials that may serve as a proxy for the individual of NPE in access transactions
    - A credential is an object or data structure that authoritatively binds an identity to a token possessed and controlled by a subscriber
  - Use the credentials to provide authorized access to an agency's resources

80

This wants to provide a general sol. to this, provide a way to provide identity digitally.
DIGITAL ID: Something uniquely associated to individual
Trusted because there's an authority creating them.

Identity management

- CORE
  Identity management is concerned with assigning attributes to a digital identity and connecting that digital identity to an individual or NPE
- Goal is to establish a trustworthy digital identity that is independent of a specific application or context
  ↳ bound to a government
- Most common approach to access control for applications and programs is to create a digital representation of an identity for the specific use of the application or program
  - … but maintenance and protection of the identity itself is treated as secondary to the mission associated with the application!

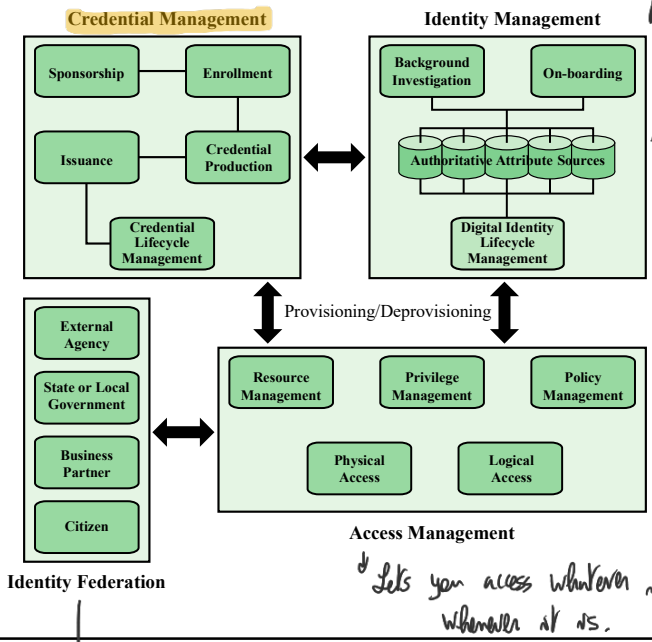
81

Identity management

- Unlike accounts, identity is not associated to job title, job duties, location or any other item
  - Those items may just be attributes tied to an identity
  - In this way access control is based on the context and relevant attributes of a user—not solely their identity
- Individuals will have a single digital representation of themselves
  - it can be leveraged across departments and agencies for multiple purposes, including access control.

82

Key functions in identity management



83

Identity management

- Final element is lifecycle management which includes:
  - Mechanisms, policies, and procedures for protecting personal identity information
  - Controlling access to identity data
  - Techniques for sharing authoritative identity data with applications that need it
  - Revocation of an enterprise identity

84

Credential management

- A credential is an object or data structure that authoritatively binds an identity to a token possessed and controlled by a subscriber.
  - Examples of credentials are smart cards, private/public cryptographic keys, and digital certificates
- The management of the life cycle of the credential encompasses five logical components:
  1. An authorized individual sponsors an individual or entity for a credential to establish the need for the credential
  2. The sponsored individual enrolls for the credential
     - Process typically consists of identity proofing and the capture of biographic and biometric data
     - This step may also involve incorporating authoritative attribute data, maintained by the identity management component
  3. A credential is produced
     - Depending on the credential type, production may involve encryption, the use of a digital signature, the production of a smart card or other functions
  4. The credential is issued to the individual or NPE
  5. A credential must be maintained over its life cycle
     - Might include revocation, reissuance/replacement, reenrollment, expiration, personal identification number (PIN) reset, suspension, or reinstatement

85

Access management

Deals with the management and control of the ways entities are granted access to resources

Covers both logical and physical access

May be internal to a system or an external element

Purpose is to ensure that the proper identity verification is made when an individual attempts to access a security sensitive building, computer systems, or data

Three support elements are needed for an enterprise-wide access control facility:

- Resource management
- Privilege management
- Policy management

86

Three support elements are needed for an enterprise-wide access control facility:

Resource management

- Concerned with defining rules for a resource that requires access control
- Rules would include credential requirements and what user attributes, resource attributes, and environmental conditions are required for access of a given resource for a given function

Privilege management

- Concerned with establishing and maintaining the entitlement or privilege attributes that comprise an individual's access profile
- These attributes represent features of an individual that can be used as the basis for determining access decisions to both physical and logical resources
- Privileges are considered attributes that can be linked to a digital identity

