# Rewrite-Based Access Control Policies in Distributed Environments


Maribel Fernández

King's College London

Joint work with Clara Bertolissi (LIF, Univ. Marseilles)


12th CREST Open Workshop - Security and Code
April 2011

# Motivations - Access Control

Access control is of fundamental importance in computer security.

Formal specifications of access control models and policies make it possible to

- compare policies rigorously,
- understand the consequences of changes
- prove properties of policies.

# Motivations - Authorisation models

Over the last few years, a wide range of access control models have been developed.

- Access Control Lists
- Discretionary Access Control
- Mandatory Access Control
- Role-based Access control
- Task-based Access Control
- Event-based Access Control
- ...

# Motivations - Authorisation models

Over the last few years, a wide range of access control models have been developed.

- Access Control Lists
- Discretionary Access Control
- Mandatory Access Control
- Role-based Access control
- Task-based Access Control
- Event-based Access Control
- ...

# MetaModel

Barker [Sacmat09] proposes a general meta-model for access control based on the primitive notion of a category.

Advantages:

- a core set of principles of access control, can be specialised for domain-specific applications
- abstracts away many of the complexities of specific access control models
- helps to understand and write policies

# Rewriting

We propose an operational semantics for the access control metamodel, using term rewriting.

Advantages:

- Expressivity: rewriting systems have been used to specify computational paradigms and access control models (e.g. ACL, RBAC, dynamic models s.a. DEBAC and ASAC)
- Well-developed theory: rewriting techniques used to prove properties of policies
- Tools such as ELAN, MAUDE, CiME, TOM, etc. for rapid prototyping of access control policies.

# Contributions

- rewrite-based specification of the category-based access control metamodel — operational semantics
- technique to prove totality and consistency of access control policies
- encoding of well-known access control models: RBAC, MAC, DAC and DEBAC (expressive power)
- A distributed version of the metamodel:
  - centralised or distributed access request evaluation
  - distributed federations where each site may run a different access control policy (possibly with a different access control model)

# This talk

- The category-based meta-model $\mathcal{M}$
- Introduction to term rewriting
- Rewrite-based specification of $\mathcal{M}$:
    - definition
    - request evaluation
    - properties
    - expressive power
- Distributed metamodel
- Conclusions and future work

## The metamodel $\mathcal{M}$

Based on the notion of category:
a class, group, or domain, to which entities or concepts belong

Particular cases: *role*, *security clearance*, discrete *measure of trust* and other standard groupings used in access control

# The metamodel $\mathcal{M}$

Entities in $\mathcal{M}$ are denoted by constants:

- countable set $\mathcal{C}$ of categories: $c_0, c_1, \ldots$
- countable set $\mathcal{P}$ of principals: $p_1, p_2, \ldots$
- countable set $\mathcal{A}$ of actions: $a_1, a_2, \ldots$
- countable set $\mathcal{R}$ of resources: $r_1, r_2, \ldots$
- countable set $\mathcal{S}$ of situational identifiers (locations, times)

Entities are assigned to distinct classes or groups: categories.

# $\mathcal{M}$: relationships between entities

- Principal-category assignment $\mathcal{PCA}$:
  $(p, c) \in \mathcal{PCA}$ iff $p \in \mathcal{P}$ is assigned to $c \in C$
- Permissions $\mathcal{ARCA}$:
  $(a, r, c) \in \mathcal{ARCA}$ iff action $a \in A$ on resource $r \in R$ may be performed by principals in the category $c \in \mathcal{C}$
- Authorisations $\mathcal{PAR}$:
  $(p, a, r) \in \mathcal{PAR}$ iff $p \in \mathcal{P}$ may perform action $a \in \mathcal{A}$ on resource $r \in \mathcal{R}$

$\mathcal{PAR}$ defines the set of authorisations that hold according to the policy that specifies $\mathcal{PCA}$ and $\mathcal{ARCA}$

