Securitatea sistemelor distribuite

Partea a 2-a

Controlul Accesului

Bazat pe 2 premise

- identificarea corecta a utilizatorului
  - nici un utilizator sa nu poata lua drepturile de acces ale altuia
  - asigurata prin autentificare
- informatia despre drepturile de acces este protejata contra modificarilor neautorizate

Model de securitate

- o reprezentare mai precisa si mai detaliata a unei politici de securitate
- folosit ca referinta pentru construirea securitatii si pentru evaluarea ei

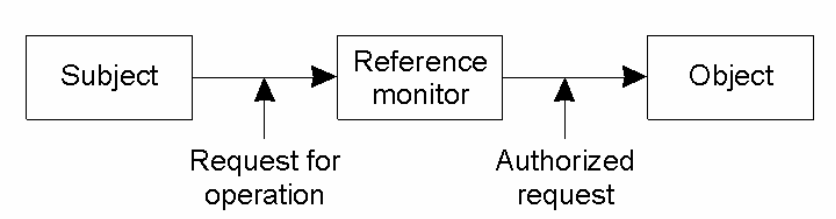
Controlul Accesului – Elemente de baza

Problema: Are subiectul s dreptul de access a la obiectul o?

- Tuplul (s, o, a) constituie Autorizatia
- Controlul accesului este o functie f (s, o, a) care intoarce *true* sau *false*

Reference monitor implementeaza aceasta functie

- toate cererile senzitive trec prin Reference monitor
- monitorul decide daca operatia poate continua



Clasificarea modelelor de acces

Discretionare sau mandatorii

- Discretionary Access Control (DAC)
  - utilizatorii pot transfera altora drepturile pe care le detin, la discretia lor
- Mandatory Access Control (MAC)
  - utilizatorii nu pot transfera drepturile pe care le detin

Controleaza accesul sau fluxul de informatii

- fluxul de informatii
  - modele multinivel
- acces
  - matrice de acces
  - RBAC – Role Based Access Control

Model de confidentialitate multinivel

Fiecare obiect
- are un nivel de senzitivitate sau rang:
  - neclasificat
  - restrictionat
  - confidential
  - secret
  - top secret
- este asociat cu unul sau mai multe proiecte (compartimente)
  - accesul este permis celor care "trebuie sa stie" pentru ca lucreaza in proiectul respectiv

Clasa de acces a unui obiect este combinatia <rang, compartimente>

Fiecare subiect
- are o autorizare (clearance) exprimata ca o clasa de acces <rang, compartimente>
  - la ce rang de senzitivitate are acces
  - in ce proiecte lucreaza

Relatia de dominanta intre clase de acces
- Ci domina Ck (sau Ck este dominat de Ci)
  Ci $\geq$ Ck $\Leftrightarrow$
  rang (Ci) $\geq$ rang (Ck) si
  compartimente (Ci) $\supseteq$ compartimente (Ck)
- Ci domina strict Ck, Ci > Ck $\Leftrightarrow$
  rang (Ci) > rang (Ck) si
  compartimente (Ci) $\supset$ compartimente (Ck)
- Ci si Ck sunt incomparabile Ci <> Ck daca
  nici Ci $\geq$ Ck nici Ck $\geq$ Ci

Exemple

Clase de acces

C1 = (TopSecret, {Nuclear, Armata})

C2 = (TopSecret, {Nuclear})

C3 = (Confidential, {Armata})

$C1 \geq C2$

$C1 > C3$

TopSecret > Confidential si

$\{Nuclear, Armata\} \supset \{Armata\}$

C2 < > C3

Modelul Bell-La Padulla (BLP)

Pastreaza secretul - previne divulgarea neautorizata a informatiei; ex. Militare

Securitatea simpla (*no-read-up*)

- Subiectele au acces doar la informatia pentru care au clasa de acces necesara
- Formal: Un subiect s are acces *read* la un obiect o doar cand clasa sa de acces domina clasa obiectului

$$C(s) \geq C(o)$$

Proprietatea Star (*) (*no-write-down*)