Policy management

- Governs what is allowable and unallowable in an access transaction

87

Identity federation

- Term used to describe the technology, standards, policies, and processes that allow an organization to trust digital identities, identity attributes, and credentials created and issued by another organization
- Addresses two questions:
  - How do you trust identities of individuals from external organizations who need access to your systems
  - How do you vouch for identities of individuals in your organization when they need to collaborate with external organizations

88

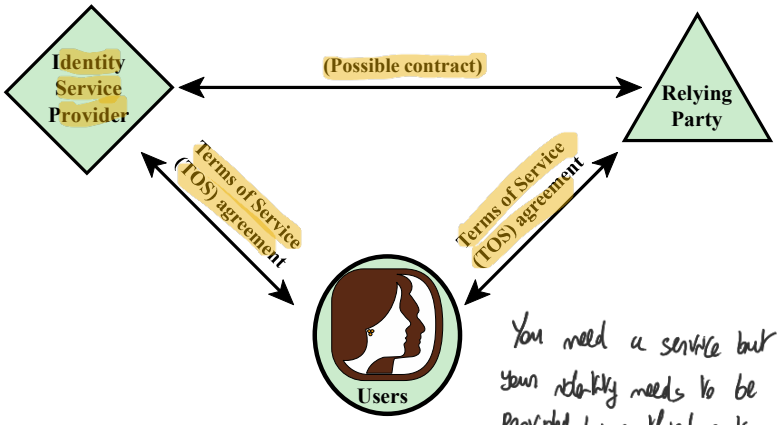
Trust

You have an authority that gives you digital id that you use to access services.

- Trust, identity and attributes have become a primary concern in internet business, network service providers etc.
- Consider the case of e-commerce:
  - What do you need to know (which attributes) about someone in order to deal with them?
  - Depends case by case:
    - professional registration or license number, organization and department, staff ID, security clearance, customer reference number, credit card number, unique health identifier, allergies, blood type, Social Security number, address, citizenship status, social networking handle, pseudonym, …
    - The attributes of an individual that must be known and verified to permit a transaction depend on context
- This all is going beyond e-commerce

89

Identity Information exchange approaches



You need a service but your identity needs to be provided by a third party.

Traditional triangle of parties involved in an exchange of identity information

You have to trust the exchange of your attributes.

90

Trust aspects of the parties.

Identity Information exchange approaches

The previous schema must meet several requirements:

- The relying party:
  - requires that the user has been authenticated to some degree of assurance,
  - that the attributes imputed to the user by the identity service provider are accurate,
  - that the identity service provider is authoritative for those attributes
- The identity service provider:
  - requires assurance that it has accurate information about the user
  - if it shares information, the relying party will use it in accordance with contractual terms and conditions and the law
- The user:
  - requires assurance that the identity service provider and relying party can be entrusted with sensitive information
  - and that they will abide by the user's preferences and respect the user's privacy.
- all the parties:
  - want to know if the practices described by the other parties are actually those implemented by the parties, and how reliable those parties are.

91

Open Identity Trust Framework

A standard to the exchange of identity information. Some definitions:

- OpenID
  - An open standard that allows users to be authenticated by certain cooperating sites using a third party service
- OIDF
  - OpenID Foundation is an international nonprofit organization of individuals and companies committed to enabling, promoting, and protecting OpenID technologies
- ICF
  - Information Card Foundation is a nonprofit community of companies and individuals working together to evolve the Information Card ecosystem
- OITF
  - Open Identity Trust Framework is a standardized, open specification of a trust framework for identity and attribute exchange, developed jointly by OIDF and ICF
- OIX
  - Open Identity Exchange Corporation is an independent, neutral, international provider of certification trust frameworks conforming to the OITF model
- AXN
  - Attribute Exchange Network is an online Internet-scale gateway for identity service providers and relying parties to efficiently access user asserted, permissioned, and verified online identity attributes in high volumes at affordable costs

92

Open Identity Trust Framework

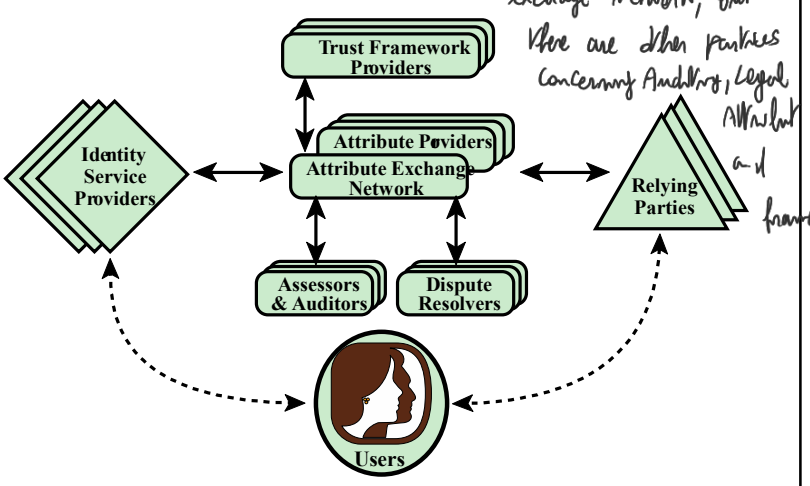
A trust framework functions as a certification program:

