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
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.
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
- ...
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
- ...
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
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
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.
M: relationships between entities

- Principal-category assignment $\mathcal{PCA}$:
  \[(p, c) \in \mathcal{PCA} \text{ iff } p \in \mathcal{P} \text{ is assigned to } c \in \mathcal{C}\]

- Permissions $\mathcal{ARCA}$:
  \[(a, r, c) \in \mathcal{ARCA} \text{ iff action } a \in \mathcal{A} \text{ on resource } r \in \mathcal{R} \text{ may be performed by principals in the category } c \in \mathcal{C}\]

- Authorisations $\mathcal{PAR}$:
  \[(p, a, r) \in \mathcal{PAR} \text{ iff } p \in \mathcal{P} \text{ may perform action } a \in \mathcal{A} \text{ on resource } r \in \mathcal{R}\]

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

\[
(a_1) \quad \forall p \in P, \forall a \in A, \forall r \in R, \forall c \in C, \\
(p, c) \in PCA \land (\exists c' \in C, c \subseteq c' \land (a, r, c') \in ARCA) \\
\implies (p, a, r) \in PAR
\]

where \( \subseteq \) is a relationship between categories, e.g. equality, set inclusion, \ldots

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 $\mathcal{X}$. $\text{Var}(t)$ denotes the set of variables occurring in $t$.

Rewrite rules: $R = \{l_i \rightarrow r_i\}_{i \in I}$, where $l_i, r_i$ are terms, $l_i \not\in \mathcal{X}$, and $\text{Var}(r_i) \subseteq \text{Var}(l_i)$.

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

Irreducible terms are in normal form.
Term Rewriting: Example

- Natural numbers: 0, s(0), s(s(0)), ...
- Booleans: True, False
- Lists of numbers: nil, cons(0, nil), cons(s(0), nil), ...

- Conditional:
  
  $$\text{if-then-else}(\text{True}, X, Y) \rightarrow X$$

  $$\text{if-then-else}(\text{False}, X, Y) \rightarrow Y$$
Term Rewriting: Example

Operators on sets represented as lists:

\[
\begin{align*}
\text{Union}(\text{nil}, x) & \rightarrow x \\
\text{Union}(\text{cons}(x, y), z) & \rightarrow \text{if In}(x, z) \text{ then } \text{Union}(y, z) \\
& \text{ else cons}(x, \text{Union}(y, z)) \\
\text{Inter}(\text{nil}, x) & \rightarrow \text{nil} \\
\text{Inter}(\text{cons}(x, y), z) & \rightarrow \text{if In}(x, z) \text{ then cons}(x, \text{Inter}(y, z)) \\
& \text{ else Inter}(y, z)
\end{align*}
\]

where In is a membership operator defined by rewrite rules

Example:

\[
\begin{align*}
\text{Union}(\text{cons}(0, \text{nil}), \text{cons}(0, s(0))) & \rightarrow \text{if In}(0, \text{cons}(0, s(0))) \\
& \text{ then Union}(\text{nil}, \text{cons}(0, s(0))) \\
& \text{ else cons}(0, \text{Union}(\text{nil}, \text{cons}(0, s(0)))) \\
& \rightarrow^* \text{Union}(\text{nil}, \text{cons}(0, s(0))) \\
& \rightarrow \text{cons}(0, s(0))
\end{align*}
\]
Rewrite-based specification of the axiom (a1):

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

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 pca(P)
\(\in\) is a membership operator on lists
arca returns the list of all the permissions assigned to the
categories in a set
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):
$par(p, a, r) \rightarrow^* grant$ if and only if $(p, a, r) \in \mathcal{PAR}$
**DTRRS** 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.
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 $\nu_2$. 
We add to the generic rules:

\[
\text{pca}(P) \rightarrow \begin{cases} 
\text{if SeniorMng} \in \text{pca}_{\nu_1}(P) \\
\text{then (if } P \in \text{ExecBoard then } [\text{SeniorExec}] \\
\text{else } [\text{SeniorMng}] \\
\text{else } [\text{Manager}] 
\end{cases}
\]

\[
\text{arca}(\text{SeniorExec}) \rightarrow \text{zip-read}(\text{managers}(\text{profbranch}_{\nu_2}))
\]

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

\text{profbranch, defined at site } \nu_2, \text{ returns the list of branches that are profitable}

\text{manager returns the name of the manager of a branch } B \text{ given as a parameter (managers does the same for a list of branches).}
Evaluating access requests

\[
\begin{align*}
\text{par}(Smith, \text{read}, \text{TomSalary}) & \rightarrow \text{if } (\text{read, TomSalary}) \in \text{arca}(	ext{pca}(Smith)) \\
& \text{then grant } \text{else deny} \\
\text{par}(Smith, \text{read}, \text{TomSalary}) & \rightarrow \text{if } (\text{read, TomSalary}) \in \text{arca}(\text{SeniorExec}) \\
& \text{then grant } \text{else deny} \\
\text{par}(Smith, \text{read}, \text{TomSalary}) & \rightarrow \text{if } (\text{read, TomSalary}) \in [(\text{read, GreenFile}), \ldots, (\text{read, TomSalary})] \\
& \text{then grant } \text{else deny} \\
\text{par}(Smith, \text{read}, \text{TomSalary}) & \rightarrow \text{grant}
\end{align*}
\]

assuming

\[
\begin{align*}
\text{ExecBoard} & \rightarrow [\text{Taylor, Smith, Clarke}] \\
\text{profbranch}_{\nu_2} & \rightarrow [\text{Strand, Union}] \\
\text{manager}(\text{Strand}) & \rightarrow [\text{Tom}]
\end{align*}
\]
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 $\text{par}(p, a, r)$.

