Feasibility Analysis of Access Control Policy Mining

PhD Dissertation Defense

Shuvra Chakraborty
Institute for Cyber Security
Department of Computer Science
The University of Texas at San Antonio

Committee:
Dr. Ravi Sandhu (Advisor and Chair)
Dr. Palden Lama
Dr. Wei Wang
Dr. Xiaoyin Wang
Dr. Ram Krishnan

November 4, 2021
Can we access objects?

### Introduction

Legitimate users get legitimate access only
i.e., Role-Based Access Control (RBAC), Attribute-Based Access Control (ABAC)

*World-Leading Research with Real-World Impact!*
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?
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)
As a matter of growing real-world challenges and advancements in technology, migration of one access control system to another is an emerging problem. The complete or partially automated solution to this migration process is called the access control policy mining problem. During the mining process, a set of assumptions and criteria are imposed to precisely define the migration goals.

The feasibility analysis of the access control policy mining problem formulates the logical framework of the problem, resolves the infeasibility issues possibly arising during the policy mining process so that the solution can satisfy those imposed criteria, and provides a rigorous foundation for the migration process.
Works with the domain of access control models.

- RBAC, ABAC, ReBAC, etc.

Performance measurement is limited to mathematical proof and analyzing algorithmic complexity.

Clear boundary of feasibility issues is yet to be defined.

Depends on how access control models are defined. A separate study is required to extend this.

Does not compete with human expertise at all.

Focusses on exact solutions mostly.
Summary of Contributions

Feasibility Analysis of Access Control Policy Mining

- Feasibility Definition and Solution Algorithm
- Infeasibility Definition and Solution Algorithm
- Proof-of-Concept Implementation
- Exact
- Approximate

- ABAC Policy Mining (Chapter 3)
- AReBAC Policy Mining (Chapter 5)
- ReBAC Policy Mining (Chapter 4)
- Extended Direction for ReBAC and ABAC Feasibility (Chapter 6)

World-Leading Research with Real-World Impact!
Chapter 3
Enumerated Authorization System (EAS) is a tuple

\(<U, O, OP, AUTH, checkAccess_{EAS}>\)

- U, O, and OP are finite sets of users, objects and operations, respectively.
- AUTH \(\subseteq UXOXOP\)

**Example 1:**
- U = \{John, Lina, Ray, Tom\}, OP = \{read, write\}, O = \{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 write operation on Obj2 but read operation is not allowed.</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}>
- RPA : Role Permission Assignment
- RUA: Role User Assignment
- Permission is an object-operation pair
- RH is the role hierarchy relation

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

---

**EAS and RBAC systems 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
- $UA = \{\text{Position, Dept.}\}$, $OA = \{\text{Type}\}$

<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>RangeSet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position</td>
</tr>
<tr>
<td>Dept.</td>
</tr>
<tr>
<td>Type</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>

- RuleSet contains one separate rule for each operation, $\{\text{Rule}_{\text{read}}, \text{Rule}_{\text{write}}\}$
- **ABAC system is incomplete in Example 3**
Workflow-1

Given EAS with supporting data

Check ABAC RuleSet Existence (partition-based approach)

No

Infeasibility correction (use additional attributes with random values)

yes

Rule Generation
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
Workflow-2

(a) Given RBAC system only
(b) Given RBAC system with supporting data

Check ABAC RuleSet Existence (partition-based approach)

No

Infeasibility correction (partition-based approach) (use role-based attributes)

yes

Rule Generation

***Steps are demonstrated with RBAC System (Example 2)
Step 1. Generate role-based attribute set

- For a user u, role-based user attribute denotes the set of roles possessed by u.
- For an object-operation pair (obj, op), role-based object attribute denotes the set of roles where each role contains permission (obj, op).