- It enables a party who accepts a digital identity credential (called the relying party) to trust the identity, security, and privacy policies of the party who issues the credential (called the identity service provider) and vice versa.
- A trust framework is developed by a community whose members have similar goals and perspectives.
- It defines the rights and responsibilities of that community's participants; specifies the policies and standards specific to the community; and defines the community-specific processes and procedures that provide assurance.

- Different trust frameworks can exist and sets of participants can tailor trust frameworks to meet their particular needs.

93

Identity information exchange approaches



Identity attribute exchange elements

94

Functions and roles for banking – case study

The Dresdner Bank (1872) has implemented an RBAC system that serves as a useful practical example
The problem:

- The bank uses a variety of computer applications:
  - Old applications on a mainframe
  - New applications client-server
  - New applications in servers
- Prior to 1990 it was used a simple DAC system on each server and mainframe:
  - Administrators maintained a local access control file on each host
  - and defined the access rights for each employee on each application on each host.
  - Cumbersome system time-consuming and error-prone. Why?

To improve the system, the bank introduced an RBAC scheme.

95

Functions and roles for banking – case study

(a) Functions and Official Positions

Role	Function	Official Position
A	financial analyst	Clerk
B	financial analyst	Group Manager
C	financial analyst	Head of Division
D	financial analyst	Junior
E	financial analyst	Senior
F	financial analyst	Specialist
G	financial analyst	Assistant
•••	•••	•••
X	share technician	Clerk
Y	support e-commerce	Junior
Z	office banking	Head of Division

96

Functions and roles for banking – case study

Financial analyst - clerk

Financial analyst – group manager

(b) Permission Assignments

Role	Application	Access Right
A	money market instruments	1, 2, 3, 4
	derivatives trading	1, 2, 3, 7, 10, 12
	interest instruments	1, 4, 8, 12, 14, 16
B	money market instruments	1, 2, 3, 4, 7
	derivatives trading	1, 2, 3, 7, 10, 12, 14
	interest instruments	1, 4, 8, 12, 14, 16
	private consumer instruments	1, 2, 4, 7
•••	•••	•••

(c) PA with Inheritance

Role	Application	Access Right
A	money market instruments	1, 2, 3, 4
	derivatives trading	1, 2, 3, 7, 10, 12
	interest instruments	1, 4, 8, 12, 14, 16
B	money market instruments	7
	derivatives trading	14
	private consumer instruments	1, 2, 4, 7
•••	•••	•••

1. Partial ordering of roles w.r.t. permissions. For example, permissions of A ≤ permissions of B
   - However, there's no hierarchical relationship between Office banking/Group Manager and financial analyst/Clerk because they work in different functional areas
2. Inheritance of permissions: B inherits permissions of A

97

Functions and roles for banking – case study

With the original DAC scheme the direct assignment of access rights to the individual user occurred at the application level and was associated with the individual application.

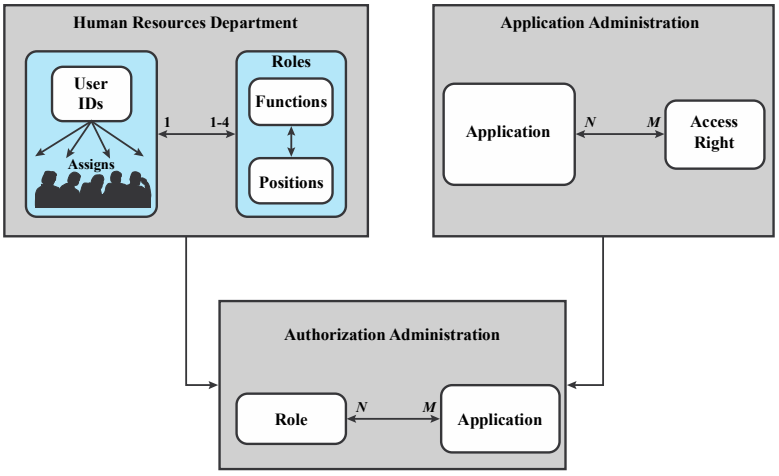
With the new schema scheme,

- an application administration determines the set of access rights associated with each individual application
- a user may not be permitted all of the access rights associated with an application:
  - the application grants access on the basis of a centrally provided security profile.
  - a separate authorization administration associated access rights with roles, and creates the security profile for a use on the basis of the user's role.
- A user is statically assigned to a role (typically one, but in some cases more than one)
- A user selects the appropriate role when starts an application (NIST concept of session)