- Previne fluxul de informatie inspre obiecte cu clase de acces inferioare sau incomparabile
- Formal: Un subiect s care are acces *read* la un obiect p, poate avea acces *write* la un obiect o doar cand clasa de acces a lui o domina clasa lui p

$$C(o) \geq C(p)$$

Exemplu

Rang	Subiect	Obiect
Top secret	Ion, Rodica	Fisiere de personal
Secret	Sanda, Vasile	Fisiere e-mail
Confidential	Costi, Ioana	Fisiere log
Neclasificat	Mara, Mihai	Fisiere numere telefon

Senzitivitatea scade de sus in jos

Securitate simpla

Ioana nu are acces la Fisiere de personal

Rodica are acces la toate fisierele

Proprietatea-*

Rodica poate citi Fisiere de personal dar nu le poate scrie in Fisiere log pentru accesul Ioanei

Modelul de integritate Biba

Defineste nivele de integritate I analoage nivelelor de senzitivitate

Integritatea simpla

- Subiectul s poate modifica (acces *write*) un obiect o care are integritatea mai mica

$$I(s) \geq I(o)$$

- Justificare: un subiect cu integritate mai mica ar scadea integritatea obiectului modificat de el

Proprietatea-*

- Daca subiectul s are acces *read* la obiectul p cu integritatea I(p), atunci el poate avea acces *write* la obiectul o doar daca

$$I(p) \geq I(o)$$

- Justificare: Obiectul p ar scadea integritatea obiectului o

Caracteristici comune

Modelele Bell La-Padula si Biba

- nu specifica modul de definire sau de modificare a claselor de acces si de autorizare
- nu trateaza delegarea sau transferul drepturilor de acces
- topici tratate de alte modele
- ideea generala (Graham-Denning):
  - definirea unei matrice de acces care sa specifice drepturile de acces pentru fiecare combinatie de subiecte si obiecte

Modelul Matricei de Acces

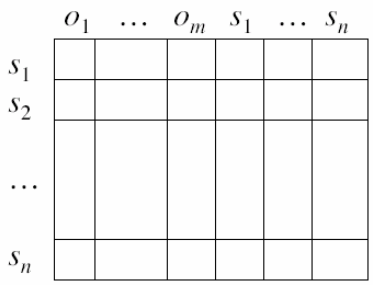
Modelul este definit in termeni de stari si tranzitii
- o stare este reprezentata de o matrice
- tranzitiile sunt descrise prin actiuni

subiecte	S = {s1,...,sn}	
obiecte	O = {o1,...,om}	
drepturi	R = {r1,...,rk}	
intrari	A [s,o] $\subseteq$ R	subiectul s are drepturile {r1,...,rk} asupra obiectului o

Graham-Denning

drepturile sunt definite ca

- actiuni A [s,o] pe care subiectul s le poate executa asupra obiectului o
- actiuni A[si,sj] pe care subiectul si le poate executa asupra subiectului sj



Exemplu

Drepturile proceselor P1 si P2 asupra

fisierelor f1 si f2

proceselor P1 si P2

	f1	f2	P1	P2
P1	read write own	read	read write execute own	write
P2	append	read own	read	read write execute own

Modelul Graham - Denning

- fiecare obiect are un subiect **propietar** (owner) cu drepturi speciale
- fiecare subiect are un alt subiect cu drepturi speciale (**controlor**)

Actiuni primitive executate de subiectul x: r* inseamna ca dreptul **r** transmis de **x** lui **s** este *transferabil*, adica s poate transfera r sau r* altor subiecte

