Relationship-Based Access Control (ReBAC)

Prof. Ravi Sandhu
Executive Director and Endowed Chair

Lecture 6

ravi.utsa@gmail.com
www.profsandhu.com
Access Control

Fixed policy

Discretionary Access Control (DAC), 1970

Role Based Access Control (RBAC), 1995

Mandatory Access Control (MAC), 1970

Flexible policy

Attribute Based Access Control (ABAC), ????
Access Control

Discretionary Access Control (DAC), 1970

Mandatory Access Control (MAC), 1970

Role Based Access Control (RBAC), 1995

Attribute Based Access Control (ABAC), ????

Relationship Based Access Control (ReBAC), 2008

Fixed policy

Flexible policy
Access Control

Fixed policy

Discretionary Access Control (DAC), 1970

Mandatory Access Control (MAC), 1970

Role Based Access Control (RBAC), 1995

Attribute Based Access Control (ABAC), ????

Relationship Based Access Control (ReBAC), 2008

Flexible policy
ReBAC Models
Online Social Networks (OSNs)

- Social graph is modeled as a directed labeled simple graph $G = (U, E, \Sigma)$
  - Nodes $U$ as users
  - Edges $E$ as relationships
  - $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_n, \sigma_1^{-1}, \sigma_2^{-1}, \ldots, \sigma_n^{-1}\}$ as relationship types supported

Fig. 3. A Sample Social Graph
Access Control in OSNs

- Policy Individualization
  - Users define their own privacy and activity preferences
  - Related users can configure policies too
  - Collectively used by the system for control decision

- User and Resource as a Target
  - e.g., poke, messaging, friendship invitation

- User Policies for Outgoing and Incoming Actions
  - User can be either requester or target of activity
  - Allows control on 1) activities w/o knowing a particular resource and 2) activities against the user w/o knowing a particular access requestor
  - e.g., block notification of friend’s activities; restrict from viewing violent contents
**U2U ReBAC (UURAC) Model**

- **UA**: Accessing User
- **UT**: Target User
- **UC**: Controlling User
- **RT**: Target Resource
- **AUP**: Accessing User Policy
- **TUP**: Target User Policy
- **TRP**: Target Resource Policy
- **SP**: System Policy

- **Policy Individualization**
- **User and Resource as a Target**
- **Separation of user policies for incoming and outgoing actions**
- **Regular Expression based path pattern w/ max hopcounts (e.g., \(<u_a, (f*c,3)>)\)**
Access Request and Evaluation

• Access Request $<u_a, \text{action}, \text{target}>$
  – $u_a$ tries to perform action on target
  – Target can be either user $u_t$ or resource $r_t$

• Policies and Relationships used for Access Evaluation
  – When $u_a$ requests to access a user $u_t$
    • $u_a$’s AUP, $u_t$’s TUP, SP
    • U2U relationships between $u_a$ and $u_t$
  – When $u_a$ requests to access a resource $r_t$
    • $u_a$’s AUP, $r_t$’s TRP, SP
    • U2U relationships between $u_a$ and $u_c$
### Policy Representations

<table>
<thead>
<tr>
<th>Accessing User Policy</th>
<th>$&lt;\text{action}, (\text{start}, \text{path rule})&gt;$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target User Policy</td>
<td>$&lt;\text{action}^{-1}, (\text{start}, \text{path rule})&gt;$</td>
</tr>
<tr>
<td>Target Resource Policy</td>
<td>$&lt;\text{action}^{-1}, u_c, (\text{start}, \text{path rule})&gt;$</td>
</tr>
<tr>
<td>System Policy for User</td>
<td>$&lt;\text{action}, (\text{start}, \text{path rule})&gt;$</td>
</tr>
<tr>
<td>System Policy for Resource</td>
<td>$&lt;\text{action}, (r\text{-typename}, r\text{-typevalue}), (\text{start}, \text{path rule})&gt;$</td>
</tr>
</tbody>
</table>

- $\text{action}^{-1}$ in TUP and TRP is the passive form since it applies to the recipient of action
- TRP has an extra parameter $u_c$ to specify the controlling user
  - U2U relationships between $u_a$ and $u_c$
- SP does not differentiate the active and passive forms
- SP for resource needs $r\text{-typename}, r\text{-typevalue}$ to refine the scope of the resource
Example

- Alice’s policy $P_{\text{Alice}}$:
  - $< \text{poe}, (u_a, (f \ast, 3)) >, < \text{poe}^{-1}, (u_t, (f, 1)) >$
  - $< \text{read}, (u_a, (\Sigma \ast, 5)) >$

- Harry’s policy $P_{\text{Harry}}$:
  - $< \text{poe}, (u_a, (cf \ast, 5) \lor (f \ast, 5)) >, < \text{poe}^{-1}, (u_t, (f \ast, 2)) >$

