Automated Analysis of Rule-Based Access Control Policies

Clara Bertolissi  Worachet Uttha

LIF, CNRS UMR 7279 & Aix-Marseille University
clara.bertolissi@lif.univ-mrs.fr, worachet.uttha@lif.univ-mrs.fr

Programming Languages meets Program Verification (PLPV)
January 22, 2013
Outline

1. Introduction
2. Category-Based Access Control Meta-model (CMM)
3. Term rewriting system
4. Specification of access control policy
5. Automated analysis
6. Conclusion and Future work
How to share information and retain control over it?
⇒ Access Control Policy

How to model the policy?
⇒ MAC, DAC, RBAC, etc.

How to formalize access control?
⇒ First-order logic, Term Rewriting system, Purpose-built Logics, etc.

How to prove properties of access control?
⇒ First-order Theorem proving, Term Rewriting technique, Model-checking, etc.
Our approach

- Modeling Access Control Policy using Category-based Meta-Model (CMM)\(^1\)
- Specifying Access Control Policy using a rewrite-based framework via a Java interface
- Verifying properties of policy using CiME (rewriting tool)

---

Expressiveness of the meta-model

A range of existing access control models can be represented as specialized instances of the meta-model CMM:

- (static) access control models, such as DAC and MAC (including the well-known Bell-LaPadula model),
- RBAC (including temporal and location constraints),
- the Chinese Wall policy,
- as well as dynamic models, such as DEBAC.
The CiME Tool

- Project at LRI, supported by STIC department of the CNRS
- http://cime.lri.fr/
- Performs checking of rewriting properties such as local confluence and termination
- Able to produce checkable traces of rewriting properties
- Obtained results are close to natural language
Category-Based Access Control Meta-model

Features of the AC meta-model

Entities are denoted by constants in a many sorted domain of discourse, including:

- A countable set $\mathcal{C}$ of *categories*, denoted $c_0, c_1, \ldots$
- A countable set $\mathcal{P}$ of *principals*, denoted $p_0, p_1, \ldots$
- A countable set $\mathcal{A}$ of *named actions*, denoted $a_0, a_1, \ldots$
- A countable set $\mathcal{R}$ of *resource identifiers*, denoted $r_0, \ldots$
- A finite set $\mathcal{Auth}$ of possible *answers* to access requests.

Note

Entities are assigned to distinct classes or groups called categories.
### Relationships between entities (1)

- **Principal-category assignment**: \( \mathcal{PCA} \subseteq \mathcal{P} \times \mathcal{C} \), such that \((p, c) \in \mathcal{PCA}\) iff a principal \(p \in \mathcal{P}\) is assigned to the category \(c \in \mathcal{C}\).

- **Permissions**: \( \mathcal{ARCA} \subseteq \mathcal{A} \times \mathcal{R} \times \mathcal{C} \), such that \((a, r, c) \in \mathcal{ARCA}\) iff the action \(a \in \mathcal{A}\) on resource \(r \in \mathcal{R}\) can be performed by principals assigned to the category \(c \in \mathcal{C}\).

- **Authorizations**: \( \mathcal{PAR} \subseteq \mathcal{P} \times \mathcal{A} \times \mathcal{R} \), such that \((p, a, r) \in \mathcal{PAR}\) iff a principal \(p \in \mathcal{P}\) can perform the action \(a \in \mathcal{A}\) on the resource \(r \in \mathcal{R}\).

Thus, \( \mathcal{PAR} \) defines the set of authorizations that hold according to an access control policy that specifies \( \mathcal{PCA} \) and \( \mathcal{ARCA} \).
**Relationships between entities (2)**

- **Banned actions on resources:** $\text{BARCA} \subseteq A \times R \times C$, such that $(a, r, c) \in \text{BARCA}$ iff the action $a \in A$ on resource $r \in R$ is forbidden for principals assigned to the category $c \in C$.

- **Banned access:** $\text{BAR} \subseteq P \times A \times R$, such that $(p, a, r) \in \text{BAR}$ iff performing the action $a \in A$ on the resource $r \in R$ is forbidden for the principal $p \in P$.

