Logic and Logic Programming in Distributed Access Control (Part Two)

Ninghui Li
Department of Computer Science
and CERIAS
Purdue University
Security Analysis in Trust Management

Publications:

The Abstract Security Analysis Problem

- Given an initial state $P$,
  - a query $Q$,
  - and a rule $R$ that restricts how states can change (defines reachability among states);

- Ask
  - **Is $Q$ possible?** (existential)
    - whether $\exists$ reachable $P'$ s.t. $P' \triangleright Q$
  - **Is $Q$ necessary?** (universal)
    - whether $\forall$ reachable $P'$, $P' \triangleright Q$
Statements in \( RT_0 = RT[\leftrightarrow, \cap] \)

- **Type-1:** \( K.r \leftarrow K_1 \)
  - \( \text{mem}[K.r] \supseteq \{K_1\} \)
  - \( K_{HR\cdot manager} \leftarrow K_{Alice} \)

- **Type-2:** \( K.r \leftarrow K_1.r_1 \)
  - \( \text{mem}[K.r] \supseteq \text{mem}[K_1.r_1] \)
  - \( K_{SSO\cdot admin} \leftarrow K_{HR\cdot manager} \)
Statements in RT[\(\leftrightarrow, \cap\)]

- **Type-3:** \(K.r \leftarrow K.r_1.r_2\)
  - Let \(\text{mem}[K.r_1]\) be \(\{K_1, K_2, \ldots, K_n\}\)
    
    \[\text{mem}[K.r] \supseteq \text{mem}[K_1.r_2] \cup \text{mem}[K_2.r_2] \cup \ldots \cup \text{mem}[K_n.r_2]\]
  - \(K_{SSO}.\text{delegAccess} \leftarrow K_{SSO}.\text{admin.access}\)

- **Type-4:** \(K.r \leftarrow K_1.r_1 \cap K_2.r_2\)
  - \(\text{mem}[K.r] \supseteq \text{mem}[K_1.r_2] \cap \text{mem}[K_2.r_2]\)
  - \(K_{SSO}.\text{access} \leftarrow K_{SSO}.\text{delegAccess} \cap K_{HR}.\text{employee}\)
The Query Q

- Form-1: \( \text{mem}[K.r] \supseteq \{K_1, \ldots, K_n\} \) ?
- Form-2: \( \{K_1, \ldots, K_n\} \supseteq \text{mem}[K.r] \) ?
- Form-3: \( \text{mem}[K_1.r_1] \supseteq \text{mem}[K.r] \) ?
The Semantic Relation

- A statement ⇒ a Datalog rule
  - $K.r \leftarrow K_2 \quad \Rightarrow \quad m(K, r, K_2)$
  - $K.r \leftarrow K_1.r_1 \quad \Rightarrow \quad m(K, r, z) :\quad m(K_1, r_1, z)$
  - ...

- A state $P \Rightarrow$ a Datalog program $SP[P]$
  - $\text{mem}[K.r] \equiv \{ K' \mid m(K,r,K') \text{ is in the minimal Herbrand model of } SP[P] \}$
Example Queries & Answers

1. \( K_{SSO}.access \leftarrow K_{SSO}.admin \)
2. \( K_{SSO}.admin \leftarrow K_{HR}.manager \)
3. \( K_{HR}.employee \leftarrow K_{HR}.manager \)
4. \( K_{HR}.manager \leftarrow K_{Alice} \)
5. \( K_{HR}.employee \leftarrow K_{David} \)

\[
\text{mem}[K_{SSO}.access] \supseteq \{K_{David}\}? \quad \text{No}
\]
\[
\{K_{Alice}, K_{David}\} \supseteq \text{mem}[K_{SSO}.employee]? \quad \text{Yes}
\]
\[
\text{mem}[K_{HR}.employee] \supseteq \text{mem}[K_{SSO}.access]? \quad \text{Yes}
\]
The Restriction Rule $R$

- $R=(G,S)$
  - $G$ is a set of growth-restricted roles
    - if $K.r \in G$, then cannot add "$K.r \leftarrow \ldots"$
  - $S$ is a set of shrink-restricted roles
    - if $K.r \in S$, then cannot remove "$K.r \leftarrow \ldots"

- Motivation:
  - Definitions of roles that are not under one’s control may change

2nd Int’l Summer School in Computation Logic
June 17, 2004
Ninghui Li (Purdue University)
Sample Analysis Queries

- Simple safety (existential form-1):
  - Is mem[K.r] $\supseteq \{K_1\}$ possible?