Actiune	**Preconditie**	**Efect**
create object o	-	Adauga coloana o la A; adauga *Owner* la A[x,o]
delete object o	*Owner* in A[x,o]	Sterge coloana o
create subject s	-	Adauga linia s si col s la A; adauga *Control* la A[x,s]
delete subject s	*Control* in A[x,s]	Sterge linia s si col s
read access rights of s on o	*Control* in A[x,s] sau Owner in A[x,o]	Citeste A[s,o]
grant access right r to s on o	*Owner* in A[x,o]	Adauga r la A[s,o]
delete access right r of s on o	*Control* in A[x,s] sau *Owner* in A[x,o]	Sterge r din A[s,o]
transfer access right r or r to s on o*	r* in A[x,o]	Adauga r sau r* la A[s,o]

Alte modele – HRU

The Harrison-Ruzzo-Ullman (HRU) bazat pe:

- S set de subiecte
- O set de obiecte
- R set de drepturi de acces
- O matrice de acces $M = (M_{so})_{s \in S, o \in O}$
- intrarea M_{so} este un subset din R specificand drepturile subiectului s asupra obiectului o

Operatii primitive

- **enter** r into M_{so}
- **delete** r from M_{so}
- **create subject** s
- **delete subject** s
- **create object** o
- **delete object** o

Comenzi

Comenzile au formatul:

command $c(x_1,.....,x_k)$
if r_1 in $M_{s1,o1}$ **and**
if r_2 in $M_{s2,o2}$ **and**
⋮
if r_m in $M_{sm,om}$
then $op_1,.....,op_n$
end

$s_1,.....,s_m$ si $o_1,.....,o_m$ sunt subiecte si obiecte care apar in lista de parametri $x_1,.....,x_k$

Daca toate conditiile sunt indeplinite atunci se executa lista de operatii

Comenzi mono-operatii

Exemple

Crearea unui fisier:

command CREATE_FILE(s,o)
create object o
enter own into *A*[s,o]
end CREATE_FILE

Transferul unui drept; de ex "read":

command CONFER_READ(s1,o,s2)
if own ∈ *A*[s1 ,o]
then enter read into *A*[s2 ,o]
end CONFER_READ

Revocarea unui drept; de ex "write":

command REVOKE_WRITE(s1,o,s2)
if (own ∈ *A*[s1 ,o]) and (write ∈ *A*[s2 ,o])
then delete write from *A*[s2 ,o]
end REVOKE_WRITE

Ce rezolva HRU - Sisteme de Protectie

Un sistem de protectie:

- Set finit de drepturi
- Set finit de comenzi

Un sistem de protectie este unul stari-tranzitii

Fiecare stare este descrisa de matricea de acces

Definitie. O stare lasa sa se *scurga* dreptul *r* daca exista o comanda *c* care adauga dreptul *r* intr-o intrare din *M* care anterior nu continea *r*. Mai precis, exista *s* si *o* astfel ca $r \notin M_{so}$ si, dupa executia lui *c*, $r \in M'_{so}$

Nota: Faptul ca un drept se poate "scurge" nu este neaparat rau; multe sisteme permit subiectilor sa delege drepturi de acces altor subiecti

Exemplu de sistem "nesigur"

Fie comenzile:

command grant_execute (*s*,*p*,*f*)
  if own in $M_{s,f}$
  then enter execute into $M_{p,f}$
end

command modify_own_right (*s*,*f*)
  if execute in $M_{s,f}$
  then enter write into $M_{s,f}$
end

Exemplu

Bob a dezvoltat o aplicatie si vrea ca ea sa fie executata de alt utilizator dar nu modificata de acesta

Sistemul anterior este nesigur; permite urmatoarea situatie:

- Bob: grant_execute (Bob, Tom, P1)
- Tom: modify_own_right (Tom, P1)

Face ca in M, intrarea $M_{Tom,P1}$ sa contina dreptul de acces w

Siguranta in modelul HRU

Definitie. O stare *M* intr-un sistem de protectie este sigura relativa la un drept *r* daca nici o secventa de comenzi nu poate transforma *M* intr-o stare in care se scurge *r*.

