Access Control Policy Mining: Feasibility Analysis

L10-1
CS6393
Spring 2020
Can we access objects?

Legitimate users get legitimate access only e.g., Role-Based Access Control (RBAC), Attribute-Based Access Control (ABAC)
Problem: migration from an existing access control model to another one

- New access control
- Organization size changes
- Changing mode of operation
- Switch to existing better one

Manual effort often error-prone, time consuming and costly

Is automation possible?

@Shuvra Chakraborty
Introduction

Access Control List / Log / RBAC + Supporting attribute data → ABAC policy mining

Access Control List + Supporting Relationship data → ReBAC policy mining

Given an access control system + Supporting data → Another access control model

General term: Access control policy mining

Mining is partially automated so far...

*** Relationship-Based Access Control (ReBAC)

@Shuvra Chakraborty
The feasibility analysis of the access control mining problem studies whether the migration process is possible or not under the set of imposed criteria.
Developing feasibility analysis algorithms for certain set of access control mining problem with complexity analysis.

In case of infeasibility, solution algorithms are presented to make it feasible under given criteria.
To the best of our knowledge: feasibility analysis of access control policy mining is proposed for the first time. Hence, no directly related background work.

Some access control policy mining works:
- Role Mining
- ABAC policy mining [from authorization, RBAC, log data, sparse log] etc.

Our study includes 3 types of Access Control System:
- Enumerated Authorization System (EAS)
- RBAC System
- ABAC System

@Shuvra Chakraborty
EAS is a tuple $<U, O, OP, AUTH, \text{checkAccess}_{EAS}>$

- $U, O, \text{ and } OP$ are finite sets of users, objects and operations, respectively
- $AUTH \subseteq U \times O \times OP$

**Example 1:**

- $U = \{\text{John, Lina, Ray, Tom}\}$, $OP = \{\text{read, write}\}$, $O = \{\text{Obj1, Obj2}\}$

<table>
<thead>
<tr>
<th>AUTH</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>(John, Obj1, write)</td>
<td>e.g., John is allowed to do read operation on Obj1 but not allowed to do read operation on Obj2</td>
</tr>
<tr>
<td>(John, Obj2, write)</td>
<td></td>
</tr>
<tr>
<td>(John, Obj1, read)</td>
<td></td>
</tr>
<tr>
<td>(Lina, Obj2, write)</td>
<td></td>
</tr>
<tr>
<td>(Tom, Obj1, read)</td>
<td></td>
</tr>
<tr>
<td>(Ray, Obj1, read)</td>
<td></td>
</tr>
</tbody>
</table>
RBAC system

- is a tuple <U, O, OP, Roles, RPA, RUA, RH, checkAccess_{RBAC}>

- Roles is a finite set of roles
- RH is the role hierarchy relation [RH': reflexive transitive closure of RH]
- RPA: Role Permission Assignment
- RUA: Role User Assignment
- Permission is an object-operation pair
- authPerm(r) = \{p \in RPA(r')|(r, r') \in RH'}\}, where r,r' \in Roles
- authUser (r) = \{u \in RUA(r')|(r', r) \in RH'}\} where r,r' \in Roles

- checkAccess_{RBAC}(u:U, o:O, op:OP) ≡ \exists r \in Roles.(u \in authUser(r) \land (o, op) \in authPerm(r)

@Shuvra Chakraborty
**Example 2:**
- $U = \{\text{John, Lina, Ray, Tom}\}$, $OP = \{\text{read, write}\}$, $O = \{\text{Obj1, Obj2}\}$ [same as Example 1]
- Roles = $\{R1, R2, R3\}$
- $\text{RPA}(R1) = \{(\text{Obj1, write})\}$, $\text{RPA}(R2) = \{(\text{Obj2, write})\}$, $\text{RPA}(R3) = \{(\text{Obj1, read})\}$
- $\text{RUA}(R1) = \{\text{John}\}$, $\text{RUA}(R2) = \{\text{Lina}\}$, $\text{RPA}(R3) = \{\text{Ray, Tom}\}$
- $\text{RH} = \{(R1, R2), (R1, R3)\}$ [R1 is a senior role than R2, R3]

**Equivalency**

Two access control systems are equivalent iff

- $U$, $O$, and $OP$ are equal for both systems
- $\forall (u, o, op) \in U \times O \times OP. \text{checkAccess}_{\text{system1}}(u,o,op) \equiv \text{checkAccess}_{\text{system2}}(u,o,op)$