98

Functions and roles for banking – case study

Examples of access control administration



99

Functions and roles for banking – case study

Some figures about this system are of interest:

- Within the bank, there are 65 official positions
  - ranging from a Clerk in a branch, through the Branch Manager, to a Member of the Board.
- These positions are combined with 368 different job functions provided by the human resources database.
- Potentially, there are 23,920 different roles
- The number of roles in current use is about 1,300.

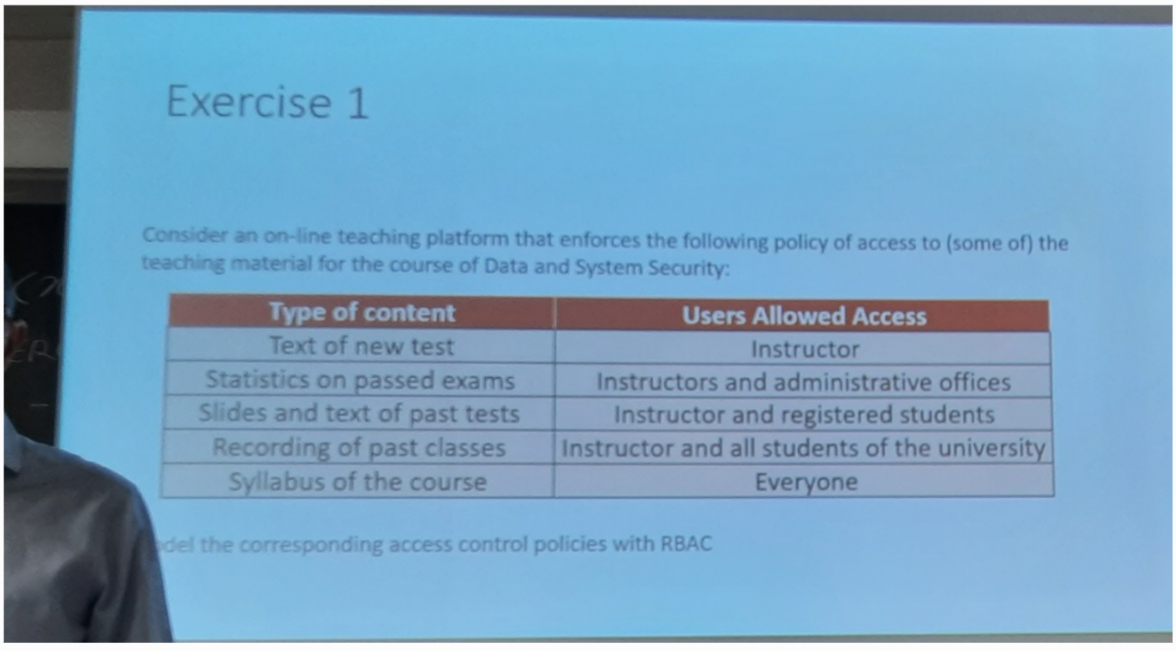
Not all roles are useful for the bank.

100

Summary

- Access control principles
  - Access control context
  - Access control policies
- Subjects, objects, and access rights
- Discretionary access control
  - Access control model
  - Protection domains
- UNIX file access control
  - Traditional UNIX file access control
  - Access control lists in UNIX
- Role-based access control
  - RBAC reference models
- Attribute-based access control
  - Attributes
  - ABAC logical architecture
  - ABAC policies
- Identity, credential, and access management
  - Identity management
  - Credential management
  - Access management
  - Identity federation
- Trust frameworks
  - Traditional identity exchange approach
  - Open identity trust framework
- Bank RBAC system

101


Exercise 1
Consider an on-line teaching platform that enforces the following policy of access to (some of) the teaching material for the course of Data and System Security:
Type of content
Users Allowed Access
Text of new test
Instructor
Statistics on passed exams
Instructors and administrative offices
Slides and text of past tests
Instructor and registered students
Recording of past classes
Instructor and all students of the university
Syllabus of the course
Everyone
del the corresponding access control policies with RBAC