# Axioms

Core axiom:

$$
\begin{array}{ll}
(a1) & \forall p \in \mathcal{P},\ \forall a \in \mathcal{A},\ \forall r \in \mathcal{R},\ \forall c \in \mathcal{C}, \\
 & (p, c) \in \mathcal{PCA} \wedge (\exists c' \in \mathcal{C}, c \subseteq c' \ \wedge (a, r, c') \in \mathcal{ARCA}) \\
 & \hfill \Rightarrow (p, a, r) \in \mathcal{PAR}
\end{array}
$$

where $\subseteq$ is a relationship between categories, e.g. equality, set inclusion, ...
Operationally, $(a_1)$ is realised through a set of function definitions

# Term Rewriting

Term rewriting systems are defined by a set of terms and a set of rewrite rules that are used to 'reduce' terms.

*Terms*: $T(\mathcal{F}, \mathcal{X})$ built up from a signature $\mathcal{F}$ (*function symbols* with fixed arities) and a set of *variables* $\mathcal{X}$.
$Var(t)$ denotes the set of variables occurring in $t$.

*Rewrite rules*: $R = \{l_i \to r_i\}_{i \in I}$, where $l_i, r_i$ are terms, $l_i \notin \mathcal{X}$, and $Var(r_i) \subseteq Var(l_i)$.

Rewrite step in $R$:
$t \to_R u$ (reflexive-transitive closure: $t \to_R^* u$).

Irreducible terms are in *normal form*.

# Term Rewriting: Example

- Natural numbers: $0$, $s(0)$, $s(s(0))$, $\ldots$
  Booleans: $\mathrm{True}, \mathrm{False}$
  Lists of numbers: $nil$, $cons(0, nil)$, $cons(s(0), nil)$, $\ldots$

- Conditional:

$$\mathit{if\text{-}then\text{-}else}(\mathrm{True}, X, Y) \to X$$

$$\mathit{if\text{-}then\text{-}else}(\mathrm{False}, X, Y) \to Y$$

# Term Rewriting: Example

Operators on sets represented as lists:

$$\begin{array}{rcl}
\mathrm{Union}(\mathrm{nil}, x) & \to & x \\
\mathrm{Union}(\mathit{cons}(x, y), z) & \to & \mathit{if}\ \mathrm{In}(x, z)\ \mathit{then}\ \mathrm{Union}(y, z) \\
 & & \mathit{else}\ \mathit{cons}(x, \mathrm{Union}(y, z)) \\
\mathrm{Inter}(\mathrm{nil}, x) & \to & \mathrm{nil} \\
\mathrm{Inter}(\mathit{cons}(x, y), z) & \to & \mathit{if}\ \mathrm{In}(x, z)\ \mathit{then}\ \mathit{cons}(x, \mathrm{Inter}(y, z)) \\
 & & \mathit{else}\ \mathrm{Inter}(y, z)
\end{array}$$

where In is a membership operator defined by rewrite rules

Example:

$$\begin{array}{rcl}
\mathrm{Union}(\mathit{cons}(0, \mathrm{nil}), \mathit{cons}(0, s(0))) & \to & \mathit{if}\ \mathrm{In}(0, \mathit{cons}(0, s(0))) \\
 & & \mathit{then}\ \mathrm{Union}(\mathrm{nil}, \mathit{cons}(0, s(0))) \\
 & & \mathit{else}\ \mathit{cons}(0, \mathrm{Union}(\mathrm{nil}, \mathit{cons}(0, s(0)))) \\
 & \to^* & \mathrm{Union}(\mathrm{nil}, \mathit{cons}(0, s(0))) \\
 & \to & \mathit{cons}(0, s(0))
\end{array}$$

# $\mathcal{M}$: Operational Semantics

Rewrite-based specification of the axiom (a1):

$$(a2) \quad \mathrm{par}(P, A, R) \quad \rightarrow \quad \textit{if } (A, R) \in \mathrm{arca}^*(\mathrm{contain}(\mathrm{pca}(P))) \\ \textit{then } \mathrm{grant} \textit{ else } \mathrm{deny}$$

grant and deny are answers
pca returns the list of categories assigned to a principal
contain computes the set of categories that contain any of the categories given in the list $\mathrm{pca}(P)$
$\in$ is a membership operator on lists
arca returns the list of all the permissions assigned to the categories in a set

# Evaluating access requests

An access request by a principal $p$ to perform the action $a$ on the resource $r$ is evaluated simply by rewriting $par(p, a, r)$ to normal form.