- Policy of file2 $P_{\text{file2}}$:
  - $< \text{read}^{-1}, \text{Harry}, (uc, \neg (p+, 2)) >$

- System’s policy $P_{\text{Sys}}$:
  - $< \text{poe}, (u_a, (\Sigma \ast, 5)) >$
  - $< \text{read}, (\text{ftype}, \text{photo}), (u_a, (\Sigma \ast, 5)) >$

- “Only Me”
  - $< \text{poe}, (u_a, (\emptyset, 0)) >$ says that $u_a$ can only poke herself
  - $< \text{poe}^{-1}, (u_t, (\emptyset, 0)) >$ specifies that $u_t$ can only be poked by herself

- The Use of Negation Notation
  - $(f f f c \land \neg fc)$ allows the coworkers of the user’s distant friends to see, while keeping away the coworkers of the user’s direct friends
Beyond U2U Relationships

- There are various types of relationships between users and resources in addition to U2U relationships and ownership
  - e.g., share, like, comment, tag, etc
- U2U, U2R and R2R
- U2R further enables relationship and policy administration
U2U, U2R & R2R ReBAC (URRAC) Model

AU: Accessing User
AS: Accessing Session
TU: Target User
TS: Target Session
O: Object
P: Policy
PAU: Accessing User Policy
PAS: Accessing Session Policy
PTU: Target User Policy
PTS: Target Session Policy
PO: Object Policy
PP: Policy for Policy
PSys: System Policy

Decision Module (DM)

Social Graph (SG)
Differences with UURAC

- Access Request
  - (s, act, T) where T may contain multiple objects
- Policy Administration
- User-session Distinction
- Hopcount Skipping
  - Local hopcount stated inside “[[]]” will not be counted in global hopcount.
  - E.g., “([f*,3][[c*, 2]],3)”, the local hopcount 2 for c* does not apply to the global hopcount 3, thus allowing f* to have up to 3 hops.
System-defined conflict resolution for potential conflicts among user-specified policies

Disjunctive, conjunctive and prioritized order between relationship types

- <share-1, (own ∨ tag ∨ share)>
- <read-1, (own ∧ tag)>
- <friend_request, (parent > @)>
Beyond Relationships

- ReBAC usually relies on type, depth, or strength of relationships, but cannot express more complicated topological information
- ReBAC lacks support for attributes of users, resources, and relationships
- Useful examples include common friends, duration of friendship, minimum age, etc.
Attribute-based Policy

- $<\text{quantifier}, f(\text{ATTR}(N), \text{ATTR}(E)), \text{count} \geq i>$

\[
\forall [+1, -2], \text{age}(u) > 18 \\
\exists [+1, -1], \text{weight}(e) > 0.5 \\
\exists [+1, +2, -1], \text{gender}(u) = \text{“male”}
\]
Attribute-based Policy

- **Node attributes**
  - Define user’s identity and characteristics: e.g., name, age, gender, etc.

- **Edge attributes**
  - Describe the characteristics of the relationship: e.g., weight, type, duration, etc.

- **Count attributes**
  - Occurrence requirements for the attribute-based path specification, specifying the minimum
Example: No Attributes
Example: Node Attributes

<access, (u_a, ((f^*, 4): ∃ [+1, -1], occupation = 'student', count ≥ 3))>
Example: Edge Attributes

<read, Photo1, (u_a, ((f*, 3): ∀ [+1, -1], duration ≥ 3 month, _)))>
ReBAC Models
Object-to-Object
Object Relationships in ReBAC

- ReBAC for OSN generally considers only user to user relationship
- OSN has very specific types of resources – photos, notes, comments, which are strongly tied to users.
- Even though some ReBAC models consider general computing systems beyond OSNs they still need users/subjects existence in relationship graph.
ReBAC in General Beyond OSNs

A sample Relationship Graph for Organizational Environment

[RPPM, Crampton et al. ,2014 ]
Existence of Object Relationship Independent of User

Object Relationship in Object-Oriented System (Inheritance, Composition and Association)

History of a Git Project (Version Control System) is a DAG
Limitations of Existing ReBAC Models

- Cannot configure relationship between objects independent of user.
- Cannot express authorization policy solely considering object relationship.
Object to Object Relationship Based Access Control Policy Level Example

ACL(o₁) = {u₁}
ACL(o₂) = {}  
ACL(o₃) = {u₂}

Policy Level Example

policyLevel(a₁,o₁) = 2
policyLevel(a₂,o₁) = 0
policyLevel(a₁,o₂) = 1
policyLevel(a₂,o₂) = 0
policyLevel(a₁,o₃) = 3
policyLevel(a₂,o₃) = 2
policyLevel(a₁,o₄) = 2
policyLevel(a₂,o₄) = 0
OOReBAC: Model Components and Definition