**Note**

Additionally, a relation $\text{UNDE}T$ could be defined if $\text{PAR}$ and $\text{BAR}$ are not complete.
Axioms

The relations satisfy the following core axioms,
∀p ∈ P, ∀a ∈ A, ∀r ∈ R, ∀c ∈ C:

- (p, c) ∈ PCA ∧ (a, r, c) ∈ ARCA ⇔ (p, a, r) ∈ PAR
- (p, c) ∈ PCA ∧ (a, r, c) ∈ BARCA ⇔ (p, a, r) ∈ BAR
- (p, c) ∈ PCA ∧ (a, r, c) /∈ ARCA ∧ (a, r, c) /∈ BARCA ⇔ (p, a, r) ∈ UNDET
- PAR ∩ BAR = ∅
Term Rewriting

Definition

- $\mathcal{T}(F, X)$ denotes the set of terms built up from a signature $F$ and a set of variables $X$.
- A set of rewrite rules $R = \{l_i \rightarrow r_i\}_{i \in I}$, where $l_i, r_i \in \mathcal{T}(F, X)$, $l_i \notin X$.
- A term $t$ rewrites to a term $u$ at position $p$ with the rule $l \rightarrow r$ and the substitution $\sigma$, written $t \rightarrow u$, if $t|_p = l\sigma$ and $u = t[r\sigma]_p$.
- The multi-step reduction relation $\rightarrow^*$ is the reflexive-transitive closure of $\rightarrow$.
- Irreducible terms are said to be in normal form.
A term rewriting system $R$ is:
- **confluent** if for all terms $t, u, v$: $t \rightarrow^* u$ and $t \rightarrow^* v$ implies $u \rightarrow^* s$ and $v \rightarrow^* s$, for some $s$
- **terminating** if all reduction sequences are finite.
Case study: Banking scenario (1)

Specification of access control policy

Set of permissions associated to each category:

- **Manager**: he/she can consult the account of any client; he/she can consult the loan list and the loan demand list.
- **Banker**: same as the manager, plus the rights or accepting or refusing a loan demand.
- **Clerk**: he/she can consult the account of any client and modify user data.
- **Client**: he/she can consult his/her account;
- **Gold-client**: same as client, plus the right of asking for a loan.
Case study : Banking scenario (2)

Specification of access control policy

Set of prohibitions associated to each category:

- **Manager**: a user is assigned to the category *manager* if he/she has a master’s degree and a work experience of more than five years.

- **Gold-Client**: a user is assigned to the category *gold-client* if he/she is an adult (more than 20 years old) and he/she is not in the blacklist (due to a negative account balance for instance).
Rewrite-based specification

**PCA** as Rewrite rule:

\[
pca(p) \rightarrow \begin{cases} \text{if } \text{age}(p) > 20 \text{ and notBlacklisted}(p) \\ \text{then } [\text{GoldClient}] \text{ else } [\text{Client}] \end{cases}
\]

**ARCA** as Rewrite rule:

\[
arca(\text{manager}) \rightarrow [(\text{consult}, \text{account}), (\text{consult}, \text{loanList}), \text{accept, loan}), (\text{refuse, loan})]
\]

**BARCA** as Rewrite rule:

\[
\text{barca}(\text{manager}) \rightarrow [(\text{accept, loan}), (\text{refuse, loan})]
\]
The rewrite-based specification of the axioms in previous section is given by the rewrite rule:

$$\text{par}(p, a, r) \rightarrow \text{if } (a, r) \in \text{arca}(\text{pca}(p)) \text{ then grant}$$

$$\quad \text{else if } (a, r) \in \text{barca}(\text{pca}(p)) \text{ then deny}$$

$$\quad \text{else undeterminate}$$

where
- the function $\in$ is a membership operator on lists.
- grant, deny and undeterminate are answers,
- pca returns a category assigned to a principal
- arca returns a list of permissions assigned to a category
Access request evaluation