Proposition:
The rewrite-based definition of $\mathcal{PAR}$ is a correct realisation of the axiom (a1):
$\mathrm{par}(p, a, r) \rightarrow^* \mathrm{grant}$ if and only if $(p, a, r) \in \mathcal{PAR}$

# DTRS: distributed rewriting

*DTRSs* are term rewriting systems where rules are partitioned into modules (associated to sites). Each module has a unique identifier and function symbols are annotated with module identifiers.

$f_\nu$ indicates that the definition of $f$ is in the site $\nu$.

If a symbol $f$ is used without a site annotation, we assume the function is local.

# Example policy

Employees in a company are classified as managers, senior managers or senior executives.

To be categorised as a senior executive (*SeniorExec*), a principal must be a senior manager (*SeniorMng*) according to the information in site $\nu_1$ and must be a member of the executive board.

Any senior executive is permitted to read the salary of an employee, provided the employee works in a profitable branch and is categorised as a Manager (*Manager*).

All managers' names are recorded locally, and the list of profitable branches is kept up to date at site $\nu_2$ .

# Example policy in $\mathcal{M}$

We add to the generic rules:

$$
\begin{array}{lcl}
\mathsf{pca}(P) & \rightarrow & \mathit{if}\ \mathsf{SeniorMng} \in \mathsf{pca}_{\upsilon 1}(P) \\
 & & \mathit{then}\ (\mathit{if}\ \ P \in \mathsf{ExecBoard}\ \mathit{then}\ [\mathsf{SeniorExec}] \\
 & & \qquad\quad \mathit{else}\ [\mathsf{SeniorMng}]) \\
 & & \mathit{else}\ [\mathsf{Manager}] \\
\mathsf{arca}(\mathsf{SeniorExec}) & \rightarrow & \mathsf{zip\text{-}read}(\mathsf{managers}(\mathsf{profbranch}_{\upsilon 2})
\end{array}
$$

zip-read, given a list $L = [l_1, \ldots, l_n]$, returns a list of pairs $[(\mathsf{read}, l_1), \ldots, (\mathsf{read}, l_n)]$

profbranch, defined at site $\upsilon_2$, returns the list of branches that are profitable

manager returns the name of the manager of a branch $B$ given as a parameter (managers does the same for a list of branches).

# Evaluating access requests

$$
\begin{array}{rl}
 & \mathrm{par}(\mathit{Smith}, \mathrm{read}, \mathit{TomSalary}) \\
\to & \mathit{if}\ (\mathrm{read}, \mathit{TomSalary}) \in \mathrm{arca}(\mathrm{pca}(\mathit{Smith})) \\
 & \mathit{then}\ \mathrm{grant}\ \ \mathit{else}\ \mathrm{deny} \\
\to & \mathit{if}\ (\mathrm{read}, \mathit{TomSalary}) \in \mathrm{arca}(\mathit{SeniorExec}) \\
 & \mathit{then}\ \mathrm{grant}\ \ \mathit{else}\ \mathrm{deny} \\
\to & \mathit{if}\ (\mathrm{read}, \mathit{TomSalary}) \in [(\mathrm{read}, \mathit{GreenFile}), \ldots, (\mathrm{read}, \mathit{TomSalary}) \\
 & \mathit{then}\ \mathrm{grant}\ \ \mathit{else}\ \mathrm{deny} \\
\to & \mathit{grant}
\end{array}
$$

assuming

$$
\begin{array}{lcl}
\mathit{ExecBoard} & \to & [\mathit{Taylor}\ ,\ \mathit{Smith},\ \mathit{Clarke}] \\
\mathit{profbranch}_{\nu_2} & \to & [\mathit{Strand}, \mathit{Union}] \\
\mathit{manager(Strand)} & \to & [\mathit{Tom}]
\end{array}
$$

# Properties

Totality: Each request from a valid principal $p$ to perform a valid action $a$ on a resource $r$ receives as answer.