<table>
<thead>
<tr>
<th>User(U)</th>
<th>uroleAtt</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>{R1, R2, R3}</td>
</tr>
<tr>
<td>Lina</td>
<td>{R2}</td>
</tr>
<tr>
<td>Ray</td>
<td>{R3}</td>
</tr>
<tr>
<td>Tom</td>
<td>{R3}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Object(O)</th>
<th>oroleAtt_{write}</th>
<th>oroleAtt_{read}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obj1</td>
<td>{R1}</td>
<td>{R1, R3}</td>
</tr>
<tr>
<td>Obj2</td>
<td>{R1, R2}</td>
<td>{}</td>
</tr>
</tbody>
</table>

Next step: partition set is generated on set UXO based on similarity in attribute value assignment.
Step 2

Partition set w.r.t. write

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

Partition set w.r.t. read

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

Partition set is conflict-free w.r.t. read and write → YES
Step 3

- Given an operation $op$, if partition set is conflict-free and each partition is uniquely identified by the set of (attribute name, value) pair then RuleSet can be generated [Proved]
- A conjunction of (attribute name, value) pair is made for each conflict-free bold black partition and OR’ed to $Rule_{op}$

\[ Rule_{\text{read}} \equiv (uroleAtt(u) = \{R3\} \land oroleAtt_{\text{write}}(o) = \{R1\} \land oroleAtt_{\text{read}}(o) = \{R1, R3\}) \lor (uroleAtt(u) = \{R1, R2, R3\} \land oroleAtt_{\text{write}}(o) = \{R1\} \land oroleAtt_{\text{read}}(o) = \{R1, R3\}) \]

**Rule write can be constructed same way**

*RuleSet = \{Rule_{\text{write}}, Rule_{\text{read}}\}

**Equivalent ABAC system generation is always possible!**
(b) With supporting data

**Supporting Data**

### Equivalent ABAC system

#### Role Based Access Control System

#### RangeSet

- **Position**: \{Officer, Student, Faculty\}
- **Dept.**: \{CS, EE\}
- **Type**: \{File, Printer, Scanner\}

#### 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>

**Step 1:** Generate partition set based on similarity in attribute value assignment. Partition set might have conflicts!
*Partition set has conflict w.r.t. write → YES
Next step: Apply infeasibility correction
Partition set: corrected

Rule\textsubscript{write} ≡ \langle Position(u) = officer \land Dept(u) = CS \land uroleAtt(u) = \{R1, R2, R3\} \land Type(o) = File \rangle \lor \langle Position(u) = officer \land Dept(u) = CS \land uroleAtt(u) = \{R1, R2, R3\} \land Type(o) = Printer \rangle \lor \langle Position(u) = student \land Dept(u) = CS \land Type(o) = Printer \rangle \rangle

*RuleSet = \{Rule\textsubscript{write}, Rule\textsubscript{read}\}
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

- Ensure minimal partition split
- More compact set of rule generation
- Negative ABAC rules
- Exact solution
  - Reduce number of split partitions
  - Change number of attributes required
  - Effect on changing existing attribute set
Chapter 4
ReBAC ≡ Relationship-Based Access Control

- ReBAC expresses authorization in terms of various direct and indirect relationships amongst entities, most commonly between users.
- Access conditions are usually based on type, depth, or strength of relationships.

Assumption

- Relationship Graph (RG) where users (node) are connected (edge) by social relationships (edge label). Each edge in the RG is labeled with a relation type.
- Only user-to-user relationships are considered.
The feasibility analysis of the ReBAC policy mining problem studies whether the migration process from a given authorization set to ReBAC policy is feasible or not under the set of imposed criteria:

- Relationship Graph (RG) is given
- ReBAC rule structure is given
- Use of entity ID is not allowed
  - Existing literature allows ID
- Equivalent set of ReBAC rules are required