\[
\text{par}(\text{GringoJoe}, \text{consult}, \text{loanList}) \\
\rightarrow \text{if } (\text{consult},\text{loanList}) \in \text{arca(}\text{pca(GringoJoe)}) \\
\text{then grant} \\
\text{else if } (\text{consult},\text{loanList}) \in \text{barca(}\text{pca(GringoJoe)}) \\
\text{then deny else undeterminate} \\
\rightarrow^* \text{if } (\text{consult},\text{loanList}) \in \text{arca(}\text{manager}) \\
\text{then grant else ...} \\
\rightarrow^* \text{if } (\text{consult},\text{loanList}) \in [\ldots, (\text{consult},\text{loanList}),\ldots] \\
\text{then grant else ...} \\
\rightarrow^* \text{grant}
\]
Properties of policy

We are interested in the following properties:

- **Totality**: Each access request from a valid user \( p \in \mathcal{P} \) to perform a valid action \( a \in \mathcal{A} \) on the resource \( r \in \mathcal{R} \) receives an answer (e.g. grant, deny or undeterminate).

- **Consistency**: For any \( p \in \mathcal{P}, \ a \in \mathcal{A}, \ r \in \mathcal{R} \), it is not possible to derive more than one result for a request from \( p \) to access \( r \) for \( a \).

- **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 (resp. denied) if and only if \( p \) belongs to a category that has the permission (resp. prohibition) \((a, r)\).
We use the rewrite-based toolbox CiME3 for automated verification of access control properties.

CiME3 (http://a3pat.ensiie.fr/) performs checking of rewrite properties by discovering with full automation, termination and confluence proofs for term rewriting systems.
Automated analysis

- Proofs of properties do not have to be generated by a security administrator
- Verification is launched via the application interface
- Results are interpreted in access control terms and displayed back to the administrator
Termination verification

```plaintext
let R_trs_variable = variables "p,a,r,c,..." ;
let R_trs_signature =
signature "pca : 1; arca : 1; par : 3;
isblacklist : 1; ..." ;
let R_trs_algebra = algebra R_trs_signature ;
let R_trs =
  trs R_trs_algebra "
    barca(banker) -> cons(modify-data, nil);
    arca(banker) -> ...
    ...
rules definition...
  " ;
termination R_trs;
```
Detecting inconsistency

```plaintext
worachetuttha@ubuntu:~/Memoire_CiME/TRS

balance(fertofrancois) -> s(s(0));

isblacklist(fertofrancois) -> false;

age(gringojoe) -> s(s(s(s(s(0))))));

balance(gringojoe) -> s(0);

isblacklist(gringojoe) -> false;

pca(hertzodupont)
  -> if(ge(age(hertzodupont),s(0)),cons(manager,nil),cons(banker,nil))
  "

The rule [36] arca(manager) -> cons(Modifdata,nil) overlaps the rule
[35] arca(manager) ->
cons(consultcompte,cons(consultprete,cons(Modifdata,nil))) at position (/)
yielding the non-joinable critical pair cons(consultcompte,
  cons(consultprete,
  cons(Modifdata,nil)))=

cons(Modifdata,nil)
  - : bool = false
```
Introduction
Category-Based Access Control Meta-model (CMM)
Term rewriting system
Specification of access control policy
Automated analysis
Conclusion and Future work