Consistency: For any $p \in \mathcal{P}$, $a \in \mathcal{A}$, $r \in \mathcal{R}$, at most one result is possible for a request $\mathrm{par}(p, a, r)$.

Soundness and Completeness: For any $p \in \mathcal{P}$, $a \in \mathcal{A}$, $r \in \mathcal{R}$, an access request by $p$ to perform the action $a$ on $r$ is granted if and only if $p$ belongs to a category that has the permission $(a, r)$.

# Properties

Totality and consistency can proved by checking:

- confluence — results are unique.
- termination — all requests produce a result

There are several results that provide sufficient conditions for these properties to hold.
[Muller92,KlopOomstRaams93,Breazu-Tannen]
[Gallier,Bakel,Barbanera,Fernandez,Blanqui,Jouannaud,Okada,...]

# Application

The policy in the company example is consistent and total.

It is also sound and complete: if $p$ is a SeniorExec and $m$ is the manager of a profitable branch $b$, then a request from $p$ to read $m$'s file will be granted.

Proof: using properties of the rewrite system defining the policy (ortogonality, hierarchical union).

# Expressiveness

A range of existing access control models can be represented as specialised instances of $\mathcal{M}$ [ESSOS 2010]:

- (static) access control models: DAC, MAC (including Bell-LaPadula),
- RBAC (including time and location constraints)
- Chinese Wall
- dynamic models: DEBAC

# Distributed Category-Based Metamodel

$\mathcal{S}$: identifiers for sites (locations)

Families of relations $\mathcal{PCA}_s$, $\mathcal{ARCA}_s$ and $\mathcal{PAR}_s$, and in addition $\mathcal{BARCA}_s$ (banned actions) and $\mathcal{BAR}_s$ (non-authorised access).
Also a relation $\mathcal{UNDET}_s$, if $\mathcal{PAR}_s$ and $\mathcal{BAR}_s$ are not complete, i.e., some access requests are neither authorised nor denied (undeterminate answer).

$\mathcal{PAR}$ defines the global authorisation policy as a composition of the local policies defined by $\mathcal{PAR}_s$ and $\mathcal{BAR}_s$ (conflict resolution).

