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.

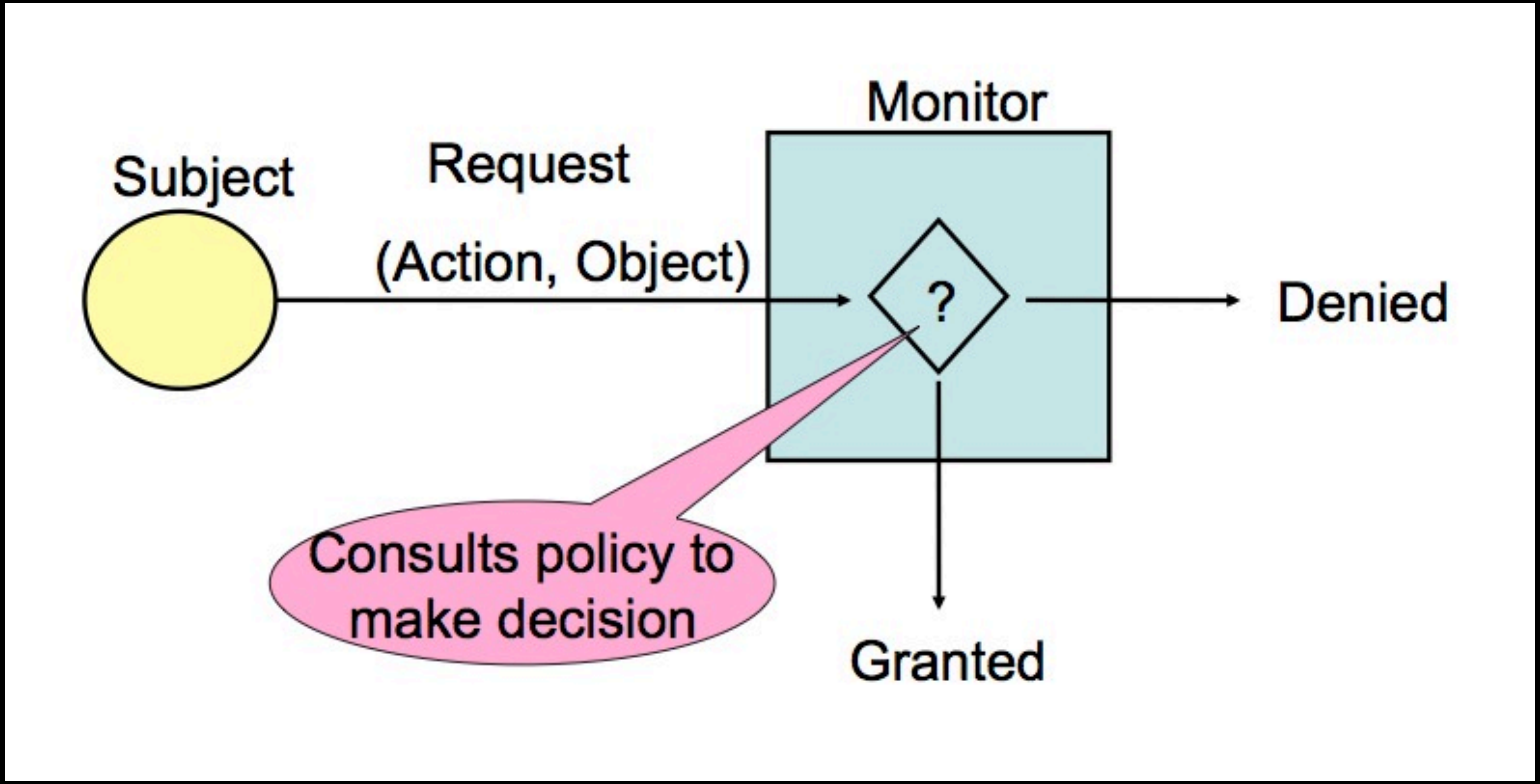
Access Control

- 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 <u>incomplete</u>.

Access Control Matrix

- Popular choice for specifying policies.
- But incomplete.
- Lacks high-level descriptions of <u>why</u> current permissions are set w.r.t. current system <u>state.</u>

PSU ACL Example

A[s][o]	Food Carts	Linux Lab
Alice	{eat}	{login}
Bart	{eat}	{}
Tom	{eat}	{}

PSU ACL Example

A[s][o]	Food Carts	Linux Lab
Alice	{eat}	{login}
Bart	{eat}	{}
Tom	{eat}	{}

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. may(k, food_cart, eat)

$\forall$k. (student(k) $\wedge$ cs_course(k))
$\Rightarrow$ may(k, linux_lab, login)

student(alice)   student(bart)

cs_course(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 <u>says</u> may(alice, linux_lab, login)

univ <u>says</u> ∀k. may(k, food_cart, eat)

cs_dep <u>says</u> ∀k.
(cs_course(k) ∧
univ <u>says</u> student(k))
⇒ may(k, linux_lab, login)

univ <u>says</u> student(alice)

univ <u>says</u> student(bart)

cs_dep <u>says</u> cs_course(alice)

Logical Judgments

parameters(true) = Proposition

$p \wedge q \Rightarrow q$ true

parameters(affirms) = Principal × Proposition

univ affirms student(alice)

K affirms P ≡ K says P true

univ affirms student(alice) ≡ univ says student(alice) 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.

```
cs_dept says ∀k.
univ says student(k) ⇒ student(k)
```

Nexus Authorization Logic

- Dependencies among principles and statements occur when nesting *says* propositions.
  - e.g. CPU <u>says</u> (OS <u>says</u> process_running(0))
- A <u>sub-principle</u> is one that only ever says things its parent says it does (it is "materialized by" its parent.)
  - e.g. CPU.OS <u>says</u> process_running(0)
- Can represent statements by a principle at various points in time.
  - e.g. CPU.OS.1, CPU.OS.2, etc

Nexus Authorization Logic

- Groups of principles can be used to get the mode of the union of modes of each principles.
- Intentionally specified.
    - e.g. [k : may(k, file1, read)] <u>says</u> may(bob, file1, write) ⇒ may(bob, file1, 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 <u>premise</u> 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