- U is a set of users
- O is a set of objects
- R ⊆ \{ z | z ⊂ O ∧ | z | = 2 \}
- G=(O,R) is an undirected relationship graph with vertices O and edges R
- A is a set of actions
- P^d(o_1) = \{ o_2 | there exists a simple path of length p in graph G from o_1 to o_2 \}
- policyLevel: O × A → N
- ACL: O → 2^U which returns the Access control List of a particular object.
- There is a single policy configuration point. Authorization Policy. for each action a ∈ A, Authz_a(u;U;o;O) is a boolean function which returns true or false and u and o are formal parameters.
- Authorization Policy Language:
  Each action “a” has a single authorization policy Authz_a(u;U;o;O) specified using the following language.
  \[ \phi := u \in PATH_\delta \]
  \[ PATH_\delta := ACL(P^0(o)) \cup \cdots \cup ACL(P^\delta(o)) \text{ where } i = \min(|O| - 1, policyLevel(a,o)) \]
  where for any set X, ACL(X) = \bigcup_{x \in X} ACL(x)
OOReBAC: An Example

Sequence of operations and its outcome:

- $U = \{u_1, u_2, u_3\}$
- $O = \{o_1, o_2, o_3, o_4\}$
- $R = \{\{o_1, o_2\}, \{o_2, o_3\}, \{o_3, o_4\}\}$
- $ACL(o_1) = \{u_1\}$
- $ACL(o_2) = \{u_2\}$
- $ACL(o_3) = \{u_2\}$
- $ACL(o_4) = \{u_3\}$
- $policyLevel(read, o_1) = 2$
- $policyLevel(write, o_1) = 0$
- $policyLevel(read, o_2) = 2$
- $policyLevel(write, o_2) = 1$
- $policyLevel(read, o_3) = 0$
- $policyLevel(write, o_3) = 0$
- $policyLevel(read, o_4) = 2$
- $policyLevel(write, o_4) = 1$

Configuration:

- $A = \{read, write\}$
- $Authz_{read}(u: U, o: O) \equiv u \in P_{policyLevel(read, o)}$
- $Authz_{write}(u: U, o: O) \equiv u \in P_{policyLevel(write, o)}$

Sequence of operations and its outcome:

- $read(u_1, o_3)$, $write(u_1, o_3)$ are denied
- $read(u_2, o_1)$ is allowed, $write(u_2, o_1)$ is denied
- $read(u_1, o_4)$, $write(u_1, o_4)$ are denied
An OOReBAC Instantiation

- $U = \{ u_{pp}, u_{gs}, u_{od}, u_{op}, u_{od}, u_{rp} \}$
- $O = \{ mr_{pp}, mr_{gs}, mr_{cd}, mr_{op}, mr_{cd}, mr_{rp} \}$
- $R = \{ \{ mr_{pp}, mr_{gs}, mr_{cd}, mr_{op}, mr_{cd}, mr_{rp} \} \}$
- $ACL(mr_{pp}) = \{ u_{pp} \}$
- $ACL(mr_{gs}) = \{ u_{gs} \}$
- $ACL(mr_{cd}) = \{ u_{cd} \}$
- $ACL(mr_{op}) = \{ u_{op} \}$
- $ACL(mr_{cd}) = \{ u_{cd} \}$
- $ACL(mr_{rp}) = \{ u_{rp} \}$
- Action = \{ read, write \}

- $policyLevel(read, mr_{pp}) = \infty$, $policyLevel(write, mr_{pp}) = 0$
- $policyLevel(read, mr_{gs}) = \infty$, $policyLevel(write, mr_{gs}) = 0$
- $policyLevel(read, mr_{cd}) = \infty$, $policyLevel(write, mr_{cd}) = 0$
- $policyLevel(read, mr_{op}) = \infty$, $policyLevel(write, mr_{op}) = 0$
- $policyLevel(read, mr_{cd}) = \infty$, $policyLevel(write, mr_{cd}) = 0$
- $policyLevel(read, mr_{rp}) = \infty$, $policyLevel(write, mr_{rp}) = 0$
- Authorization policy:
  - $Auth_{read}(u, o) \equiv u \in \{ policyLevel(read, o) \}$
  - $Auth_{write}(u, o) \equiv u \in \{ policyLevel(write, o) \}$

Sequence of Operations and Outcomes

1) read($u_{rp}$, $mr_{pp}$) : authorized
2) read($u_{cd}$, $mr_{rp}$) : authorized
3) write($u_{rp}$, $mr_{rp}$) : authorized
4) write($u_{rp}$, $mr_{pp}$) : denied
5) write($u_{rp}$, $mr_{pp}$) : denied
ABAC-ReBAC Comparison