Cassandra: Distributed Access Control Policies with Tunable Expressiveness

Moritz Y. Becker and Peter Sewell

Computer Laboratory, University of Cambridge, U.K.
Cassandra: Yet Another PSL?

**Cassandra**
- distributed Trust Management
- rule-based policy specification language (PSL)
- role-based: activation, deactivation, actions
- distributed: credential management

Why YAPSL?
- wide range of applications need tunable expressiveness
- formal semantics: language and dynamics
- distributed query evaluation with guaranteed termination
- practical foundation: real-life case study
Cassandra: Yet Another PSL?

**Cassandra**
- distributed Trust Management
- rule-based policy specification language (PSL)
- role-based: activation, deactivation, actions
- distributed: credential management

**Why YAPSL?**
- wide range of applications → need tunable expressiveness
- formal semantics: language and dynamics
- distributed query evaluation with guaranteed termination
- practical foundation: real-life case study
Cassandra Overview

Cassandra Entity

Access Control Engine

Policy Evaluator

Policy (rules & credentials)

Resources (Actions)

Interface

perform action
activate role
deactivate role
request credential

invoke
modify
grant access

query
remote query
Access Control Semantics (1/2)

- What: specifies dynamic meaning of 4 requests
- Why: makes subtle design decisions explicit
Access Control Semantics (1/2)

- **What**: specifies dynamic meaning of 4 requests
- **Why**: makes subtle design decisions explicit

- can $P$ **perform** action $A$ on $S$’s service?
  - deduce $\text{permits}(P, A)$
Access Control Semantics (1/2)

- What: specifies dynamic meaning of 4 requests
- Why: makes subtle design decisions explicit

- can $P$ **perform** action $A$ on $S$’s service?
  - deduce $\text{permits}(P, A)$

- can $P$ **activate** role $R$ on $S$’s service?
  - deduce $\text{canActivate}(P, R)$
  - add $\text{hasActivated}(P, R)$ to $S$’s policy
Access Control Semantics (2/2)

- can $P$ **deactivate** $V$’s role $R$ on $S$’s service?
  - deduce $\text{canDeactivate}(P, V, R)$
  - under the assumption $\text{isDeactivated}(V, R)$,
    deduce all $\text{isDeactivated}(?, ?)$ on $S$
  - remove all corresponding $\text{hasActivated}(?, ?)$ from $S$’s policy
Access Control Semantics (2/2)

- can $P$ **deactivate** $V$’s role $R$ on $S$’s service?
  - deduce $\text{canDeactivate}(P, V, R)$
  - under the assumption $\text{isDeactivated}(V, R)$, deduce all $\text{isDeactivated}(?, ?)$ on $S$
  - remove all corresponding $\text{hasActivated}(?, ?)$ from $S$’s policy

- can $P$ **request credential** $E_{iss}.p(x) \leftarrow c$ from $S$?
  - deduce $\text{canReqCred}(P, E_{iss}.p(x) \leftarrow c)$ to get $c'$
  - deduce $E_{iss}.p(x) \leftarrow c'$
Policy Specification

- entities control access to their resources with a **Cassandra** policy
- a *policy* is a set of rules based on Datalog
- *rules* are of the form

\[ p_0(\vec{e}_0) \leftarrow loc_1 @ iss_1 . p_1(\vec{e}_1), \ldots, loc_n @ iss_n . p_n(\vec{e}_n), c. \]

(where \( loc_i, iss_i \) are entities and \( c \) is a constraint from the constraint domain)
Policy Specification

- entities control access to their resources with a **Cassandra** policy
- a *policy* is a set of rules based on Datalogₐ
- *rules* are of the form

  \[ p_0(e_0) \leftarrow loc_1@iss_1.p_1(e_1), \ldots, loc_n@iss_n.p_n(e_n), c. \]

  (where \(loc_i, iss_i\) are entities and \(c\) is a constraint from the constraint domain)

- predicates with special access control meaning:
  permits, hasActivated, canActivate, canDeactivate, isDeactivated, canReqCred
Policy Specification

- entities control access to their resources with a **Cassandra** policy
- a *policy* is a set of rules based on Datalog
- *rules* are of the form

\[
p_0(\overline{e}_0) \leftarrow \text{loc}_1@\text{iss}_1.p_1(\overline{e}_1),...,\text{loc}_n@\text{iss}_n.p_n(\overline{e}_n), c.
\]

(where \(\text{loc}_i, \text{iss}_i\) are entities and \(c\) is a constraint from the constraint domain)

- predicates with special access control meaning:
  permits, hasActivated, canActivate, canDeactivate, isDeactivated, canReqCred

- **Example:** suppose a hospital’s policy contains

  \[
  \text{canActivate}(x, \text{Doctor}(spcty)) \leftarrow \\
  x@\text{NHS}.\text{canActivate}(x, \text{CertifiedDoctor}(spcty)), x \neq Alice
  \]
Constraint Domains for Tuning Expressiveness

