Lab 6 Controlul Accesului: Matricea de Control al Accesului

Controlul Accesului:
1. Mecanisme care permit sau interzic accesul unei entitati la o resursa.
2. Bazat pe doua premise
   – identificarea corecta a utilizatorului (nici un utilizator nu poata lua drepturile de acces ale altuia, asigurata prin autentificare)
   – informatia despre drepturile de acces este protejata contra modificarilor neautorizate

Unul dintre cele mai simple mecanisme: ***Matricea de Control al Accesului.***

Modelul este reprezantat prin doua dimensiuni: subiecti si obiecte. In fiecare intrare S × O: multime de drepturi/permisiuni – un subiect poate fi si obiect (e.g. un proces): dreptul de a citi/scrie (comunica) / executa un alt proces.

Subiect = user, proces al computerului ce actioneaza din partea userului (activ)
Obiect = resursa accesibila pe un sistem de calcul (pasiv)
Operatie, actiune = proces activ invocat de un subiect
Permisie, drept = autorizatie de a realiza o actiune (operatie)

Definirea unei matrice de acces care sa specifice drepturile de acces pentru fiecare combinatie de subiecti si obiecte. Elemente de baza: Are subiectul **s** dreptul de access **d** la obiectul **o**? – Tripletul **(s, o, r)** constituie Autorizatia

Clasificarea modelelor de acces
1. Discretionary Access Control (DAC) - utilizatorii pot transfera altora drepturile pe care le detin, la discretia lor. Permite utilizatorilor individuali (tipic: owner) sa seteze mecanismul prin care se permite/interzice accesul.
2. Mandatory Access Control (MAC) - utilizatorii nu pot transfera drepturile pe care le detin. Mecanismul de acces e determinat de sistem, nu poate fi schimbat de utilizatori.
3. Role Based Access Control (RBAC) - pe baza functiilor pe care un utilizator le poate realiza, drepturile de acces nu pot fi transferate. Politica este determinata de sistem, in functie de rolul activ al unui subiect.

Mecanisme care controleaza accesul: Matricea de control al accesului, Role Based Access Control

Modelul Matricei de Control

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

Multimea subiectilor $S = \{s_1,...,s_n\}$ este o submultime a multimii obiectelor $O = \{o_1= s_1,..., o_n= s_n, o_{n+1}, ..., o_{n+m}\}$. $R = \{r_1,...,r_k\}$ multimea drepturilor. Intrari: $A [s,o] \subseteq R$ subiectul s are drepturile $\{r_1,...,r_k\}$ asupre obiectului o.

Drepturile sunt definite ca
- actiuni A [s,o] pe care subiectul s le poate executa asupra obiectului o

- actiuni A[s_i,s_j] pe care subiectul s_i le poate executa asupra subiectului s_j

	s_1	s_2	...	s_n	o_{n+1}	…	o_{n+m}
s_1							
s_2							
...							
s_n							

Exemplu Drepturile proceselor $p_1$si p_2 asupra fisierelor f_1, f_2 , proceselor $p_1$si p_2

	p_1	p_2	f_1	f_2
p_1	all	r, w, x	r	r,w,d
p_2	x	all	r,w,d	r

Operatii primitive

1. enter r into (X_s, X_o) unde r - drept, (X_s, X_o) celula a matricii; // adauga dreptul r in celula matricii (X_s, X_o) (meaning: r access right of X_s on X_o)
2. delete r from (X_s, X_o) unde r – drept; // sterge dreptul r din celula matricii (X_s, X_o) (meaning: delete access right r of X_s on X_o)
3. create subject X_s; // Adauga linia X_s si col X_s la A
4. create object X_o; // Adauga coloana X_o la A
5. destroy subject X_s; // Sterge linia X_s si col X_s
6. destroy object X_o; // Sterge coloana X_o

Comenzi

1. comanda fara garda:
```
command α(X1, . . . , Xk)
        op1, . . . , opn
end
```
2. comanda cu garda:
```
command α(X1, . . . , Xk)
     if r1 in (Xs1 , Xo1 )
          . . .
     if rm in (Xsm, Xom)
         then op1, . . . , opn
end
```

unde $\{s_1, \ldots s_m, o_1, \ldots, o_m\} \subseteq \{1, \ldots, k\}$. Daca toate conditiile sunt indeplinite atunci se executa lista de 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
```

Sisteme de Protectie HRU (Harrison-Ruzzo-Ullman)

Un sistem de protectie este format dintr-un set finit de drepturi, un set finit de comenzi si un set de stari-tranzitii, unde fiecare stare este descrisa de matricea de control al accesului.

Definitie Comanda **α** scurge dreptul **r** in starea **Q** daca
- **α** se poate aplica starii **Q** sub o substitutie **σ**;
- prin aplicarea operatiilor comenzii **α** o celula a starii Q ce nu continea **r** ajunge sa contina **r**.

Starea **Q** *este nesigura* peste **r** sub sistemul de protectie **C** (multime de comenzi) daca exista o stare $\mathbf{Q_0}$ accesibila din **Q** si **C** scurge dreptul **r** in $\mathbf{Q_0}$.

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
```

Aplicatie: Bob a dezvoltat o aplicatie P_1 si vrea ca ea sa fie executata de alt utilizator (exemplu Tom) dar sa nu poata fi si modificata de acesta. Sistemul anterior este nesigur deoarece permite urmatoarea situatie: - Bob: `grant_execute (Bob, Tom, P1)`
- Tom: `modify_own_right (Tom, P1)`

Deci dupa aplicare in matricea M, intrarea $M_{Tom,P1}$ va contine dreptul de acces `write`.

Siguranta in HRU

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

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

Observatie Problema sigurantei:
1. Este decidabila pentru sisteme de protectie mono-operatie sau daca se permit doar operatii simple: nu au conditii multiple si sunt monotone (se permite crearea dar nu si distrugerea de subiecti¸ obiecte).
2. Nu este intotdeauna decidabila pentru alte tipuri de sisteme de protectie – protectia in UNIX cere mai mult de o operatie per comanda.

Exemplu

Se da urmatoarea matrice de control al accesului:

	KP	UP_1	UP_2	$file_1$
KP	c,o,d,x,r,u	c,o,d,x,r,u	c,d,x,r,u	-
UP_1	-	c,o,d,x,r,u	-	c,o,d,r,u
UP_2	-	-	d,x,r,u	-

Observatii:

- semnificatie drepturi: c=create, o=own, d=delete, x=eXecute, r=read, u=update.
- un subiect UP_i poate crea un obiect de tip $file_j$ doar daca are dreptul c asupra lui insusi ($c \in (UP_i , UP_i)$).
- un subiect UP_i poate sterge un obiect de tip $file_j$ doar daca are drepturile o, d asupra lui ($o, d \in (UP_i , file_j)$).

Cerinte:

a) Scrieti comenzile necesare pentru ca un subiect UP_i ($i = 1, 2$) sa creeze un nou obiect $file_2$ cu drepturile {o, d, r, u}; descrieti modificarile efectuate asupra matricii la aplicarea comenzilor pentru UP_1, respectiv UP_2;

b) Scrieti comenzile necesare pentru ca UP_1 sa poata sterge fisierul ($file_1$).