# Distributed Axioms

$$
\begin{array}{ll}
(b1) & \forall p \in \mathcal{P},\ \forall a \in \mathcal{A},\ \forall r \in \mathcal{R},\ \forall c \in \mathcal{C},\ \forall s \in \mathcal{S} \\
& (p,c) \in \mathcal{PCA}_s \wedge (\exists c' \in \mathcal{C}, c \subseteq c'\ \wedge (a,r,c') \in \mathcal{ARCA}_s) \\
& \Rightarrow (p,a,r) \in \mathcal{PAR}_s \\
(c1) & \forall p \in \mathcal{P},\ \forall a \in \mathcal{A},\ \forall r \in \mathcal{R},\ \forall c \in \mathcal{C},\ \forall s \in \mathcal{S} \\
& (p,c) \in \mathcal{PCA}_s \wedge (\exists c' \in \mathcal{C}, c \subseteq c' \wedge (a,r,c') \in \mathcal{BARCA}_s) \\
& \Rightarrow (p,a,r) \in \mathcal{BAR}_s \\
(d1) & \forall p \in \mathcal{P},\ \forall a \in \mathcal{A},\ \forall r \in \mathcal{R},\ \forall c \in \mathcal{C},\ \forall s \in \mathcal{S} \\
& (p,c) \in \mathcal{PCA}_s \wedge (a,r,c) \notin \mathcal{ARCA}_s \wedge (a,r,c) \notin \mathcal{BARCA}_s \\
& \Rightarrow (p,a,r) \in \mathcal{UNDET}_s \\
(e1) & \forall s \in \mathcal{S},\quad \mathcal{ARCA}_s \cap \mathcal{BARCA}_s = \emptyset \\
(f1) & \forall p \in \mathcal{P},\ \forall a \in \mathcal{A},\ \forall r \in \mathcal{R}, \\
& (p,a,r) \in \mathcal{OP}_{par}\{\mathcal{PAR}_s, \mathcal{BAR}_s | s \in S\} \Rightarrow (p,a,r) \in \mathcal{PAR} \\
(g1) & \forall p \in \mathcal{P},\ \forall a \in \mathcal{A},\ \forall r \in \mathcal{R}, \\
& (p,a,r) \in \mathcal{OP}_{bar}\{\mathcal{PAR}_s, \mathcal{BAR}_s | s \in S\} \Rightarrow (p,a,r) \in \mathcal{BAR} \\
(h1) & \mathcal{PAR} \cap \mathcal{BAR} = \emptyset
\end{array}
$$

# Combining policies

$\mathcal{UNDET}_s \subseteq \mathcal{P} \times \mathcal{A} \times \mathcal{R}$:
$(p, a, r) \in \mathcal{UNDET}_s$ iff the action $a \in \mathcal{A}$ on resource $r \in \mathcal{R}$ is neither allowed nor forbidden for the principal $p \in \mathcal{P}$ at site $s \in \mathcal{S}$.
The final authorisation is computed by specialising the definition of the operators $\mathcal{OP}_{par}$ and $\mathcal{OP}_{bar}$ (application dependent)

Example: in a system with two sites $s, t \in \mathcal{S}$,
$\mathcal{OP}_{bar} = (\mathcal{BAR}_s \vee \mathcal{BAR}_t)$
$\mathcal{OP}_{par} = ((\mathcal{PAR}_s / \mathcal{BAR}_t) \vee (\mathcal{PAR}_t / \mathcal{BAR}_s))$
corresponds to a union operator giving priority to deny.

# Operational semantics

$$
\begin{array}{lll}
(b2) & \mathrm{par}_s(P, A, R) & \rightarrow \quad \mathit{if} \\
& & (A, R) \in \mathrm{arca}_s^*(\mathrm{contain}(\mathrm{pca}_s(P)) \\
& & \mathit{then}\ \mathrm{grant}\ \mathit{else}\ \mathrm{deny} \\
(c2, d2) & \mathrm{par}_s(P, A, R) & \rightarrow \quad \mathit{if} \\
& & (A, R) \in \mathrm{arca}_s^*(\mathrm{contain}(\mathrm{pca}_s(P)) \\
& & \mathit{then}\ \mathrm{grant}\ \mathit{else} \\
& & \qquad \mathit{if} \\
& & (A, R) \in \mathrm{barca}_s^*(\mathrm{contain}(\mathrm{pca}_s(P \\
& & \mathit{then}\ \mathrm{deny}\ \mathit{else}\ \mathrm{undet} \\
(f2, g2) & Auth(P, A, R, s_1, \ldots, s_n) & \rightarrow \\
& \multicolumn{2}{r}{\mathrm{fauth}(op, \mathrm{par}_{s_1}(P, A, R), \ldots, \mathrm{par}_{s_n}(P, A, R))}
\end{array}
$$

# Conclusion

- Global policies easily built as a combination of local policies using rewrite systems.
- Different combinations can be expressed
- Properties of the policies, such as totality and consistency, follow from modularity results of rewriting.
- Executable specifications, using rewrite-based programming languages.
- Several case studies: virtual museum, bank, hospital
- Implementation methods, programming language design