RABAC : Role-Centric
Attribute-Based Access Control

MMM-ACNS 2012
Xin Jin, Ravi Sandhu, Ram Krishnan
University of Texas at San Antonio
San Antonio, TX, USA

World-Leading Research with Real-World Impact!
OUTLINE

- Motivation
- Proposed Model
- XACML Profile
- Conclusion

World-Leading Research with Real-World Impact!
Role Based Access Control

Role Explosion

Role number is supposed to be much more than users.

Role Explosion: Different roles have to be defined for slightly different sets of permissions.
One doctor role for each set of patients.

One VisitDoctor role for each project.

World-Leading Research with Real-World Impact!
Related Work

- Role Template, Parameterized Role, Attributed role, etc
- Two level RBAC (SACMAT 12)
- Environment Role, Object Role
- Automatic user-role assignment, TrustBAC
- Relationship based access control (ReBAC)
- Role and organization based access control (ROBAC)

They need modification in user-role and role-permission assignment. Role engineering is the most costly work in constructing RBAC system.

Why can’t we design a solution which can be enforced with least impact to current deployment?
Motivation

applications in areas ranging from health care to defense, in addition to the mainstream commerce systems for which it was designed. As of 2010, the majority of users in enterprises of 500 or more are now using RBAC, according to the Research Triangle Institute. For more information, please contact us at: rbac-info@nist.gov.

**Economic Benefits of Role Based Access Control** Analyzes economic value of RBAC for the enterprise and for the national economy, and provides quantitative economic benefits of RBAC per employee for adopting firms. Of particular interest to firms considering RBAC, report calculates savings from reduced employee downtime, more efficient provisioning, and more efficient access control policy administration, beyond the added security provided by RBAC. NIST’s RBAC research was estimated to have contributed $1.1 billion in economic value. (pdf - Feb. 2011, Research Triangle Institute)

**RBAC vs. ABAC - attribute based access control.** ABAC is a rule-based approach to access control that can be easy to set up but complex to manage. We are investigating both practical and theoretical aspects of ABAC and similar approaches. The following papers discuss ABAC and tradeoffs in design:

D.R. Kuhn, "**Vulnerability Hierarchies in Access Control Configurations**", 4th Symposium on Configuration Analytics and Automation (SAFECONFIG) 2011, IEEE Oct. 31 – Nov. 1 Arlington, Virginia. pp. 1-9: shows that hierarchies of vulnerability detection conditions exist in ABAC rules, such that tests which detect one class of vulnerability are guaranteed to detect other classes.

NIST proposed three alternative revisions to RBAC standard

- **Attribute Centric**
  - Totally attribute based, role as a user attribute
  - Related work: ABAC–alpha model [Jin, DBSEC12], etc

- **Dynamic Roles**
  - Automatically user-role assignment [Kahtani & Sandhu], etc

- **Role Centric RBAC**
  - Not too much research.

With previous work in ABAC-alpha, We provide a formal model for Role-Centric attribute based access control.
OUTLINE

- Motivation
- Proposed Model
- XACML Profile
- Conclusion
Model Components

World-Leading Research with Real-World Impact!
Model Components

- UATT and OATT represent finite sets of user and object attribute functions respectively.
- For each att in UATT ∪ OATT, Range(att) represents the attribute’s range, a finite set of atomic values.
- attType: UATT ∪ OATT → \{set, atomic\}. Specifies attributes as set or atomic valued.
- Each attribute function maps elements in USERS and OBS to atomic or set values.

\[ \forall ua \in UATT. \, ua : USERS \rightarrow \begin{cases} \text{Range}(ua) & \text{if } \text{attType}(ua) = \text{atomic} \\ 2^{\text{Range}(ua)} & \text{if } \text{attType}(ua) = \text{set} \end{cases} \]

\[ \forall oa \in OATT. \, oa : OBS \rightarrow \begin{cases} \text{Range}(oa) & \text{if } \text{attType}(oa) = \text{atomic} \\ 2^{\text{Range}(oa)} & \text{if } \text{attType}(oa) = \text{set} \end{cases} \]