Teorema. Data fiind *M* si dreptul *r*, verificarea sigurantei lui *M* relativa la *r* este o problema nedecidabila in cazul general.

Problema sigurantei:

- este decidabila pentru sisteme de protectie mono-operatie
- nu este intotdeauna decidabila pentru alte tipuri de sisteme de protectie
  - protectia in UNIX cere mai mult de o operatie per comanda
- poate fi reformulata astfel:

  Este decidabil daca un *subiect* ar putea obtine vreodata un anumit *drept* relativ la un *obiect* ?

Modelul Take-Grant

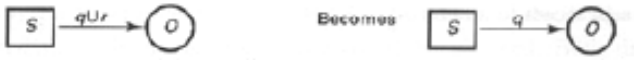
Are patru operatii primitive

- reprezentate prin grafuri
  - subiecte si obiecte -> noduri
  - drepturi -> arce etichetate de la subiect la obiect

Urmatoarele operatii sunt executate de s

create(o, r):

revoke(o,r):



grant(o,p,r):

Subiectul s "deleaga" lui o dreptul de acces r asupra lui p.
Preconditii:
s are drept de delegare (Grant) a unor drepturi catre o
s are dreptul r asupra lui p

take(o,p,r):

Subiectul s preia de la o dreptul de acces r asupra lui p

Preconditii

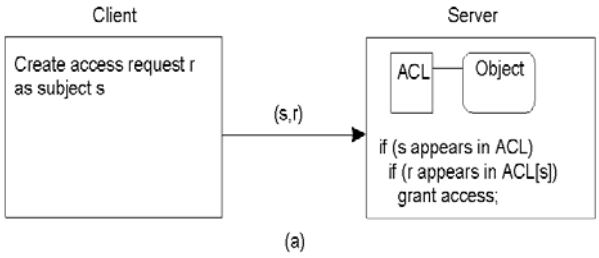
- s are drept de preluare (Take) de drepturi de la o
- o are dreptul r asupra lui p

Cu acest model se poate decide daca

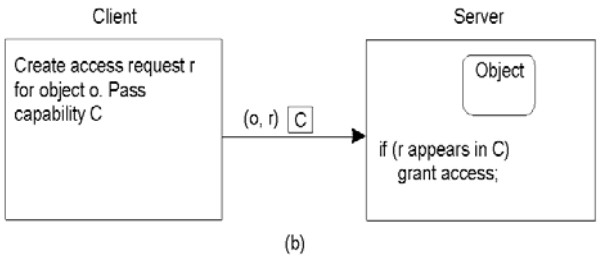
- un subiect poate partaja un obiect cu un alt subiect
- un subiect poate "fura" accesul la un obiect de la un alt subiect

Implementare Matrice Control Acces

ACL
fiecare obiect pastreaza MCA[*,O]



Capabilitati
fiecare subiect are capabilitatile din MCA[S,*]



Utilizarea modelelor de securitate

- Orange Book introduce
  - clase de securitate: A la D
  - tehnici de clasificare pentru sisteme de operare
- Nivel D: nici o cerinta, securitate minimala
  - MSDOS, Windows 95/98/Me
- Nivel C: protectie discretionara
  - C1: arhitectura cu OS mod protejat, autentificare utilizatori sau grupuri, control discretionar acces, testare, documentatie (proiectare, testare, instalare, utilizare)
  - C2: control discretionar acces la nivel utilizatori (individual), obiecte re-initializate la zero inainte de reutilizare (stergere informatie veche), auditare evenimente securitate
  - Unele variante de Unix si NT au nivel C2

Nivele Orange Book

- Nivele B & A
  - B1: etichete pentru clasa obiectelor si autorizarea subiectilor, utilizat in sisteme de date clasificate, respecta Bell-LaPadula
  - B2: protectie structurata - sistem proiectat top-down in maniera modulara, verificabila, nu permite covert channel
  - B3: domenii de securitate - ACLs cu utilizatori si grupuri, TCB verificata formal, auditare, recuperare crash fara compromitere securitate
  - A1: ca B3 dar model formal de protectie, demonstrare corectitudine model, demo formala ca implementarea respecta modelul
- Clase inalte greu de atins
  - — OSes pentru aplicatii militare au rate ridicate