- Simple availability (universal form-1):
  - Is mem[K.r] $\supseteq \{K_1\}$ necessary?

- Bounded safety (universal form-2):
  - Is $\{K_1,\ldots,K_n\} \supseteq$ mem[K.r] necessary?

- Containment (universal form-3):
  - Is mem[K_1.r_1] $\supseteq$ mem[K.r] necessary?
Security Analysis: Usage Cases

- Guarantee safety and availability properties of an access control system:
  - Properties one wants to guarantee are encoded in a set of queries & desirable answers
  - $R$ represents how much control one has
    - parts not under one’s control may change in $R$
    - parts under one’s control are considered fixed in $R$
  - Before making changes, one can use analysis to guarantee properties are not violated
An Example

1. \( \text{K}_{\text{SSO}}\text{.access} \leftarrow \text{K}_{\text{SSO}}\text{.admin} \)
2. \( \text{K}_{\text{SSO}}\text{.access} \leftarrow \text{K}_{\text{SSO}}\text{.delegAccess} \cap \text{K}_{\text{HR}}\text{.employee} \)
3. \( \text{K}_{\text{SSO}}\text{.admin} \leftarrow \text{K}_{\text{HR}}\text{.manager} \)
4. \( \text{K}_{\text{SSO}}\text{.delegAccess} \leftarrow \text{K}_{\text{SSO}}\text{.admin.access} \)
5. \( \text{K}_{\text{HR}}\text{.employee} \leftarrow \text{K}_{\text{HR}}\text{.manager} \)
6. \( \text{K}_{\text{HR}}\text{.employee} \leftarrow \text{K}_{\text{HR}}\text{.engineer} \)
7. \( \text{K}_{\text{HR}}\text{.manager} \leftarrow \text{K}_{\text{Alice}} \)
8. \( \text{K}_{\text{Alice}}\text{.access} \leftarrow \text{K}_{\text{Bob}} \)

Legend: fixed
can grow, can shrink
A Simple Availability Query

1. $K_{SSO}.access \leftarrow K_{SSO}.admin$
2. $K_{SSO}.access \leftarrow K_{SSO}.delegAccess \cap K_{HR}.employee$
3. $K_{SSO}.admin \leftarrow K_{HR}.manager$
4. $K_{SSO}.delegAccess \leftarrow K_{SSO}.admin.access$
5. $K_{HR}.employee \leftarrow K_{HR}.manager$
6. $K_{HR}.employee \leftarrow K_{HR}.engineer$
7. $K_{HR}.manager \leftarrow K_{Alice}$
8. $K_{Alice}.access \leftarrow K_{Bob}$

Query: Is $\text{mem}[K_{SSO}.access] \supseteq \{K_{Alice}\}$ necessary?
Answer: Yes. (Available)
Why: Statements 1, 3, and 7 cannot be removed.
A Simple Safety Query

1. \( K_{SSO}\text{-access} \leftarrow K_{SSO}\text{-admin} \)
2. \( K_{SSO}\text{-access} \leftarrow K_{SSO}\text{-delegAccess} \cap K_{HR}\text{-employee} \)
3. \( K_{SSO}\text{-admin} \leftarrow K_{HR}\text{-manager} \)
4. \( K_{SSO}\text{-delegAccess} \leftarrow K_{SSO}\text{-admin}\text{-access} \)
5. \( K_{HR}\text{-employee} \leftarrow K_{HR}\text{-manager} \)
6. \( K_{HR}\text{-manager} \leftarrow K_{Alice} \)
7. \( K_{HR}\text{-employee} \leftarrow K_{HR}\text{-engineer} \)
8. \( K_{Alice}\text{-access} \leftarrow K_{Bob} \)

Query: Is \( \text{mem}[K_{SSO}\text{-access}] \supseteq \{K_{Eve}\} \) possible?

Answer: Yes. (Unsafe)

Why: Both \( K_{HR}\text{-engineer} \) and \( K_{Alice}\text{-access} \) may grow.