- Solution is guaranteed even if inconsistency arises
  - Infeasibility problem
Evaluation of access request \((a, b, \text{op})\)
- for each \(\text{pathLabelExpr}\) in \(\text{Rule}_{\text{op}}\) substitute True if there exists a simple path \(p\) from \(a\) to \(b\) in RG with path label \(\text{pathLabelExpr}\), otherwise substitute False
- the resulting boolean expression evaluates true \(\rightarrow\) grant, deny otherwise

\[
\begin{align*}
\text{Rule}_{\text{op}} & := \text{Rule}_{\text{op}} \lor \text{Rule}_{\text{op}} \mid \text{pathRuleExpr} \\
\text{pathRuleExpr} & := \text{pathRuleExpr} \land \text{pathRuleExpr} \mid \text{pathLabelExpr} \\
\text{pathLabelExpr} & := \text{pathLabelExpr}.\text{pathLabelExpr} \mid \text{edgeLabel} \\
\text{edgeLabel} & := \sigma, \sigma \in \Sigma
\end{align*}
\]
Feasibility Detection

**Input:**
- Authorizations
- RG
- ReBAC rule structure

**Feasibility detection Algorithm**

**Output:**
- Feasible / Infeasible Status

RG is directed

Failed authorization list is returned

Complexity !!
Feasible

(Bob, Cathy, op)
(Ray, Cathy, op)

Rule_{op} = F

Infeasible

i) (Bob, Cathy, op)
ii) (Cathy, Ray, op)
Infeasible

i) (Bob, Cathy, op)
ii) (Cathy, Ray, op)

Rule_{op} = op
Solution 1

Infeasible

i) (Bob, Cathy, op)

ii) (Cathy, Ray, op)

Rule_{op} = op

Simple

Operation \cap \text{Relationship types} = \{\}

\mid \text{Authorization} \mid \text{edges at worst!}
Table represents path variations with original, non-relationship, inverse and non-relationship inverse edges (row 1, 2, 3, and 4, respectively).

- a,b: users, E and Σ are the sets of edges and relationship type specifiers

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>SCP</th>
<th>SPP</th>
<th>SCPP</th>
</tr>
</thead>
<tbody>
<tr>
<td>((a, b, \sigma) \rightarrow (a, b, \sigma) \in E, \sigma \in \Sigma)</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>((a, b, \overline{\sigma}) \rightarrow (a, b, \sigma) \notin E, \overline{\sigma} \notin \overline{\Sigma})</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td>((a, b, \sigma^{-1}) \rightarrow (b, a, \sigma) \in E, \sigma^{-1} \in \Sigma^{-1})</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td>((a, b, \overline{\sigma^{-1}}) \rightarrow (b, a, \sigma) \notin E, \overline{\sigma^{-1}} \notin \overline{\Sigma^{-1}})</td>
<td></td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>
Path Variations Cont.

Non-relationship edge (iii)

Non-relationship inverse edge (iv)
RREP Variations

- **RREP-0** → SP (i)
- **RREP-1** → SCP (i + iii)
- **RREP-2** → SPP (i + ii)
- **RREP-3** → SCPP (i + ii + iii + iv)

Rule minimization techniques are described in the paper.
Challenges

- Complexity is exponential
- Inexact solution
- More path variations
- Cope up with changes in rule structures
- Other infeasibility solutions
- Extend beyond user-user context
Chapter 5
Background: AReBAC

- AReBAC ≡ Attribute-aware ReBAC
  - Integrate attribute information with ReBAC
  - Makes policy generation more flexible and convenient
  - Attribute-aware Relationship Graph (ARG)

Assumption
- ARG where users (node) are connected (edge) where user and edge have attributes
- Each user and edge have corresponding user and edge attribute values, respectively
- Only user-to-user relationships are considered
The feasibility analysis of the AReBAC policy mining problem studies whether the migration process from a given authorization set to AReBAC policy is feasible or not under the set of imposed criteria:

- Attribute-aware Relationship Graph (ARG) is given
- AReBAC rule structure is given
- Use of entity ID is not allowed
  - Existing literature allows ID
- Equivalent set of AReBAC rules are required

- Solution is guaranteed even if inconsistency arises
  - Infeasibility problem
Evaluation of access request \((a, b, \text{op})\)
- Checks with user attribute values of \(a\) and \(b\)
- If there exists simple path from \(a\) to \(b\) in ARG, Checks with them too!
- The resulting boolean expression evaluates to true \(\rightarrow\) grant, deny otherwise
Feasibility Detection

Input:
Authorizations
ARG
AReBAC rule structure

Feasibility detection
Algorithm

Output:
Feasible / Infeasible Status

ARG is directed

Complexity !!

