Recent Advances in Access Control

by

Amit Konar

Dept. of Math and CS, UMSL

Closed Policy

Examples

Give Jim read access to foo.

Give Bob r/w access to bar.

Jim: I'd like to read foo.

- Access is granted

Jim: I'd like to read bar.

- Access is denied.

Why call Closed Policy ? –Only those with access right can perform r/d operations.

Open Policy

Examples:

Deny Jim read access to foo.

Deny Bob r/w access to bar.

Jim: I' d like to read foo.- access is denied.

Jim: I'd like to read bar.- access is denied.

Why Open Policy: Because anyone except those listed here have access.

Closed Policy with Negative Authorization and Denial-takes-precedence

Examples:

Give Jim r/w access to foo.

Deny Jim w access to foo.

Jim: I'd like to read foo.- Access granted.

Jim: I'd like to write foo.- Access denied.

Denial is given priority for security reasons.

Closed Policy with Negative Authorization and Most specific Authorizations take precedence

Examples:

Give CS students r/w access to foo.

Deny Jim w access to foo.

Jim: I ‘d like to read foo. – Access granted.

Jim: I ‘d like to write foo. – Aceess denied.

Jim’s denial on write access to foo is most specific and thus takes precedence.

Closed Policy with Negative Authorization and Most Specific Authorizations take precedence

Examples:

Deny CS students r/w access to foo.

Give Jim r access to foo.

Jim: I ‘d like to read foo.-Access granted.

Jim: I’d like to write foo.- Access denied.

Here also, the most specific authorization took precedence.

Separation of Duties

Prevention by fraud and errors is by far the dominant security issue in commercial organizations.

No individual is given sufficient authority to perpetrate fraud on his/her own.

This is achieved by breaking larger actions into smaller steps executed by distinct individuals.

Static Separation of Duty

1. A clerk prepares a voucher and assigns an account.
2. The voucher and account are approved by a supervisor.
3. The check is issued by a clerk **different** from the clerk in step 1.

It will take two clerks and a supervisor to perpetrate fraud.

Dynamic Separation of Duty

Example:

A particular clerk is prohibited from performing steps 1 and 3 for the same voucher. However, he/she can perform step 1 on some vouchers and step 3 on some other ones.

A Predicate Logic Based Language for Access Control

An authorization is of the form: (s,o<sign>,a)

An authorization specification AS consists of a set of:

- authorization: **cando**
- derivation: **dercando**
- conflict resolution: **do** (avoiding over/under specification)
- Integrity: **error**

Authorization Rule

Used to specify accesses to be allowed or denied

cando(o, s,<sign> a) ← L_1& L_2& …& L_n

where L_i is in, typeof literals.

Examples: cando(file1,CS-dept., +write) **←.**

cando(file2,s, +write) ← in(s, CS-faculty).

cando(o, Secretary, +write) ←typeof(o, Letters).

cando(file1, George, -write) **←.**

Derivation Rule

Used to derive implied authorizations

dercando(o, s,<sign>a)←L_1&…&L_n.

where L_i is cando, dercando, in or typeof literal.

dercando(o, s, +a) ←cando(o, s′, +a) & in (s, s′).

dercando(o, s, -a) ←cando(o, s′, -a) & in(s, s′).

dercando(file1, s,-read) ← dercando (file2,s,read)

Resolution Rule

States that a subject must be allowed/forbidden to exercise an access:

do(o, s, <sign>a) ←L_1& …&L_n.

where L_i is cando, dercando, in, done or typeof literal.

do(file1, s +a) ←dercando(file1, s +a).

do(file1, s,+a) ←¬dercando(file1,s,-a).

Done Rule

done(o, s, R, a, t) ←.

This rule is true if a subject s with roles in R active has executed action a on object o at time t.

Useful for policies in which future accesses are based on those in the past.

Closed Policy in Authorization Specification Language (ASL)

dercando(u, o,+a) ←cando(s,o,+a) & in(u,s).

do(u, o, +a) ←dercando(u, o, +a).

do(u, o, -a) ←¬do(u,o,+a).

error(s,o,a) ← cando(s,o, -a).

Denials take precedence

$do(u,o,-a) \leftarrow dercando(u,o,+a)$
$\quad \& \; dercando(u,o,-a)$

$do(u,o,-a) \leftarrow \neg do(u,o,+a)$

Static Separation of Duty

error() ← do(s, budget, submitting)
& do(s, budget, evaluating)
& do(s, budget, approving).

Same person s cannot do submission, evaluation and aprroval of budget.

Dynamic Separation of Duty

error() ←done(u,o,R, submitting, t)

& done(u,o, $R^{/}$, approving, $t^{/}$)

& done(u,o,$R^{//}$, approving, $t^{//}$)

& typeof (o, Order).

Here, subject u plays three roles in three different times.

Formalization of Role Based Access Control

Defn.1: A **role** is a collection of job functions. Each role r is authorized to perform one or more transaction (actions in support of a job function). The set of authorized transactions for r is written as **trans(r).**

More Definitions

Defn. 2: The **active role of a subject s**, written **actr(s),** is the role that s is currently performing.

Defn. 3: The **authorized roles of a subject s**, written **authr(s),** is the set of roles that s is authorized to assume.

Defn. 4: The predicate **canexec(s, t)** is true if the subject s can execute transaction t at time t.

Rules to Execute a Transaction

Axiom 1: Let S be the set of subjects and T be the set of transactions. The **rule of role assignment is**

$(\forall s \in S)(\forall t \in T)\ [canexec(s, t) \rightarrow actr(s) \neq \varnothing]$

This means that if a subject can execute any transaction, then he has an active role.

Axiom 2: Let S be the set of subjects. Then the **rule of role authorization** is

$(\forall s \in S)\ [actr(s) \subseteq authr(s)]$.

This means that the subject must be authorized to assume its active role.

Rule of Transaction Authorization

Axiom 3: Let S be the set of subjects and T be the set of transactions. The rule of transaction authorization is

$(\forall s \in S)\ )(\forall t \in T)\ [canexec(s, t) \rightarrow t \in trans\ (actr(s))]$.

It means that the subject cannot execute a transaction for which its current role is not authorized.