- $C_{\text{min}}$, The simplest constraint domain:
  
  $$e ::= x \mid E \in \text{Entities}$$
  
  $$c ::= \text{true} \mid \text{false} \mid (e_1 = e_2) \mid c_1 \land c_2 \mid c_1 \lor c_2$$

Constraint domains must support satisifiability checking, projection, and subsumption checking. For guaranteed termination, constraint domains have to be constraint compact.
Constraint Domains for Tuning Expressiveness

- $C_{\text{min}}$, The simplest constraint domain:
  \[
  e \ ::= \ x \mid E \in \text{Entities} \\
  c \ ::= \ \text{true} \mid \text{false} \mid (e_1 = e_2) \mid c_1 \land c_2 \mid c_1 \lor c_2
  \]

- $C_0$, a useful one for complex policies:
  \[
  e \ ::= \ ... \mid n \mid c \mid (e_1,..,e_n) \mid \pi_i(e) \mid f(e) \mid R(e_1,..,e_n) \mid \]
  \[
  A(e_1,..,e_n) \mid \emptyset \mid \Omega \mid \{e_1,..,e_n\} \mid e_1 \cup e_2 \mid e_1 \cap e_2 \mid e_1 \setminus e_2
  \]
  \[
  c \ ::= \ ... \mid e_1 \neq e_2 \mid e_1 < e_2 \mid e_1 \subseteq e_2
  \]
Constraint Domains for Tuning Expressiveness

- $C_{\text{min}}$, The simplest constraint domain:
  
  \begin{align*}
  e & ::= \ x \mid E \in \text{Entities} \\
  c & ::= \text{true} \mid \text{false} \mid (e_1 = e_2) \mid c_1 \land c_2 \mid c_1 \lor c_2
  \end{align*}

- $C_0$, a useful one for complex policies:
  
  \begin{align*}
  e & ::= \ldots \mid n \mid c \mid (e_1,..,e_n) \mid \pi_i(e) \mid f(e) \mid R(e_1,..,e_n) \mid A(e_1,..,e_n) \mid \emptyset \mid \Omega \mid \{e_1,..,e_n\} \mid e_1 \cup e_2 \mid e_1 \cap e_2 \mid e_1 \setminus e_2 \\
  c & ::= \ldots \mid e_1 \neq e_2 \mid e_1 < e_2 \mid e_1 \subseteq e_2
  \end{align*}

- Constraint domains must support
  - satisfiability checking
  - projection
  - subsumption checking

- For guaranteed termination, constraint domains have to be \textit{constraint compact}
Policy Idioms in Cassandra (1/2)

- **appointment**
  
  ```
  canActivate(mgr, AppointEmployee(emp)) ←
  hasActivated(mgr, Manager())
  canActivate(emp, Employee(appointer)) ←
  hasActivated(appointer, AppointEmployee(emp))
  ```

- **appointment revocation**
  
  ```
  isDeactivated(emp, Employee(appointer)) ←
  isDeactivated(appointer, AppointEmployee(emp))
  ```
Policy Idioms in Cassandra (2/2)

- **grant-dependent vs grant-independent** appointment revocation
  
  \[
  \text{canDeactivate}(x, \text{apointer}, \text{AppointEmployee}(\text{emp})) \leftarrow \\
  x = \text{apointer} \\
  \text{canDeactivate}(x, \text{apointer}, \text{AppointEmployee}(\text{emp})) \leftarrow \\
  \text{hasActivated}(x, \text{Manager}())
  \]

- **cascading appointment revocation**
  
  \[
  \text{isDeactivated}(\text{mgr}, \text{AppointEmployee}(\text{emp})) \leftarrow \\
  \text{isDeactivated}(\text{supermgr}, \text{AppointManager}(\text{mgr}))
  \]

- **others**: role hierarchy, role delegation, separation of duties, role validity dates, cardinality/manifold constraints, trust negotiation, ...
National EHR in UK

- NHS planning ICRS with online EHR for clinicians and patients
- Difficulties:
  - huge: 100m records, 400m episodes/yr, 1bn accesses/yr
  - changing requirements
  - distributed policies
  - patient confidentiality requirements
  - access control can be configured by patients/clinicians
- Our three layer approach:
  Master Patient Index (1), EHR servers (100s), health orgs (1000s)
- **Cassandra** policies for all layers: 310 rules, 58 roles, 10 actions
- patient consent, third-party consent, personal AC configuration, legal agents, staff appointment, clinician certification
An Example from the EHR Policy

Prerequisite for Treating-clinician

\[
\text{canActivate}(cli, \text{Treating-clinician}(pat, org, spcty)) \leftarrow \\
org.\text{canActivate}(cli, \text{Group-treating-clinician}(pat, group, spcty)), \\
org@ra.\text{hasActivated}(x, \text{NHS-health-org-cred}(org, start, end)), \\
ra \in \text{NHS-registration-authorities}(), \\
\text{Current-time()} \in [start, end]
\]
Conclusion

- **Cassandra**’s expressiveness is tunable; very expressive with $C_0$.
- High-level enough for concise and readable policies.
- Low-level enough to express wide range of policies.
- Formal foundation.
- Substantial case study.
- Prototype implementation.