Probleme cu modelele "clasice"

Matrice acces (respectiv ACL, C-list):

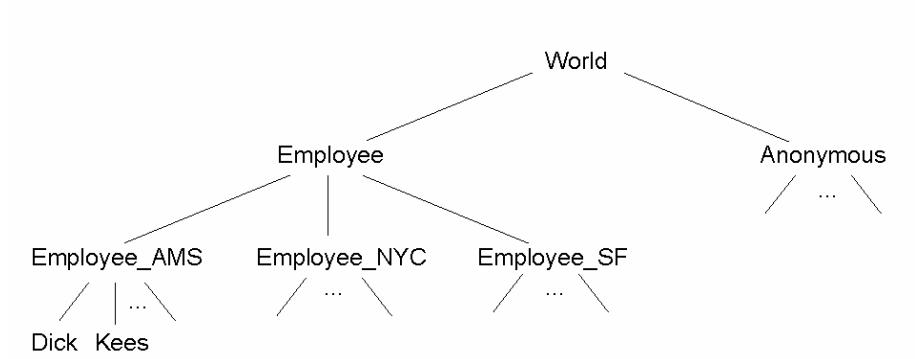
Nu pot reprezenta modele de acces mai complexe
- de ex. bazate pe reguli cum ar fi competenta,
- cele mai reduse privilegii sau
- conflict de interese

Nu suporta schimbari dinamice
- modificarea drepturilor unui subiect necesita inspectarea ACL a fiecarui obiect
- determinarea subiectilor care au acces la un obiect necesita inspectarea tuturor listelor de Capabilitati

Nu pot gestiona drepturi de acces determinate de continutul, atributele obiectelor sau de context

Nu inregistreaza utilizarea permisiunilor
- Un subiect poate utiliza permisiunile ori de cate ori

Domenii de Protectie



domeniu de protectie = set de perechi (obiect, drept acces)

Implementari posibile

- **grupuri de utilizatori (organizate ierarhic)**
- **certificate**
- **roluri**

RBAC - Role Based Access Control

Rolul este un mecanism de grupare a subiectilor dupa anumite atribute: ocupatie, responsabilitati, functii

Rolurile (si nu subiectii) au asociate permisiuni

- permit predefinire roluri
- relatia rol – permisiuni se schimba mai rar ➔ administrare mai usoara

Fiecare subiect are un set de roluri asignate

- asignarea / revocarea de roluri poate fi administrata usor
- un utilizator poate trece de la un rol la altul intr-o sesiune
- un utilizator poate avea mai multe roluri simultan

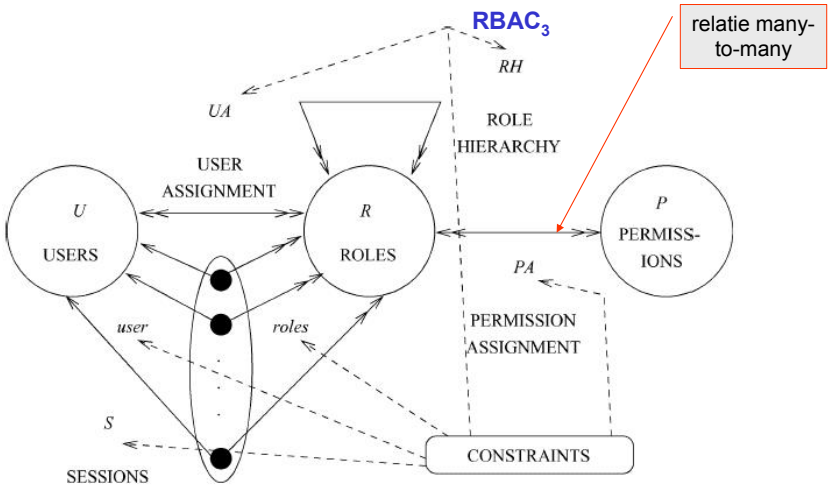
Rolurile sunt supuse unor reguli suplimentare