EAS and RBAC system defined in example 1 and 2 are equivalent
**ABAC system** is a tuple <U, O, OP, UA, OA, UAValue, OAValue, RangeSet, RuleSet, checkAccess_{ABAC}>

**Example 3**
- U, O, OP are same as Example 1
- UA = \{Position, Dept.\}, OA = \{Type\}

<table>
<thead>
<tr>
<th>UAValue</th>
<th>RangeSet</th>
<th>OAValue</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>User (U)</strong></td>
<td><strong>Position</strong></td>
<td><strong>Object (O)</strong></td>
</tr>
<tr>
<td>John</td>
<td>Officer</td>
<td>Obj1</td>
</tr>
<tr>
<td>Lina</td>
<td>Student</td>
<td>Obj2</td>
</tr>
<tr>
<td>Ray</td>
<td>Officer</td>
<td></td>
</tr>
<tr>
<td>Tom</td>
<td>Officer</td>
<td></td>
</tr>
</tbody>
</table>

- RuleSet contains one separate rule for each operation, \{Rule_{read}, Rule_{write}\}
- **ABAC system is incomplete in Example 3 (No rules given!)**
**ABAC rule structure**

For any operation \( op \in \text{OP} \), Rule\(_{op}\) grammar

\[
\text{Rule}_{op} ::= \text{Rule}_{op} \lor \text{Rule}_{op} \mid (\text{Atomicexp})
\]

\[
\text{Atomicexp} ::= \text{Atomicuexp} \land \text{Atomicoexp} \mid \text{Atomicuexp} \mid \text{Atomicoexp}
\]

\[
\text{Atomicuexp} ::= \text{Atomicuexp} \land \text{Atomicuexp} \mid \text{uexp}
\]

\[
\text{Atomicoexp} ::= \text{Atomicoexp} \land \text{Atomicoexp} \mid \text{oexp}
\]

\[
\text{uexp} \in \{\text{ua}(u) = \text{value} \mid \text{ua} \in \text{UA} \land \text{value} \in \text{Range}(\text{ua})\}
\]

\[
\text{oexp} \in \{\text{oa}(o) = \text{value} \mid \text{oa} \in \text{OA} \land \text{value} \in \text{Range}(\text{oa})\}
\]

\[
\text{checkAccess}_{\text{ABAC}} (a:U, b:O, op:OP) \equiv \text{Rule}_{op}(a:U, b:O)
\]

*** Illustrated ABAC rule examples can be found in later slides***
Contribution

EAS + Incomplete ABAC system (Paper 1) → ABAC policy mining is feasible or not

RBAC + Incomplete ABAC system (Paper 2)

Not feasible?

Use additional attribute to solve infeasibility

@Shuvra Chakraborty
On the Feasibility of Attribute-Based Access Control Policy Mining
Given EAS with supporting data

Check ABAC RuleSet Existence (partition-based approach)

- No
  - Infeasibility correction (use additional attributes with random values)
  - Rule Generation

- Yes
Attribute-Based Access Control (ABAC) limits user to object access by using properties of both user and objects, namely “attribute”.

@Shuvra Chakraborty
Does an equivalent ABAC system exist for the given access control system and supporting data?

@Shuvra Chakraborty
Example A: no ID

**Access Control System**

**Authorization (AUTH)**
- (John, obj1, write)
- (Lina, obj2, write)

**Equivalent ABAC system**

**Supporting Data**

<table>
<thead>
<tr>
<th>User (U)</th>
<th>Position</th>
<th>Dept.</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Officer</td>
<td>CS</td>
</tr>
<tr>
<td>Lina</td>
<td>Student</td>
<td>CS</td>
</tr>
<tr>
<td>Ray</td>
<td>Officer</td>
<td>CS</td>
</tr>
<tr>
<td>Tom</td>
<td>Officer</td>
<td>CS</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Object (O)</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obj1</td>
<td>File</td>
</tr>
<tr>
<td>Obj2</td>
<td>Printer</td>
</tr>
</tbody>
</table>

**RangeSet**

<table>
<thead>
<tr>
<th>Position</th>
<th>{Officer, Student, Faculty}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dept.</td>
<td>{CS, EE}</td>
</tr>
<tr>
<td>Type</td>
<td>{File, Printer, Scanner}</td>
</tr>
</tbody>
</table>

**No IDs → Not possible**
no way to separate John from Ray and Tom

@Shuvra Chakraborty
Example A: with ID

Access Control System

Authorization (AUTH)

