Composite Decentralized Access Control

Petar Tsankov, Srdjan Marinovic, Mohammad Torabi Dashti, David Basin

Institute of Information Security
ETH Zurich
Example: SweGrid

Goal
Provides computational and storage resources to researchers

Access Control Requirements

- A project leader delegates his authority over resources to principals
- A project leader composes the principals' policies (e.g., using permit-override)
Delegation

Multiple principals can issue access rights

Project leader

access rights

Researchers

Project Leader

delegations

Bob

Dave

access rights

Researchers
Multiple principals can issue access rights

Delegation

Decentralized Access Control
Composition

Policy decisions in large-scale systems

- Grant, Deny, Not-applicable, Conflict

Composition operators, e.g.:
- Permit-override
- Deny-override
- Conflict-override
Composition

Policy decisions in large-scale systems

- Grant, Deny, Not-applicable, Conflict

Composite Access Control

Composition operators, e.g.:
- Permit-override
- Deny-override
- Conflict-override
System Model

Subjects

Resources
System Model

Requirements

Principals

control access

Subjects

Resources
# Related Work

| Systems and standards |  
|-----------------------|---
| Formal foundations    |  

## Related Work

<table>
<thead>
<tr>
<th>Delegation</th>
</tr>
</thead>
</table>
| Systems and standards | SecPAL for Grid  
|                    | KeyNote PDP \( (RFC 2704) \) |
| Formal foundations | RT ('01)  
|                      | DKAL ('08)  
|                      | ... |
## Related Work

<table>
<thead>
<tr>
<th>Systems and standards</th>
<th>Delegation</th>
<th>Composition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SecPAL for Grid</td>
<td>XACML v2.0</td>
</tr>
<tr>
<td></td>
<td>KeyNote PDP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(RFC 2704)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Formal foundations</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RT ('01)</td>
<td>PBel ('08)</td>
</tr>
<tr>
<td></td>
<td>DKAL ('08)</td>
<td>D-Algebra ('09)</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>PTaCL ('12)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>
## Related Work

<table>
<thead>
<tr>
<th>Systems and standards</th>
<th>Delegation</th>
<th>Composition</th>
<th>Delegation + Composition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SecPAL for Grid</td>
<td>XACML v2.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>KeyNote PDP</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(RFC 2704)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Formal foundations    |           |             |  |
|                       | RT ('01)  | PBel ('08)  |  |
|                       | DKAL ('08) | D-Algebra ('09) |  |
|                       | ...      | PTaCL ('12) |  |
|                       |         | ...         |  |
## Related Work

<table>
<thead>
<tr>
<th>Systems and standards</th>
<th>Delegation</th>
<th>Composition</th>
<th>Delegation + Composition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Systems and standards</td>
<td>SecPAL for Grid RT ('01) DKAL ('08) ...</td>
<td>XACML v2.0 PBel ('08) D-Algebra ('09) PTaCL ('12) ...</td>
<td>SweGrid XACML v3.0 ('13) WSO2 ID Server BelLog</td>
</tr>
<tr>
<td>Formal foundations</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Delegation**: SecPAL for Grid, KeyNote PDP (RFC 2704)
- **Composition**: XACML v2.0, PBel ('08), D-Algebra ('09), PTaCL ('12)
- **Delegation + Composition**: SweGrid, XACML v3.0 ('13), WSO2 ID Server, BelLog
How to Build Access Control Systems

Specify Policy
- Formal semantics
- Support for delegation
- Support for composition

Verify Policy
- Analysis language
- Decision algorithms

Construct PDP
- Efficient evaluation algorithm
How to Build Access Control Systems

Specify Policy
- Formal semantics
- Support for delegation
- Support for composition

Verify Policy
- Analysis language
- Decision algorithms

Construct PDP
- Efficient evaluation algorithm
Belnap Logic + Datalog = BelLog

Belnap Logic

Truth ordering

Knowledge ordering

t
f

(stratified) Datalog

(Program)

(P) ::= \overrightarrow{r}

(rule)

(r) ::= a \leftarrow \overrightarrow{l}

(literal)

