Authorization Logics and Applications

Larry Diehl - CS 591 Survey Paper
Outline

1. Logic vs ACM

2. Auth Logic example

3. Logical extensions and extralogical devices used in real-world applications.
Access Control

- Policy
- Mechanism
Policy Components

- Principals/Subjects
- Objects/Resources
- Requests/Actions
- Rights
Complete Mediation

(image from Steve Zdancewic’s slides)
Access Control Matrix

- Popular choice for specifying policies.
- But incomplete.
Access Control Matrix

- Popular choice for specifying policies.
- But incomplete.
- Lacks high-level descriptions of why current permissions are set w.r.t. current system state.
# PSU ACL Example

<table>
<thead>
<tr>
<th>A[s][o]</th>
<th>Food Carts</th>
<th>Linux Lab</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>{eat}</td>
<td>{login}</td>
</tr>
<tr>
<td>Bart</td>
<td>{eat}</td>
<td>{}</td>
</tr>
<tr>
<td>Tom</td>
<td>{eat}</td>
<td>{}</td>
</tr>
</tbody>
</table>
# PSU ACL Example

<table>
<thead>
<tr>
<th>A[s][o]</th>
<th>Food Carts</th>
<th>Linux Lab</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>{eat}</td>
<td>{login}</td>
</tr>
<tr>
<td>Bart</td>
<td>{eat}</td>
<td>{}</td>
</tr>
<tr>
<td>Tom</td>
<td>{eat}</td>
<td>{}</td>
</tr>
</tbody>
</table>

What is the policy?
PSU Policy Example

• All people can access food carts.
• Admitted PSU students registered for a CS course can access the Linux Lab.
PSU Policy as a Logic

\[ \forall k. \text{may}(k, \text{food\_cart}, \text{eat}) \]

\[ \forall k. (\text{student}(k) \land \text{cs\_course}(k)) \rightarrow \text{may}(k, \text{linux\_lab}, \text{login}) \]

\text{student}(\text{alice}) \quad \text{student}(\text{bart})

\text{cs\_course}(\text{alice})
may(alice, linux_lab, login)

∀k. may(k, food_cart, eat)

∀k. (student(k) ∧ cs_course(k)) → may(k, linux_lab, login)

student(alice)  student(bart)

cs_course(alice)
Modal Logic

• Additional “modal” logical operators.

• Truth/meaning of a proposition depends on the particular “mode” it is viewed through.

• e.g. a proposition may be true only at a particular time, or may be true only w.r.t a particular authority.
Says Modality

- Not just another FOL predicate.
- Scopes all statements to the principle’s authority.
- Comes with certain logical inference rules, like everyone “says” anything that is globally provable.
cs_dep says may(alice, linux_lab, login)

univ says \( \forall k. \) may(k, food_cart, eat)

\[
\text{cs_dep says } \forall k. \\
(\text{cs_course}(k) \land \text{univ says student}(k)) \\
\Rightarrow \text{may}(k, \text{linux_lab, login})
\]

univ says student(alice)
univ says student(bart)
cs_dep says cs_course(alice)
Logical Judgments

\[ \text{parameters}(\text{true}) = \text{Proposition} \]
\[ p \land q \Rightarrow q \ \text{true} \]

\[ \text{parameters}(\text{affirms}) = \text{Principal} \times \text{Proposition} \]
\[ \text{univ} \ \text{affirms} \ \text{student}(\text{alice}) \]

\[ \text{K} \ \text{affirms} \ P \equiv \text{K} \ \text{says} \ P \ \text{true} \]
\[ \text{univ} \ \text{affirms} \ \text{student}(\text{alice}) \equiv \text{univ} \ \text{says} \ \text{student}(\text{alice}) \ \text{true} \]
Policy Dimensions

• Distributed authorization (says)
• Delegation of authority
• Sub principles and groups
• Principle authentication + non-repudiation
• Reference monitor performance
• Time and system state
• Resource availability/consumption
Delegating Authority

- Nested uses of `says` can accomplish delegating authority.

\[
\text{cs\_dept \ says} \ \forall k. \\
\text{univ \ says} \ \text{student}(k) \Rightarrow \text{student}(k)
\]
Nexus Authorization Logic

- Dependencies among principles and statements occur when nesting says propositions.
  - e.g. CPU says (OS says process_running(0))
- A sub-principle is one that only ever says things its parent says it does (it is “materialized by” its parent.)
  - e.g. CPU.OS says process_running(0)
- Can represent statements by a principle at various points in time.
  - e.g. CPU.OS.1, CPU.OS.2, etc
Groups of principles can be used to get the mode of the union of modes of each principles.

Intentionally specified.

E.g. \([ k : \text{may}(k, \text{file1}, \text{read}) \] says \text{may}(\text{bob}, \text{file1}, \text{write}) \Rightarrow \text{may}(\text{bob}, \text{file1}, \text{read})\)

Union is deductively closed.
Proof-Carrying Authentication

• Normally what a principle “says” is introduced as *a priori* rules in the logic.

• Can also add a rule with a *premise* that introduces *says* proofs valid over a particular time range, given a verification of a digital signature.

• Moves authenticity and non-repudiation inside TCB.

• Can also be done for permanent rules by not mentioning time.
Stateful Auth Logic

• Typical evidence that a policy holds is checking that what a principle supports entails it.

• Predicates in a logic can always be defined by adding new rules.

• Pragmatic addition of new state predicates, whose premise requires validation by external trusted decision procedure.

• Meta-theoretic proofs that cut and identity still hold, as well as state substitution.
Proof-Carrying File System

- Normally Reference Monitor is presented with a proof to check.
- Instead it takes the more traditional role of verifying a capability.
- Unlike the web with network overhead, in a FS proof checking is too expensive.
- Capabilities are generated offline separately for checked proofs.
- Meta-theoretic semantic access coherence proofs.
- Time and system state parameters are included in capability.
Consumable Credentials

- Linear logic proofs about resource consumption checked locally with respect to a global policy.
- Proof passed to a ratifier, performing an extralogical atomic transaction in a distributed system.
The End

references in accompanying paper