Lecture 4

Friday, 16 September 2022 15:31

role - based access control:

organizations constantly change

number of users & apps increases

earlier: DAC { capability lists : store perms with users / subjects ; access control lists : store perms with file / objects } → not scalable

↓

in practice, access depends on job function, not identity

→ role

RBAC : describe organizational AC policies based on job functions

roles for job function

permission for roles

users have access to objects based on roles

RBAC principles :

1. least privilege → give no more permission than necessary ; can be abused otherwise...
2. separation of duties → users can abuse their position within an organization
   ↓ require more than 1 user to perform critical tasks

summary → base policies on duties and responsibilities of job functions

advantage: changing user's job function is possible by updating (assigned) roles instead of individual permissions

RBAC$_0$: main concepts

user

role

permission

session → roles & permission 'in use' at a given time

Exercise

	prescription	history	image scan
alice	read write	read	insert
Bob	read write	read	
Charlie		read	insert
David	read		

RBAC$_1$: role hierarchy

roles subsume others

inheritance of roles → granting perm to role ⇒ grant perm to specialization as well

---

Exercise

	p_1	p_2	p_3	p_4	p_5
u_1	✓	✓			
u_2	✓	✓	✓		
u_3	✓	✓		✓	
u_4	✓	✓	✓	✓	✓

in green : from 'own' role

in lime : inherited permission

Sometimes, we use dominance instead of specialization

→ bottom-up inheritance

→ top-down inheritance

RBAC$_2$: constraints

mutually exclusive roles for separation of duties → ensure two or more users are involved in performing tasks → ensure certain perms combinations do not exist
↓ e.g. ordering materials & approving payments

cardinality constraints → maximum number of users per role or roles per user

prerequisite roles

static separation of duty : restrict permissions that can be assigned

dynamic separation of duty : restrict permissions that can be exercised by a user

ssod ({$perm_1$, $perm_2$, ..., $perm_n$}, m)

at least m users are required to perform all n permissions

ssod can be too restrictive; e.g. you might want only that someone cannot approve their own orders

dynamic separation of duty (dsod)

object - based dsod
→ user may not act on object they have already acted on

history - based dsod

Static mutually exclusive roles : restrictions on role assignment

smer (rs, n) → user cannot have more than n roles from set rs

dynamic mutually exclusive roles : restriction on role activation

dmer (rs, n) → user cannot activate more than n roles from set rs.

cardinality constraints → at most / at least / exactly k users should belong to role

prerequisite constraints → user can only be assigned to a role if they are already assigned to another role