Failed authorization list is returned
ARG Example

(Female, Student)  (Male, Officer)

Alice  Bob

(Female, Student)  (Male, Student)

Ron  Cathy

UA = {Gender, Profession}
EA = {Relation-type}

<table>
<thead>
<tr>
<th>ReBAC</th>
<th>ABAC</th>
<th>AReBAC</th>
<th>AUTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>✗</td>
<td>✗</td>
<td>✅</td>
<td>(Alice, Ron, op)</td>
</tr>
</tbody>
</table>

Feasible
ARG Example

Rule\textsubscript{op} = ( Gender(e.u) = Female \land Profession(e.u) = Student \land Relation-type(e) = F \land Gender(e.v) = Male \land Profession(e.v) = Student )
ARG Example

(Female, Student) Alice \( F \) (Male, Officer) Bob

ReBAC | ABAC | AReBAC | AUTH
---|---|---|---
\( \times \) | \( \times \) | \( \times \) | (Bob, Alice, op)

Infeasible

(Male, Student) Ron \( F \) (Female, Student) Cathy

World-Leading Research with Real-World Impact!

Page: 42
Infeasibility Solution

Infeasible
(Bob, Alice, op)

Rule\(_{op}\) = (Relation-type(e) = op)

Simple
Minimal edges not guaranteed
|Authorization| edges at worst!

Approximate solution
Summary

- Complexity is exponential
- Inexact solution
- Path variations can be used
- Cope up with changes in rule structures
- Path with cycle
- Other infeasibility solutions
- Extend beyond user-user context
Chapter 6

Extended ReBAC RuleSet Existence Problem (ERREP)
Extended ABAC RuleSet Existence Problem (EAREP)
EAS + ReBAC System (Identical user set)

AUTH(EAS) and AUTH(ReBAC) denote the authorizations allowed by EAS and ReBAC system, respectively

AUTH(EAS) $\neq$ AUTH(ReBAC)

*AUTH(EAS) $\subset$ AUTH(ReBAC)

*AUTH(ReBAC) $\subset$ AUTH(EAS)

*AUTH(EAS) $\cap$ AUTH(ReBAC)$\neq\emptyset$

Exact and approximate solutions are presented
EAS + ABAC System (Identical user and object sets)

AUTH(EAS) and AUTH(ABAC) denote the authorizations allowed by EAS and ABAC system, respectively

AUTH(EAS) \neq AUTH(ABAC)

EAREP-0
*AUTH(EAS) \subseteq AUTH(ABAC)

EAREP-1
*AUTH(ABAC) \subseteq AUTH(EAS)

EAREP-2
*AUTH(EAS) \cap AUTH(ABAC) \neq \emptyset

Exact and approximate solutions are presented
Conclusion and Future Work

Feasibility Analysis of Access Control Policy Mining

- Feasibility Definition and Solution Algorithm
- Infeasibility Definition and Solution Algorithm
- Exact
- Approximate
- Proof-of-Concept Implementation
- ABAC Policy Mining (Chapter 3)
- AReBAC Policy Mining (Chapter 5)
- ReBAC Policy Mining (Chapter 4)
- Extended Direction for ReBAC and ABAC Feasibility (Chapter 6)


5. Extending the Feasibility of Relationship and Attribute-Based Access Control Policy Mining [To be submitted soon]

Thank you

Questions?