- Ex. separare statica sau dinamica a obligatiilor legale (constrangeri)
  Ex.: o persoana sa nu poata initia o plata si autoriza plata
  - Static Separation of Duty Relations – ex. rolurile A si B sa nu fie asignate ambele unui utilizator
  - Dynamic Separation of Duty Relations - ex. rolurile A si B sa nu fie active in acelasi timp pentru un utilizator

Rolurile pot fi ierarhizate

- Ex. un instructor poate avea drepturile unui student si in plus alte drepturi
- simplifica gestiunea drepturilor (ex. prin mostenire)
- categorii
  - limitate (ex. ierarhii arborescente) sau
  - generale (ex. mostenire multipla)

RBAC = Familie de modele



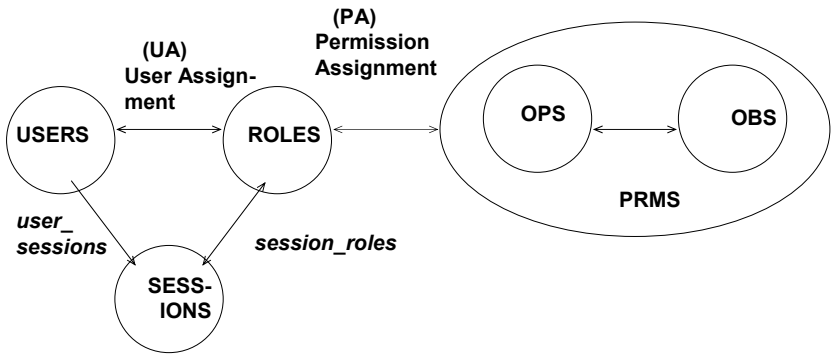
$RBAC_0$ - Core RBAC

$RBAC_1$ - Hierarchical RBAC: ierarhii de roluri

$RBAC_2$ - Static & Dynamic Separation of Duty Relations: constrangeri

$RBAC_3$ – include $RBAC_1$ si $RBAC_2$

Core RBAC



User – persoana, masina, proces, agent software, etc.

Role – functie in contextul unei organizatii, asociata cu autoritatea si responsabilitatile sale

Permission – un mod de acces asupra obiectelor (autorizare, drept de acces, privilegiu)

Session – instanta particulara a unei conexiuni a utilizatorului cu sistemul; defineste subsetul de roluri active.

Model de Referinta pentru Core RBAC

- o multime de seturi de elemente de baza
  USERS, ROLES, OPS (operatii), *OBS* (obiecte),
  SESSIONS (set de sesiuni)
- un set de relatii RBAC pe aceste seturi (asocierile valide) si
- un set functii de mapare care dau instantele de membri dintr-un set care corespund unui membru din alt set

User Assignment - $UA \subseteq USERS \times ROLES$

$assigned\_users(r{:}ROLES) = \{u \in USERS \mid (u, r) \in UA\}$

Permissions - $PRMS = 2^{(OPS \times OBS)}$

Permission Assignment - $PA \subseteq PERMS \times ROLES$

$assigned\_permissions(r : ROLES) \rightarrow 2^{PRMS}$

permisiuni → operatii $Op(p: PRMS) \rightarrow \{op \subseteq OPS\}$

permisuni → obiect $Ob(p: PRMS) \rightarrow \{ob \subseteq OBS\}$

$user\_sessions\ (u: USERS) \rightarrow 2^{SESSIONS}$

$session\_roles\ (s: SESSIONS) \rightarrow 2^{ROLES}$

$session\_roles\ (s_i) \subseteq \{r \in ROLES \mid (session\_users\ (s_i), r) \in UA\}$

