

Lecture 17

Access Control + Database Security

1

Access Control

- ❖ A “language” for expressing access control policy: who can access what, how and when...
- ❖ **Enforcement of access control**
 - ❖ Identify all resources (objects) and their granularity
 - ❖ Identify all potential users (subjects)
 - ❖ Specify rules for subject/object interaction
 - ❖ Guard them in real time

2

Model and Terminology

- ❖ **Subjects:** users or processes
- ❖ **Objects:** resources (files, memory, printers, routers, plotters, disks, processes, etc., etc.,...)

3

Focus of AC

- ❖ **What a subject is allowed to do**
- ❖ **What may be done with an object**

4

Access Modes

- ❖ "look" at object
 - ❖ Read file
 - ❖ Check printer queue
 - ❖ Print remote screen
 - ❖ Query database
 - ❖ etc., etc.
- ❖ "change" object
 - ❖ Write/append/erase file
 - ❖ Print or fax
 - ❖ Display on screen
 - ❖ etc., etc.

5

Access Rights

execute, read, append, and write

	Execute	Append	Read	Write
Observe			X	X
Alter		X		X

6

UNIX

- ❖ **execute:** execute (program) file, search directory
- ❖ **read:** read from file, list directory
- ❖ **write:** write (re-write or append) file, create or rename file in directory

7

Example: Windows NT/2000 (NTFS)

- ❖ **execute**
- ❖ **read**
- ❖ **write**
- ❖ **delete**
- ❖ **change permission**
- ❖ **change ownership**

8

AC Types

Who is in charge of setting AC policy?

- ❖ Discretionary: resource owner
- ❖ Mandatory: system-wide policy

9

Access Control Structures

- ❖ Access Control Matrix
- ❖ Capabilities
- ❖ Access Control Lists

10

Access Control Matrix

		Object		
		Bill.doc	Edit.exe	Fun.com
Subject	Alice	{0}	{execute}	{execute, read}
	Bob	{read, write}	{execute}	{execute, read, write}

11

Access Control Lists

Keep access rights to an object with that object:

- **ACL for bill.doc:**
 - Bob: read, write
- **ACL for edit.exe:**
 - Alice: execute;
 - Bob: execute
- **ACL for fun.com:**
 - Alice: execute, read;
 - Bill: execute, read, write
 - As many ACLs as there objects
 - Each ACL either signed or stored in protected place
 - Hard to Manage

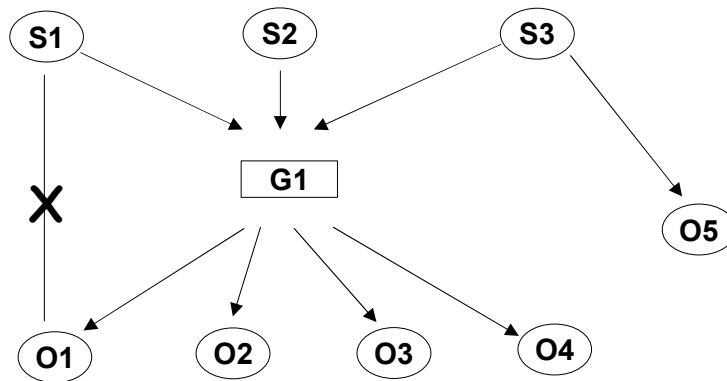
12

Access Control Lists (1)

- ❖ **Managing access rights can be difficult**
- ❖ **Groups can be helpful...**
- ❖ **Groups simplify definition of access control policies**

13

Access Control Lists (2)



14

Capabilities (1)

- ❖ Capabilities are associated with discretionary access control
- ❖ Reason: difficult to get full view of who has permission to access an object
- ❖ Very difficult to revoke a capability (owners/objects have to keep track of all issued capabilities)

- As many capabilities as there are subject/object pairs
- Each capability either signed or otherwise protected
- Hard to revoke in a distributed setting

15

Capabilities

Keep access rights with the subject:

- ❖ Alice's capabilities:
 - ❖ [edit.exe:execute];
 - ❖ [fun.com:execute,read]
- ❖ Bob's capabilities:
 - ❖ [bill.doc:read,write]
 - ❖ [edit.exe:execute]
 - ❖ [fun.com:execute,read,write]

16

In summary

- ❖ **Centralized Systems: ACLs are better**
- ❖ **Distributed Systems: capabilities are better**
- ❖ **Protection:**
 - ❖ Operating system integrity
 - ❖ Cryptographic protection

17

ROLE BASED ACCESS CONTROL (RBAC)

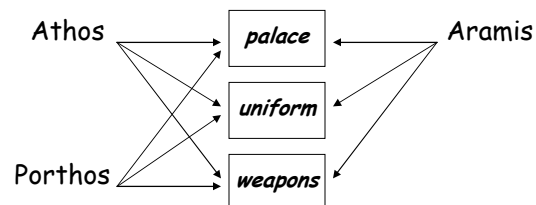
18

RBAC Basics

- Users are associated with roles.
- Roles are associated with permissions.
- A user has permission only if s/he has a role associated with that permission.

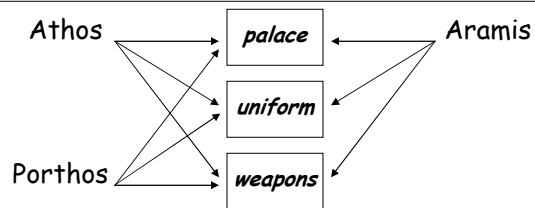
19

Example: The Three Musketeers (User/Permission Association)



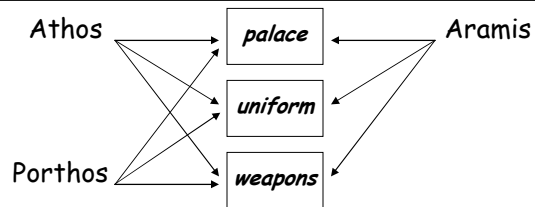
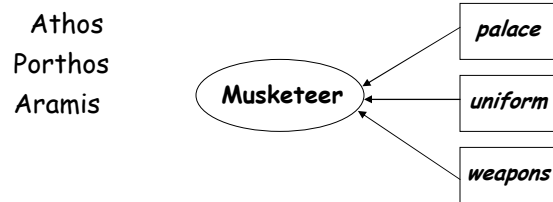
20

Example: The Three Musketeers (RBAC)



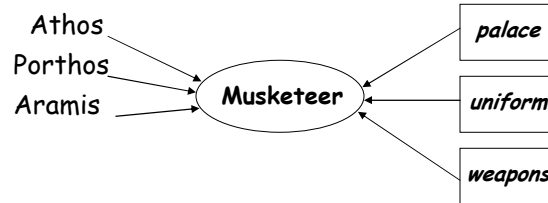
21

Example: The Three Musketeers (RBAC)

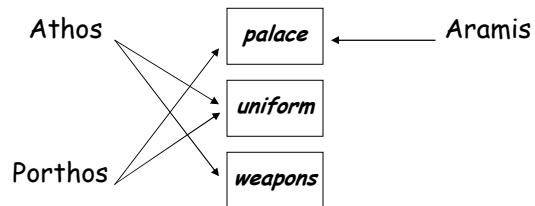


22

Example: The Three Musketeers

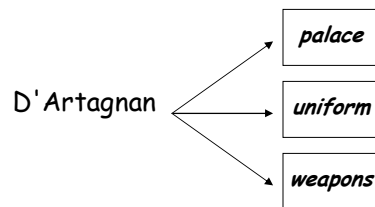
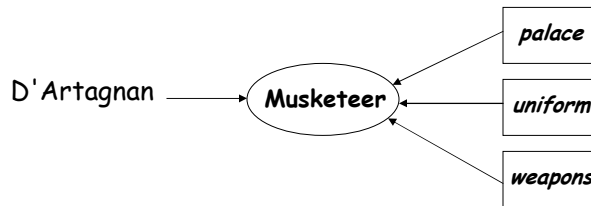


Here RBAC doesn't work...



23

Example: (D'Artagnon becomes a Musketeer)



24