Authorisation and Conflict Resolution for Hierarchical Domains

Giovanni Russello
Changyu Dong
Naranker Dulay
Ponder in a nutshell

- A policy language
- A policy interpreter that:
  - Operates on Managed Objects (MO)
  - Organizes MOs in a hierarchical domain structure
  - Enforces policies defined on MOs
A Domain Structure Example

Root domain representing a Hospital

- H1
  - staff
  - patient
    - ER
      - Operation Theatre 1
        - surgeon
        - nurse
        - doc
        - nurse
          - s_1
          - s_2
          - n_1
          - d_1
          - n_2
      - ER
        - Intensive Care
          - p_1
          - p_2
          - p_3
          - p_4

- managed object
- domain
Ponder Authorisation Policies

Control over the execution of a MO’s actions

\[ \text{auth+/- subject, action, target} \]
\[ \text{when condition} \]
From Domains to Privileges

Authorization Policy
Authorisation Policy Extension

- Traditionally auth policies are used to protect the target side of an action
- A finer grained control can be achieved if the policy enforcement points (PEP) are applied also at the subject side
Authorisation Policy Enforcement

subject

PatientAgent

auth+ PatientAgent → MedService.requestTreat()
when insured(request.InsuranceNo)

MedService

taget

Target-side enforcement
Authorisation Policy Enforcement

Subject: PatientAgent

Target: MedService

Target-side enforcement

return+ when reply.diagnosis=TERMINAL
filter reply.diagnosis:=CONTACT_US
Authorisation Policy Enforcement

Subject-side enforcement

auth+ PatientAgent.requestTreat() → MedService
when certified(MedService, NHS)
Authorisation Policy Enforcement

Subject-side enforcement

\[
\text{return when certified} (\text{reply.GPSignature, NHS})
\]
Conflicts

- **Modal Conflicts**: two or more policies with opposite sign apply to the same (subject, target action)-tuple

- **Application-specific Conflicts**: the entities defined in the policy conflict with external criteria (conflicts of interest, conflicts of duties, conflicts of resource allocation,…)

We focus on Modal Conflicts only!
Modal Conflict Resolution

- Based on domain hierarchy
- Set of rules applied during the execution of operations where:
  - Most specific authorisation policies get priority
  - It is possible to define *final* authorisation policies that overwrite any policies down the domain structure
Conflict Resolution Example

Most specific authorisation policies get priority
Conflict Resolution Example

Most generic *Final* authorization policies get priority
Policy Enforcement – Local Scenario

Subject MO  \[\rightarrow\] interpreter  \[\rightarrow\] Target MO  

node  \[\downarrow\]  policy rep
Policy Enforcement – Remote Scenario
MO Remote Invocation

H1
- staff
- ER
- doc
  - d
  - p proxy
- Authentication Service
- Naming Service
- Comm Service

H2
- ext
- patient
- ER
  - d proxy
  - p
- Authentication Service
- Naming Service
- Comm Service
MO Remote Invocation

[Diagram showing a network of nodes and services with arrows indicating flow and connections.]
Conclusions & Future Work

- Uniform framework that caters for both subject and target protection
- Deterministic resolution of conflicts

Future directions:

- Logging of conflicts and resolution steps for off-line investigation
- Synchronization of policies after domain structure changes
Filter Conflict Resolution Example

Patient record: 
\(<\text{name, age, address, symptoms}>\)
Filter Conflict Resolution Example

A1
return+
researcher → patient.readRec()
filter f1 reply.name:=NULL

A2
return+
doctor → patient.readRec()
filter f2 reply.address:=NULL

f1(<name, age, address, symptoms>)=<NULL, age, address, symptoms >
∧
f2(<name, age, address, symptoms>)=<name, age, NULL, symptoms >

<name, age, address, symptoms > + logging of the output result
Filter Conflict Resolution Example

Patient record: <name, age, address, symptoms>
Terminal Patient record: <name, age, address, symptoms, LE>
Filter Conflict Resolution Example

A1
\[
\text{return+}
\text{doc} \rightarrow \text{patient.readRec()}
\text{filter } f_1 \text{ reply.name:=NULL}
\]

A2
\[
\text{return+}
\text{doctor} \rightarrow \text{terminal.readRec()}
\text{filter } f_2 \text{ reply.LE:=NULL}
\]

\[f_1(<\text{name},\text{age},\text{address},\text{symptoms}>) = <\text{name},\text{age},\text{NULL},\text{symptoms}>\]

\[f_2(<\text{name},\text{age},\text{address},\text{symptoms},\text{LE}>) = <\text{name},\text{age},\text{address},\text{symptoms},\text{NULL}>)\]

\[<\text{name},\text{age},\text{NULL},\text{symptoms},\text{NULL}> + \text{logging of the output result}\]
(d) Currently, we use the subject path for discerning the priority of the policies.

i.e.: most specific policy = A2
Modal Conflict Resolution Rules VI

(d) Currently, we use the subject path for discerning the priority of the policies.

i.e.: most general Final policy = A1
(e) After identifying for each path an applicable policy, resolve the conflict as if policies are defined at the same level (rule (a)).
Demo

- Hospital domain structure
- Basic scenario
- Scenario 1 to 3 with different conflicts
Demo – Basic Scenario

- ALL+
  - personnel
    - nurse
    - ward1
    - Nurse1
  - patient
    - ward1
    - intensive care
    - Patient1

A1, A2, A3
Demo – Conflict Scenario 1

- Nurse1
- Nurse2
- Patient1

Diagram:
- ALL+
- nurse
  - Nurse1
  - ward1
    - Nurse2
    - ward1
    - Patient1
- patient
- intensive care

-p1
+p2
Demo – Conflict Scenario 2
Demo – Conflict Scenario 3

Personnel

Nurse

-p1

+P2

ER

Nurse1

Nurse2

Patient

Intensive care

Patient1

Ward1

Patient2