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 be 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, …]
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
$S$: 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 $\mathcal{UNDETS}$, 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

\[(b1)\] \(\forall p \in \mathcal{P}, \forall a \in \mathcal{A}, \forall r \in \mathcal{R}, \forall c \in \mathcal{C}, \forall s \in \mathcal{S} \)
\[(p, c) \in \mathcal{PCA}_s \land (\exists c' \in \mathcal{C}, c \subseteq c' \land (a, r, c') \in \mathcal{ARCA}_s) \]
\[\Rightarrow (p, a, r) \in \mathcal{PAR}_s\]

\[(c1)\] \(\forall p \in \mathcal{P}, \forall a \in \mathcal{A}, \forall r \in \mathcal{R}, \forall c \in \mathcal{C}, \forall s \in \mathcal{S} \)
\[(p, c) \in \mathcal{PCA}_s \land (\exists c' \in \mathcal{C}, c \subseteq c' \land (a, r, c') \in \mathcal{BARCA}_s) \]
\[\Rightarrow (p, a, r) \in \mathcal{BAR}_s\]

\[(d1)\] \(\forall p \in \mathcal{P}, \forall a \in \mathcal{A}, \forall r \in \mathcal{R}, \forall c \in \mathcal{C}, \forall s \in \mathcal{S} \)
\[(p, c) \in \mathcal{PCA}_s \land (a, r, c) \notin \mathcal{ARCA}_s \land (a, r, c) \notin \mathcal{BARCA}_s \]
\[\Rightarrow (p, a, r) \in \mathcal{UNDET}_s\]

\[(e1)\] \(\forall s \in \mathcal{S}, \mathcal{ARCA}_s \cap \mathcal{BARCA}_s = \emptyset\)

\[(f1)\] \(\forall p \in \mathcal{P}, \forall a \in \mathcal{A}, \forall r \in \mathcal{R}, \)
\[(p, a, r) \in \mathcal{OP}_\text{par}\{\mathcal{PAR}_s, \mathcal{BAR}_s \mid s \in \mathcal{S}\} \Rightarrow (p, a, r) \in \mathcal{PAR}\]

\[(g1)\] \(\forall p \in \mathcal{P}, \forall a \in \mathcal{A}, \forall r \in \mathcal{R}, \)
\[(p, a, r) \in \mathcal{OP}_\text{bar}\{\mathcal{PAR}_s, \mathcal{BAR}_s \mid s \in \mathcal{S}\} \Rightarrow (p, a, r) \in \mathcal{BAR}\]

\[(h1)\] \(\mathcal{PAR} \cap \mathcal{BAR} = \emptyset\)
Combining policies

\( \text{UNDET}_s \subseteq P \times A \times R: \)

\((p, a, r) \in \text{UNDET}_s \) iff the action \( a \in A \) on resource \( r \in R \) is neither allowed nor forbidden for the principal \( p \in P \) at site \( s \in S \).

The final authorisation is computed by specialising the definition of the operators \( OP_{par} \) and \( OP_{bar} \) (application dependent)

Example: in a system with two sites \( s, t \in S \),

\( OP_{bar} = (\text{BAR}_s \lor \text{BAR}_t) \)

\( OP_{par} = ((\text{PAR}_s/\text{BAR}_t) \lor (\text{PAR}_t/\text{BAR}_s)) \)

corresponds to a union operator giving priority to deny.
Operational semantics

\[(b2) \quad \text{par}_s(P, A, R) \quad \rightarrow \quad \text{if} \]
\[(A, R) \in \text{arca}_s^*(\text{contain} (\text{pca}_s(P))) \]
\[\text{then grant else deny} \]

\[(c2, d2) \quad \text{par}_s(P, A, R) \quad \rightarrow \quad \text{if} \]
\[(A, R) \in \text{arca}_s^*(\text{contain} (\text{pca}_s(P))) \]
\[\text{then grant else} \]
\[\text{if} \]
\[(A, R) \in \text{barca}_s^*(\text{contain} (\text{pca}_s(P))) \]
\[\text{then deny else undet} \]

\[(f2, g2) \quad \text{Auth}(P, A, R, s_1, \ldots, s_n) \quad \rightarrow \quad \text{fauth}(\text{op}, \text{par}_{s_1}(P, A, R), \ldots, \text{par}_{s_n}(P, A, R)) \]
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.