(l) ::= a \mid \neg a

(atom)

(a)
Belnap Logic + Datalog = BelLog

Belnap Logic

(stratified) Datalog

(Program) \( P ::= r^* \)

(rule) \( r ::= a \leftarrow l^* \)

(literal) \( l ::= a \mid \neg a \)

(atom) \( a \)
Belnap Logic + Datalog = BelLog

Belnap Logic

Truth ordering

Knowledge ordering

t

f

T

Belnap Logic

(stratified) Datalog

(Program) $P ::= \vec{r}$

(rule) $r ::= a \leftarrow \vec{l}$

(literal) $l ::= a \mid \neg a$

(atom) $a$
Belnap Logic + Datalog = BelLog

Belnap Logic

Truth ordering

Knowledge ordering

(estratified) Datalog

(Program) \( P ::= r^* \)

(rule) \( r ::= a \leftarrow \overrightarrow{l} \)

(literal) \( l ::= a \mid \neg a \mid \sim a \mid \top \)

(atom) \( a \)

Negation on truth

Negation on knowledge

BelLog

(Program) \( P ::= r^* \)

(rule) \( r ::= a \leftarrow \overrightarrow{l} \)

(literal) \( l ::= a \mid \neg a \mid \sim a \mid \top \)

(atom) \( a \)
Belnap Logic + Datalog = BelLog

**Semantics**

Extend stratified Datalog to four-valued fixed-point semantics

- (Program) $P ::= r$
- (rule) $r ::= a \leftarrow \overline{l}$
- (literal) $l ::= a \mid \neg a \mid \overline{a} \mid T$
- (atom) $a$

$\neg$ Negation on truth

$\sim$ Negation on knowledge
BelLog Examples
BelLog Examples

Transitive delegation

\( pol(Req)@X \leftarrow pol(Req)@Y, \text{delegate}(Y)@X \)
BelLog Examples

**Transitive delegation**

\[ pol(Req)@X \leftarrow pol(Req)@Y, \text{delegate}(Y)@X \]

**Policy targets**

\[ polA(Req) \leftarrow \text{target}(Req) \triangleright polB(Req) \]
BelLog Examples

**Transitive delegation**

\[ \text{pol}(\text{Req})@X \leftarrow \text{pol}(\text{Req})@Y, \text{delegate}(Y)@X \]

**Policy targets**

\[ \text{polA}(\text{Req}) \leftarrow \text{target}(\text{Req}) \uparrow \text{polB}(\text{Req}) \]

**Agreement**

\[ \text{polA}(\text{Req}) \leftarrow \text{polB}(\text{Req}) \oplus \text{polC}(\text{Req}) \]
# BelLog Examples

<table>
<thead>
<tr>
<th>Type</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transitive delegation</td>
<td>$pol(Req)@X \leftarrow pol(Req)@Y, delegate(Y)@X$</td>
</tr>
<tr>
<td>Policy targets</td>
<td>$polA(Req) \leftarrow target(Req) \triangleright polB(Req)$</td>
</tr>
<tr>
<td>Agreement</td>
<td>$polA(Req) \leftarrow polB(Req) \oplus polC(Req)$</td>
</tr>
<tr>
<td>Conflict-override</td>
<td>$polA(Req) \leftarrow polB(Req)[\top \mapsto polC(Req)]$</td>
</tr>
</tbody>
</table>
BelLog Examples

Transitive delegation: \( \text{transitive}(\text{req})@Y, \text{delegate}(Y)@X \)

Policy targets: \( \text{req} \Rightarrow \text{polB}(\text{req}) \)

Agreement: \( \text{req} \oplus \text{polC}(\text{req}) \)

Conflict-override: \( \text{req}[T \mapsto \text{polC}(\text{req})] \)

Other idioms?
How to Build Access Control Systems

Specify Policy
- Formal semantics
- Support for delegation
- Support for composition

Verify Policy
- Analysis language
- Decision algorithms

Construct PDP
- Efficient evaluation algorithm
Policy Analysis

Does the policy meet its requirements?
Policy Analysis

Does the policy meet its requirements?

