Specifying a Usage Control System

SACMAT 2023

Ulrich Schöpp, Chuangjie Xu, Amjad Ibrahim, Fathiyeh Faghih, Theo Dimitrakos
Attribute-Based Access Control

Policy standards like XACML/ALFA are sufficient specifications for access control.

ALFA policy

```
policy vm_actions {
    target clause Attributes.object == "Virtual Machine"

    apply firstApplicable

    rule start {
        target clause Attributes.action == "start"
        and Attributes.user.role == "owner"
        condition Attributes.server_free_capacity > 100
        permit
    }

    ...  
```
Attribute-Based Usage Control

Usage control with mutable attributes is not sufficiently covered by existing specifications.

Authorization
- continuous re-evaluation during access/usage

Obligations
- log access
- charge customer
- redact private data
- delete personal information

Conditions
- changing environment attributes
Attribute-Based Usage Control

Example: Passing usage policies as data

<table>
<thead>
<tr>
<th>Name</th>
<th>DOB</th>
<th>Hemoglobin</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Doe</td>
<td>21/2/68</td>
<td>14</td>
<td>John's Policy</td>
</tr>
<tr>
<td>Jane Roe</td>
<td>03/11/03</td>
<td>15.1</td>
<td>Jane's Policy</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

John’s data usage policy

```plaintext
policy data_usage {
    target clause ... 
    apply denyUnlessPermit

    rule pre {
        permit
        on permit {
            obligation { anonymizeData }
        }
    }
}
...```
Attribute-Based Usage Control

We need a **practical specification** for **implementations** of **usage control systems**.

- **Specification**
  Precise specification for policy semantics and usage control systems

- **Analysis**
  Understanding policies by automated analysis of usage control situations

- **Verification**
  Testing and monitoring for usage control
Specifying Usage Control Systems

We define a specification for usage control systems.

UCS+ Usage Control System
[Dimitrakos et al., 2020]

- C++ implementation
- Extension of UCON model
[Park & Sandhu, 2004]

ALFA Policies

```plaintext
policy vm_usage {
    apply firstApplicable

    rule before_use {
        target clause Attributes.ucs.step.pre
        condition Attributes.accept_terms
        permit
    }

    rule during_use {
        target clause Attributes.ucs.step.ongoing
        condition Attributes.accept_terms && Attributes.credits > 0
        permit
    }

    rule after_use {
        target clause Attributes.ucs.step.post
        permit
        on permit { obligation sendInvoice }
    }
}
```
Specification Approach

The **system state** abstractly models a snapshot of mutable data and consequences.

*Attributes*
- user: bob
- resource: X
- today: Mo
- Phase: pre

*Attributes*
- user: alice
- resource: Y
- today: Mo
- Phase: ongoing
Specification Approach

*State transitions* model change over time.

**State $n$**

- **PIP Attributes**
  - today: Mo
  - y: 2

  **Session 1** Attributes
  - user: bob
  - today: Mo
  - Phase
  - ongoing

**State $n + 1$**

- **PIP Attributes:**
  - today: Tu

  **Session 1** Attributes
  - user: bob
  - today: Tu
  - Phase
  - post

- **Session 2** Attributes
  - user: alice
  - today: Tu
  - Phase
  - pre
**Specification Approach**

Individual sessions implement continuous usage control

- Conceptually: reevaluate policy after any attribute change
- Non-trivial with concurrent sessions
Our specification accounts for

- policy decisions and obligations,
- multiple concurrent sessions,
- mutable attributes,
- obligation effects,
- synchronization of sessions and obligations,
- delegation,
- ...

There are several reasonable choices at many points.
Examples

Mutual exclusion [Park & Sandhu, 2004]
Limited number of simultaneous uses.

Visibility of updates
Policy writers may expect to see all updates.

```
policy visibility {
  apply denyOverrides

  rule denyIfUntrusted {
    condition Attributes.trustLevel < 50
    deny
    on deny {
      obligation requestReview
    }
  }
}
...
Implementation in Answer Set Programming (ASP)

**ALFA policy**

```
policy smartOfficePolicy {
    target clause user == "scheduler"
    and device == "heating"
    and inhabited == true

    ...
}
```

**Propositional logic (ASP)**

```
policy(smartOfficePolicy, root, firstApplicable).

match_target(smartOfficePolicy, Session, State) :-
    session_attribute(Session, "user", "scheduler", State),
    session_attribute(Session, "device", "heating", State),
    session_attribute(Session, "inhabited", true, State).

...
```

**Specification of Usage Control**

State 1
- PIP
- Attributes:
  - user: bob
  - today: Tu
  - phase: post

State 2
- PIP
- Attributes:
  - user: bob
  - today: Tu
  - phase: T+

Session 1
- Attributes:

```
session_phase(Session, PhaseNext, T+1) :-
    session_phase(Session, Phase, T),
    phase_next(Session, PhaseNext, T),
    state(T+1).

phase_next(Session, "ongoing", T) :-
    session_phase(Session, "pre", T),
    session_pdp_decision(Session, permit, T),
    start_request(Session, T).
```
Application
Tool for automated analysis of usage control situations

Query about usage control situation

ALFA policies

```plaintext
policy vm_usage {
  apply firstApplicable
  rule before_use {
    target clause ucs.step.pre
    condition accept_terms == "Yes"
    permit
    on permit {
      obligation logAccess
    }
  }
}
```

Analysis tool uca

Findings

**FINDING 1**
- session(request_id)
- PDP decision: permit
- PEP decision: permit
- Phase: pre
- Request: start, try
- Attributes
- Attributes.x = 1
- PDP obligations
- Obligations discharged (in order)
- obligation_options.test.r1.on_perm[i].o
- bilation[0].o
- Evaluation details

**FINDING 2**
- session(request_id)
- PDP decision: indeterminate
- PEP decision: deny
- Phase: ongoing
- Attributes
Many thanks.

Any questions?
Contact

PD Dr. Ulrich Schöpp
Safety and Security

fortiss GmbH
Guerickestr. 25 • 80805 Munich • GERMANY
www.fortiss.org

contact@ulrichschoepp.de