Policy analysis
Distributed Access Control

Query evaluation

```
reduction by [947] diplome(hertzodupont) -> s(0) at pos 2.0.1.0.0.0
reduction by [18] ge(s(x),s(y)) -> ge(x,y) at pos 2.0.1.0.0.0
reduction by [17] ge(0,s(y)) -> false at pos 2.0.1.0.0.0
reduction by [948] experience(hertzodupont) -> s(s(s(s(0)))) at pos 2.0.1.0.0.1
reduction by [18] ge(s(x),s(y)) -> ge(x,y) at pos 2.0.1.0.0.1
reduction by [18] ge(s(x),s(y)) -> ge(x,y) at pos 2.0.1.0.0.1
reduction by [18] ge(s(x),s(y)) -> ge(x,y) at pos 2.0.1.0.0.1
reduction by [18] ge(s(x),s(y)) -> ge(x,y) at pos 2.0.1.0.0.1
reduction by [16] ge(x,0) -> true at pos 2.0.1.0.0.1
reduction by [6] and(false,y) -> false at pos 2.0.1.0.0
reduction by [2] if(false,x,y) -> y at pos 2.0.1.0
reduction by [35] barcar(cons(c,l)) -> append(barca(c),barcar(l)) at pos 2.0.1
reduction by [42] barca(banker) -> cons(Modifdata,nil) at pos 2.0.1.0
reduction by [34] barcar(nil) -> nil at pos 2.0.1.1
reduction by [24] append(x,nil) -> x at pos 2.0.1
reduction by [20] mem(x,cons(y,l)) -> ifmem(equal(x,y),x,l) at pos 2.0
reduction by [53] equal(consultcompte,Modifdata) -> false at pos 2.0.0
reduction by [22] ifmem(false,x,l) -> mem(x,l) at pos 2.0
reduction by [19] mem(x,nil) -> false at pos 2.0
reduction by [2] if(false,x,y) -> y at pos 2
reduction by [1] if(true,x,y) -> x at pos []
:- R_trs_0_signature_term = grant
```
Detecting conflicts in permissions

```
worachetuttha@ubuntu: ~/Memoire_CiME/TRS

reduction by [30] ifinter(false,x,l1,l2) -> inter(l1,l2) at pos []
reduction by [28] inter(cons(x,l1),l2) -> ifinter(mem(x,l2),x,l1,l2) at pos []
reduction by [20] mem(x,cons(y,l)) -> ifmem(equal(x,y),x,l) at pos 0
reduction by [94] equal(consultprete,transmoney) -> false at pos 0.0
reduction by [22] ifmem(false,x,l) -> mem(x,l) at pos 0
reduction by [19] mem(x,nil) -> false at pos 0
reduction by [30] ifinter(false,x,l1,l2) -> inter(l1,l2) at pos []
reduction by [27] inter(nil,x) -> nil at pos []

s_result : string = "manager clerk"

isConflict : bool = true

conflictVerification : string = "The conflicts arise in manager clerk"
```
GUI for policy specification and testing

Define Principal-Category Assignment

Type of principal:  
- Employee
- Client

Principal:
- hertzo dupont
- thomas durant
- alertos alice

Category:
- manager

Condition:
- if
  - diploma > bac+3
  - and
  - age > 5

Alternative Category:
- banker

Verification:
- Totality
- Consistency
- Conflict

Access Request Evaluation:
- Access Request
Distributed Access Control

Axiom for distributed system:

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

The functions \( \text{par}_l \) in the site \( l \) is defined as follows:

\[ \text{par}_l(p, a, r) \rightarrow \begin{cases} 
\text{grant} & \text{if} \ (a, r) \in \text{arca}_l(p) \\
\text{deny} & \text{if} \ (a, r) \in \text{barca}_l(p) \\
\text{undeterminate} & \text{else} \end{cases} \]

The functions \( \text{par}_c \) in the site \( c \) is defined as follows:

\[ \text{par}_c(p, a, r) \rightarrow \begin{cases} 
\text{grant} & \text{if} \ (a, r) \in \text{arca}_c(p) \\
\text{deny} & \text{else} \end{cases} \]
Distributed Access Control

For our specific bank scenario, we choose the following definition:

\[
\text{authorise}(p, a, r) \rightarrow \text{combine}_{po}(\text{par}_l(p, a, r), \text{par}_c(p, a, r))
\]

together with the definition of the \(\text{combine}_{po}\) function:

\[
\begin{align*}
\text{combine}_{po}(\text{deny}, x) & \rightarrow \text{deny} \\
\text{combine}_{po}(\text{grant}, x) & \rightarrow \text{grant} \\
\text{combine}_{po}(\text{undeterminate}, x) & \rightarrow x
\end{align*}
\]
Conclusion

- Different definitions of access control model using Meta-model
- User-friendly application
- Automated analysis of access control policies

Future work

- Weaving rewrite-based access control policies
- Use Narrowing techniques