$avail\_session\_perms(s{:}SESSIONS) \rightarrow 2^{PRMS}$

permisiunile disponibile unui utilizator intr-o sesiune =

$$\bigcup_{r \in session\_roles(s)} assigned\_permissions(r)$$

Politici de activare

Single-role activation (SRA) = un rol activat

Multi-role activation (MRA) = mai multe roluri pot fi activate intr-o sesiune

se pot folosi constrangeri "separation of duty" pentru a restrictiona activarea concurenta a anumitor roluri

Specificatie Cerinte pentru Core RBAC

Functii Administrative

Creare si Intretinere seturi de elemente

Creare si Intretinere Relatii

Functii Suport Sistem (management sesiune si decizii de control al accesului)

- CreateSession - Creaza User Session si da utilizatorului un set de roluri active implicite
- AddActiveRole – Adauga un rol ca rol activ in sesiunea curenta
- DropActiveRole – Elimina un rol activ din sesiunea curenta
- CheckAccess – Determina daca subiectul are permisiunea sa execute o anumita operatie asupra obiectului.

Functii Administrative de informare (Review)

(M – Mandatory, O – Optional)

- AssignedUsers (M) - Intoarce setul de utilizatori asignat unui rol dat
- AssignedRoles (M) - Intoarce setul de roluri asignate unui utilizator dat
- RolePermissions (O) - Intoarce setul de permisiuni garantate unui rol dat
- UserPermissions (O) - Intoarce setul de permisiuni pe care un utilizator le are prin rolurile sale
- SessionRoles(O) - Intoarce setul de roluri active asociate cu o sesiune
- SessionPermissions (O) - Intoarce setul de permisiuni disponibile in sesiune (reuniunea tuturor permisiunilor asignate rolurilor active din sesiune)

Ce realizeaza RBAC?

Leaga permisiunea direct de rol (si indirect de utilizator)
- orice utilizator u care are permisiunea p are un *rol activ* in sesiunea s:
  - CheckAccess (u,p) => (exista r $\in$ AssignedRoles(u) si r $\in$ SessionRoles (s))

Orice utilizator u isi poate asuma doar roluri r autorizate
- AssignedRoles (u) $\subseteq$ {r | u $\in$ AssignedUsers(r)}

Utilizatorii nu pot avea permisiuni in afara celor date de rolurile lor
- CheckAccess(u,p) => p $\in$ UserPermissions(u)

Ierarhie de roluri:
- un rol poate include alte roluri: un instructor are permisiunile studentilor si, in plus, alte permisiuni; r_i include rolul r_k
- $r_i > r_k \Leftrightarrow$ RolePermissions (r_i) $\supseteq$ RolePermissions (r_k) si AssignedUsers (r_k) $\supseteq$ AssignedUsers (r_i)

Separarea sarcinilor (excludere mutuala a permisiunilor)
- roluri incompatibile: nu pot fi asumate simultan de un utilizator
- r_i, r_k incompatibile => $r_i \in$ AssignedRoles(u) => $r_k \notin$ AssignedRoles(u)

Caracteristici RBAC

Calitati
- utilizatorul poate trece usor de la un rol la altul in cursul unei sesiuni, fara a schimba structura accesului
  - rezultat: RBAC mai scalabil ca matricile de acces
- permite tratarea conflictelor de interese (rolurile au reguli stricte)
- reduce overhead la administrare securitate la nivel de utilizator, obiect, permisiune

Relatia cu MAC si DAC
- RBAC este un model neutru fata de politicile de control al accesului
- poate coexista cu MAC sau DAC
  - accesul trebuie sa fie permis de RBAC si MAC / DAC

Utilizare
- Microsoft Active Directory, SELinux, FreeBSD, Solaris, Oracle DBMS, PostgreSQL 8.1, SAP R/3