<table>
<thead>
<tr>
<th>User (U)</th>
<th>uID</th>
<th>Position</th>
<th>Dept.</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>u1</td>
<td>Officer</td>
<td>CS</td>
</tr>
<tr>
<td>Lina</td>
<td>u2</td>
<td>Student</td>
<td>CS</td>
</tr>
<tr>
<td>Ray</td>
<td>u3</td>
<td>Officer</td>
<td>CS</td>
</tr>
<tr>
<td>Tom</td>
<td>u4</td>
<td>Officer</td>
<td>CS</td>
</tr>
</tbody>
</table>

Supporting Data

Equivalent ABAC system

OAValue

<table>
<thead>
<tr>
<th>Object (O)</th>
<th>oID</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obj1</td>
<td>o1</td>
<td>File</td>
</tr>
<tr>
<td>Obj2</td>
<td>o2</td>
<td>Printer</td>
</tr>
</tbody>
</table>

Rule_write =

(\(uID(U)=u1 \land oID(O)=o1\)) \lor (\(uID(U)=u2 \land oID(O)=o2\))

Entity IDs → Always possible

@Shuvra Chakraborty
Example B

Determine the feasibility before rule generation!

Our solution: Partition-based strategy

@Shuvra Chakraborty
Partition set: example B

Partition Set

John, Obj1
Ray, Obj1
Tom, Obj1

John, Obj2
Ray, Obj2
Tom, Obj2

Lina, Obj1
Lina, Obj2

Partition set is conflict-free w.r.t. write → Yes

@Shuvra Chakraborty
Partition set: example A

**Conflict**

**Partition Set**

<table>
<thead>
<tr>
<th>John, Obj1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ray, Obj1</td>
</tr>
<tr>
<td>Tom, Obj1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>John, Obj2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ray, Obj2</td>
</tr>
<tr>
<td>Tom, Obj2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Lina, Obj2</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Authorization (AUTH)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(John, obj1, write)</td>
</tr>
<tr>
<td>(Lina, obj2, write)</td>
</tr>
</tbody>
</table>

Ray and Tom has no authorization compared to example B
Partition

conflict-free, (UA, OA)

Partition the sets of users and objects present
e.g., {John, Ray, Tom} is partitioned as
{John} and {Ray, Tom}

conflict

if needed
a. Add an attribute to UA (exU)
b. Add an attribute to OA (exO)

Generate a conjunctive clause

conflict-free
(UA U exU), (OA U exO)

Use unique random values to identify
e.g., exU(John)=a, exU(Ray)=b and exU(Tom)=b.
exO is not required here!

OR to Rule_{op}

Exact Solution can be achieved many ways

@Shuvra Chakraborty
Example A: Rule Generation

Conflict:

John, Obj1
Ray, Obj1
Tom, Obj1

Conflict-free:

John, Obj1
Ray, Obj1
Tom, Obj1
Lina, Obj2

Rule\textsubscript{write} \equiv (\text{Position} = \text{officer} \ \text{AND} \ \text{Dept} = \text{CS} \ \text{AND} \ \text{exU} = \text{a} \ \text{AND} \ \text{Type} = \text{File}) \ \text{OR} \ (\text{Position} = \text{student} \ \text{AND} \ \text{Dept} = \text{CS} \ \text{AND} \ \text{Type} = \text{Printer})

\textbf{ABAC system}

\langle U, O, OP, UA, OA, RangeSet, UAValue, OAValue, \{Rule\textsubscript{write}\}, \text{checkAccess}_{\text{ABAC}} \rangle

Equivalent ABAC system generation is always possible!
Partition Set

If all partitions are Conflict-free

There exists at least one conflicted partition

Generate an equivalent ABAC RuleSet

ABAC RuleSet Infeasibility Correction
Unrepresented Partition

Represented: 4
e.g., (Off., CS, F.),
(Stud., CS, Pr.)

Unrepresented: 14
e.g., (Fac., CS, Pr.),
(Stud., EE, Pr.)

Outcome of peculiarity in attribute value assignment

@Shuvra Chakraborty
Formalized notion: feasibility of ABAC policy mining for the first time
The overall asymptotic complexity of ABAC RuleSet Existence problem is:
\[ O(|OP| \times (|U| \times |O|)) \]
The overall asymptotic complexity of ABAC RuleSet Infeasibility Correction is:
\[ O(|OP| \times (|U| \times |O|)^3) \]

Challenges
- Can you replace random values?
- More compact set of rule generation
- Exact solution:
  - Reduce number of split partitions
  - Change number of attributes required
  - Changing existing attribute set, possible?
- Approximate Solution
  - Change authorization
  - Change existing attribute value assignment

@Shuvra Chakraborty