2nd Int’l Summer School in Computation Logic
June 17, 2004
Ninghui Li (Purdue University)
A Containment Analysis Query about Safety

1. $K_{SSO}.access \leftarrow K_{SSO}.admin$
2. $K_{SSO}.access \leftarrow K_{SSO}.delegAccess \cap K_{HR}.employee$
3. $K_{SSO}.admin \leftarrow K_{HR}.manager$
4. $K_{SSO}.delegAccess \leftarrow K_{SSO}.admin.access$
5. $K_{HR}.employee \leftarrow K_{HR}.manager$
6. $K_{HR}.employee \leftarrow K_{HR}.engineer$
7. $K_{HR}.manager \leftarrow K_{Alice}$
8. $K_{Alice}.access \leftarrow K_{Bob}$

Query: Is $\text{mem}[K_{HR}.employee] \supseteq \text{mem}[K_{SSO}.access]$ necessary?
Answer: Yes. (Safe)
Why: $K_{SSO}.access$ and $K_{SSO}.admin$ cannot grow and Statement 5 cannot be removed.
An Containment Analysis Query about Availability

1. \( K_{SSO}.access \leftrightarrow K_{SSO}.admin \)
2. \( K_{SSO}.access \leftarrow K_{SSO}.delegAccess \cap K_{HR}.employee \)
3. \( K_{SSO}.admin \leftrightarrow K_{HR}.manager \)
4. \( K_{SSO}.delegAccess \leftarrow K_{SSO}.admin.access \)
5. \( K_{HR}.employee \leftrightarrow K_{HR}.manager \)
6. \( K_{HR}.employee \leftrightarrow K_{HR}.engineer \)
7. \( K_{HR}.manager \leftrightarrow K_{Alice} \)
8. \( K_{Alice}.access \leftrightarrow K_{Bob} \)

Query: Is \( \text{mem}[K_{SSO}.access] \supseteq \text{mem}[K_{HR}.manager] \) necessary?
Answer: Yes. (Available)
Why: Statements 1 and 3 cannot be removed
Answering Form-1 and Form-2 Queries: Intuitions (1)

- RT[$\leftrightarrow$, $\cap$] is monotonic
  - more statements derive more role memberships

- Form-1 queries are monotonic
  - \text{mem}[K.r] \supseteq \{K1,\ldots,Kn\}

- universal form-1 queries can be answered by considering a lower-bound (minimum) state

- existential form-1 queries can be answered by considering an upper-bound (maximal) state
Answering Form-1 and Form-2 Queries: Intuitions (2)

- Form-2 queries are anti-monotonic
  - \( \{K_1, \ldots, K_n\} \supseteq \text{mem}[K.r] \)
  - universal form-2 queries can be answered by considering the upper-bound state
  - existential form-1 queries can be answered by considering the lower-bound state

- Given P and R, the lower-bound state uniquely exists, we denote it \( P|_R \)
  - it can be reached by removing all removable statements
The Lower-Bound Program \( \text{LB}(P, R) \)

- For each \( K.r \leftarrow K_1 \) in \( P|_R \), add
  \[ \text{lb}(K, r, K_1) \]

- For each \( K.r \leftarrow K_1.r_1 \) in \( P|_R \), add
  \[ \text{lb}(K, r, ?Z) :\text{lb}(K_1, r_1, ?Z) \]

- For each \( K.r \leftarrow K.r_1.r_2 \) in \( P|_R \), add
  \[ \text{lb}(K, r, ?Z) :\text{lb}(K, r_1, ?Y), \text{lb}(?Y, r_2, ?Z) \]

- For each \( K.r \leftarrow K_1.r_1 \cap K_2.r_2 \) in \( P|_R \), add
  \[ \text{lb}(K, r, ?Z) :\text{lb}(K_1, r_1, ?Z), \text{lb}(K_2, r_2, ?Z) \]
Using the Lower-Bound Program

- To answer whether a form-1 query $\text{mem}[K.r] \supseteq \{K_1,\ldots, K_n\}$ is necessary,
  - check whether $\text{LB}(P,R) \models \text{lb}(K,r,K_1) \land \ldots \land \text{lb}(K,r,K_n)$

- To answer whether a form-2 query $\{K_1,\ldots,K_n\} \supseteq \text{mem}[K.r]$ is possible
  - check whether $\{K_1,\ldots,K_n\} \supseteq \{Z\mid \text{LB}(P,R) \models \text{lb}(K,r,Z)\}$
The Upper-Bound Program UB(P,R)