Policy → Questions → Requirements
Does the policy meet its requirements?
Policy Analysis

Does the policy meet its requirements?

Policy → Questions

Fix

Analyzer

Requirements

Counter-example

Policy checked
Does the policy meet its requirements?

Policy Analysis

How do we write this?
Policy Analysis

Does the policy meet its requirements?

- Policy
- Requirements
- Questions
- Analyzer
- Fix
- Counter-example
- Policy checked

How do we write this?
Decidability?
Complexity?
Analysis Questions

Syntax

(quesiton) \( (c \Rightarrow P_1 \leq P_2) \)

(condition) \( c ::= a(X) = t \mid \exists X. c \mid \neg c \mid \cdots \)

– Is policy \textbf{P2} more permissive than \textbf{P1} for all inputs that satisfy the condition \textbf{c}?
Analysis Questions

Syntax

(queston) \((c \Rightarrow P_1 \leq P_2)\)

(condition) \(c ::= a(X) = t \mid \exists X. c \mid \neg c \mid \cdots\)

- Is policy P2 more permissive than P1 for all inputs that satisfy the condition c?

For a given input:
Analysis Questions

Syntax

(question) \( c \Rightarrow P_1 \leq P_2 \)

(condition) \( c ::= a(X) = t \mid \exists X. c \mid \neg c \mid \cdots \)

- Is policy \textbf{P2} more permissive than \textbf{P1} for all inputs that satisfy the condition \( c \)?

For a given input:

\[ \text{All requests} \]

- Requests granted by \textbf{P1}
- Requests granted by \textbf{P2}
Analysis Questions

Syntax

(question) \( (c \Rightarrow P_1 \preceq P_2) \)

(condition) \( c ::= a(X) = t \mid \exists X. \ c \mid \neg c \mid \cdots \)

– Is policy \( P_2 \) more permissive than \( P_1 \) for all inputs that satisfy the condition \( c \)?

For a given input:

Check for all inputs that satisfy the condition

- Requests granted by \( P_1 \)
- Requests granted by \( P_2 \)
Example: Analysis Question

**Policy**

\[ P \]

**Requirement**

*If the requester is a project leader, then grant access.*
Example: Analysis Question

Policy

\[ P \]

Requirement

*If the requester is a project leader, then grant access.*

Analysis Question

\[ \text{leader}(\text{Sub}) \Rightarrow P \leq \{ \text{pol}(\text{Sub}) \leftarrow t \} \]
Analysis
Analysis

Theorem 1
Policy containment is **undecidable**
Analysis

**Theorem 1**
Policy containment is **undecidable**

**Theorem 2**
Policy containment for **unary-input policies*** is in **CO-NEXP-COMPLETE**

*Unary-input policies
– Example: $pol(\text{Sub}, \text{Obj}) \leftarrow \text{leader}(\text{Sub}), \text{pub}(\text{Obj})$
Analysis

**Theorem 1**
Policy containment is **undecidable**

**Theorem 2**
Policy containment for **unary-input policies**\(^*\) is in \text{CO-NEXP-COMPLETE}

**Theorem 3**
Policy containment for **a finite universe** is in \text{CO-NP-COMPLETE}

\(^*\) **Unary-input policies**

- Example: \(\text{pol}(\text{Sub}, \text{Obj}) \leftarrow \text{leader}(\text{Sub}), \text{pub}(\text{Obj})\)
How to Build Access Control Systems

Specify Policy
- Formal semantics
- Support for delegation
- Support for composition

Verify Policy
- Analysis language
- Decision algorithms

Construct PDP
- Efficient evaluation algorithm
Constructing PDPs

Theorem 4
Policy entailment is in PTIME

Policy Interpreter
http://bellog.org

GitHub
https://github.com/ptsankov/bellog/
Limitations

- Analysis of administrative changes
- Analysis complexity and tool support
- Usability
BelLog Contributions

- A foundation for composite decentralized access control
- Policy analysis framework
- BelLog PDP (www.bellog.org)