- Add \( \text{ub}(T, ?r, ?Z) \)
- For each \( K.r \) that can grow, add \( \text{ub}(K, r, ?Z) \)
- For each \( K.r \leftarrow K_1 \) in \( P \), add \( \text{ub}(K, r, K_1) \)
- For each \( K.r \leftarrow K_1.r_1 \) in \( P \), add
  \[ \text{ub}(K, r, ?Z) \leftarrow \text{ub}(K_1, r_1, ?Z) \]
- For each \( K.r \leftarrow K.r_1.r_2 \) in \( P \), add
  \[ \text{ub}(K, r, ?Z) \leftarrow \text{ub}(K, r_1, ?Y), \text{ub}(?Y, r_2, ?Z) \]
- For each \( K.r \leftarrow K_1.r_1 \cap K_2.r_2 \) in \( P \), add
  \[ \text{ub}(K, r, ?Z) \leftarrow \text{ub}(K_1, r_1, ?Z), \text{ub}(K_2, r_2, ?Z) \]
A form-1 query \( \text{mem}[K.r] \supseteq \{K_1, \ldots, K_n\} \) is possible iff. any of the following is true,
- \( K.r \) is not growth restricted
- \( \text{up}(K,r,T) \) is true
- \( \text{UB}(P,R) |= \text{ub}(K,r,K_1) \land \ldots \land \text{ub}(K,r,K_n) \)

A form-2 query \( \{K_1, \ldots, K_n\} \supseteq \text{mem}[K.r] \) is necessary iff.
- \( \{K_1, \ldots, K_n\} \supseteq \{Z \mid \text{UB}(P,R) |= \text{ub}(K,r,Z)\} \)
What about Form-3 Queries?

- Form-3: \( \text{mem}[K_1.r_1] \supseteq \text{mem}[K.r] \)
- Neither monotonic nor anti-monotonic
  - cannot use the minimal state or the maximal state
- Difficulty: adding new members to \( K.r \) may affect \( K_1.r_1 \)
- We only consider analysis asking whether \( \text{mem}[K_1.r_1] \supseteq \text{mem}[K.r] \) is necessary
  - we call this containment analysis
Complexity Results for Containment Analysis

- **RT[]**: just type 1 and 2 statements
  - Containment analysis is in PTIME
- **RT[∩]**: type 1, 2, and 4 statements
  - Containment analysis is coNP-complete
- **RT[↔]**: type 1, 2, and 3 statements
  - Containment analysis is PSPACE-complete
  - Remains PSPACE-complete without shrinking
  - CoNP-complete without growing
- **RT[↔,∩]**: decidable in coNEXP
Containment Analysis in RT[]

- Two cases that X.u contains K.r
  1. the containment is forced by statements in P and cannot be removed
  2. the containment is caused by nonexistence of statements
     - e.g., when no statement defines K.r and K.r cannot grow, K.r is always empty, and thus is contained in every role
     - direct translation of this intuition into a positive logic program does not work
     - e.g., P={“K.r←K_1.r_1”, “K_1.r_1←K.r”, “K.r←K_2”, “X.u ←K_2”}, both K.r and K_1.r_1 are fixed, does X.u contain K.r?
The Containment Program for RT[]: BCP(P,R)

- Starts from LB(P,R)
- Add \( fc(?X,?u,?X,?u) \)
- For each \( K.r \leftarrow K_1.r_1 \) in \( P|_R \), add
  \[ fc(K,r,?Z,?w) :- fc(K_1,r_1,?Z,?w) \]
- For each \( K.r \) that can grow, add
  \[ nc(?X,?u,K,r) :- \sim fc(?X,?u,K,r) \]
- For each \( K.r \leftarrow K_1 \) in \( P \) s.t. \( K.r \) can’t grow, add
  \[ nc(?X,?u,K,r) :- \sim fc(?X,?u,K,r), \sim lb(?X,?u,K_1) \]
- For each \( K.r \leftarrow K_1.r_1 \) in \( P \) s.t. \( K.r \) can’t grow, add
  \[ nc(?X,?u,K,r) :- \sim fc(?X,?u,K,r), nc(?X,?u,K_1,r_1) \]
Solving Containment Analysis in RT[] Using Negation

- BCP(P,R) is stratified
  - we use the perfect model semantics

- Theorem: BCP(P,R) ├₀ nc(X,u, K,r) is true iff. X.u does not contain K.r
Containment Analysis in RT[∩] is coNP-complete

- It is in coNP, because a counter example can be found by considering just one new principal.
- That it is coNP-hard is shown by reducing the monotone 3-SAT problem to it.
  - intersection is conjunction,
  - a role may be defined by multiple statements (implicit disjunction)
  - containment equivalent to determining validity of formulas like $\varphi_1 \iff \varphi_2$
- where $\varphi_1$ are $\varphi_2$ positive propositional formulas.
First consider the case that no shrinking is allowed in R

View statements as rewriting rules

- \( K.r \leftarrow K_1 \) \( K r \) to \( K_1 \)
- \( K.r \leftarrow K_1.r_1 \) \( K r \) to \( K_1 r_1 \)
- \( K.r \leftarrow K.r_1.r_2 \) \( K r \) to \( K r_1 r_2 \)

A string has the form \( K r_1 r_2 r_3 r_4 \)

Lemma 0: SP[P] proves \( m(K,r,K_1) \) iff. the string \( K r \) rewrites into \( K_1 \) using P
RT[↔] and Pushdown Systems

Stack:  

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>u₁</td>
<td>u₁</td>
</tr>
<tr>
<td>u₂</td>
<td>u₂</td>
</tr>
</tbody>
</table>
| ...| ...

State: K

Apply the rewriting rule: $K r \rightarrow K r_1 r_2$

A string corresponds to a configuration

“rewrites into” equivalent to “reaches”
Characteristic Set of a Role

- Given P and R (shrinking forbidden), define:
  - \( \text{strs}_P[K.r] = \) sets of strings K r rewrites to
  - \( \chi_R = \) the set consisting of
    - all principals in P
    - all strings \( K_1 r_1 r_2 r_3 r_4 \) where \( K_1 \) appears in P and \( K_1 r_1 \) is g-unrestricted
  - \( \chi_{P,R}[K.r] = \text{strs}_P[K.r] \cap \chi_R \)
    - each string \( K_1 r_1 r_2 r_3 r_4 \) in \( \chi_{P,R}[K.r] \) is a distinct way of adding a member to K.r
- Lemma 1: Given P, R, X.u, K.r, mem[X.u] \( \supseteq \) mem[K.r] is necessary iff. \( \chi_{P,R}[X.u] \supseteq \chi_{P,R}[K.r] \)
Lemma 2:

- Lemma 2: Given P, R (shrinking forbidden), and K.r, $\chi_{P,R}[K.r]$ is recognized by an NFA that has size poly in $|P| + |R|$.

- Proof: $\chi_{P,R}[K.r] = \text{strs}_P[K.r] \cap \chi_R$
  - $\text{strs}_P[K.r]$ is recognized by a poly-size NFA
    - Bouajjani, Esparza & Maler: “Reachability Analysis of Pushdown Automata: Application to Model-Checking”, CONCUR’97
  - $\chi_R$ is recognized by a poly-size NFA
  - $\chi_{P,R}[K.r]$ is recognized by a poly-size NFA
Containment Analysis in RT[\leftrightarrow] is in PSPACE

- Theorem: Given P, R (shrinking forbidden), X.u, K.r, determining whether mem[X.u] \supseteq mem[K.r] is necessary is in PSPACE
  - follows from Lemma 1 and 2 and the fact that determining containment of languages accepted by NFA’s is in PSPACE
Containment Analysis in RT[↔] is PSPACE-hard

- Theorem: Given P, R (shrinking forbidden), X.u, K.r, determining whether mem[X.u] ⊇ mem[K.r] is necessary is PSPACE-hard
  - Reducing determining containment of languages over the alphabet {0,1} that are defined by right-linear grammars to the problem.
Proof of PSPACE-hardness

- From grammar to P:
  - $N_1 ::= N_2 1 \quad \text{K.N}_1 = \text{K.N}_2.r_1$
  - $N_2 ::= 0 \quad \text{K.N}_2 = \text{K}_1.r_0$

- The restriction rule R:
  - all K.$N_i$’s, K.$r_i$’s, and K.$1.N_i$’s are g-restricted
  - other roles, i.e., K.$1.r_0$ and K.$1.r_1$, are growth unrestricted

- Language[$N_1$] maps to $\chi_{P,R}[\text{K.N}_1]$
  - $N_1$ generates 1010 iff. K.$1.r_1.r_0.r_1.r_0 \in \chi_{P,R} [\text{K.N}_1]$
Theorem (shrinking allowed)

- Given P, R (shrinking allowed), X.u, K.r, determining whether \( \text{mem}[X.u] \supseteq \text{mem}[K.r] \) is necessary is in PSPACE
  - For every subset of P that can be obtained by legally removing statements in P, run the algorithm that does not allow shrinking
Theorem: Given P (in RT[\(\leftrightarrow \cap\)]), R, X.u, K.r, determining whether mem[X.u] \(\supseteq\) mem[K.r] is necessary is in coNEXP

- although infinitely many new principals and statements may be added, if a counter example exists, then a counter example of size exponential in P exists
- if two new principals have the same memberships in all roles appearing in P, then the two principals can be collapsed into one
Summary of Complexities for Containment Analysis

Type-1 and 2: PTIME

Type-1, 2, and 3: PSPACE-complete

Type-1, 2, and 4: coNP-complete

Type-1, 2, 3, and 4: PSPACE-hard, coNEXP