- FILTER = \{F_1, F_2, F_3, \ldots F_n\} is a finite set of boolean functions. For each \( F_i \in \text{FILTER} \). \( F_i : \text{SESSIONS} \times \text{OPS} \times \text{OBS} \rightarrow \{T, F\} \).
Filtering Policy

For each permission \((\text{ops, obs})\) in \(\text{avail\_session\_perms}(\text{se})\),

\(\text{TargetFilter}(\text{obs:OBS})\)

\(\text{Remove}(\text{ops, obs})\) from \(\text{avail\_session\_perms}(\text{se})\)

\(F1(\text{se, ops, obs})\)
\(F2(\text{se, ops, obs})\)
\(F3(\text{se, ops, obs})\)
\(F4(\text{se, ops, obs})\)
\(\ldots\)
\(F_n(\text{se, ops, obs})\)
Filtering Policy

1. Permission filtering policy.
   Language LF\text{Filter} is used to specify each filter function \( F_i(\text{se:SESSIONS}, \text{ops:OPS}, \text{obs:OBS}) \) in \text{FILTER}, where \text{se}, \text{ops} and \text{obs} are formal parameters.

2. Conditions.
   For each \( F_i \in \text{FILTER} \) there is a \text{condition}_i which is a boolean expression specified using language L\text{Condition}.

3. Target\text{Filter} is a function which maps each object to its applicable filter functions as a set. It is illustrated with the pseudo code shown as follows:

   \begin{verbatim}
   Target\text{Filter}(\text{obs:OBS})
   \{
   \hspace{1cm} \text{filter} := \{\};
   \hspace{1cm} \text{condition}_1: \text{filter} := \text{filter} \cup F_1;
   \hspace{1cm} \text{condition}_2: \text{filter} := \text{filter} \cup F_2;
   \hspace{1cm} \cdots
   \hspace{1cm} \text{condition}_n: \text{filter} := \text{filter} \cup F_n;
   \hspace{1cm} \text{return filter;}
   \}
   \end{verbatim}

   \text{Where } F_1, F_2 \ldots F_n \in \text{FILTER} \text{ and } \text{obs} \text{ is formal parameter.}
Common Policy Language (CPL):

\[
\varphi ::= \varphi \land \varphi \mid \varphi \lor \varphi \mid (\varphi) \mid \neg \varphi \mid \exists x \in \text{set}.\varphi \mid \forall x \in \text{set}.\varphi \mid \text{set setcompare set} \mid \text{atomic } \in \text{set} \mid \\
\text{atomic atomiccompare atomic}
\]

\[
\text{setcompare ::= } \subseteq \mid \subseteq \mid \notin \\
\text{atomiccompare ::= } < \mid = \mid \leq
\]

**LCondition**, used to specify each condition, is an instance of CPL where:

\[
\text{set ::= } \text{seto}a(\text{obs}) \mid \text{ConsSet} \\
\text{atomic ::= } \text{atomico}a(\text{obs}) \mid \text{ConsAtomic}
\]

**Example:**

\[
\text{type}(o) = \text{studentrecord } \land (\text{owner}(o) \in \text{GameClub} \lor (\exists \text{reader} \in \text{reader}(o). \text{reader} = \text{user}3))
\]
**Language for Policy**

LFilter, used to specify each filter, is an instance of CPL where:

```plaintext
set ::= setua (sessionowner(se)) | setoa(obs) | ConsSet
atomic ::= atomicua (sessionowner(se)) | atomicoa(obs) | ConsAtomic
setua ∈ {ua | ua ∈ UATT ∧ attType(ua) = set}
atomicua ∈ {ua | ua ∈ UATT ∧ attType(ua) = atomic}
setoa ∈ {oa | oa ∈ OATT ∧ attType(oa) = set}
atomicoa ∈ {oa | oa ∈ OATT ∧ attType(oa) = atomic}
```

**Example:**

\[
major(u) = major(o) \land (\text{location}(u) = \text{utsa} \lor \exists \text{project} \in \text{involvedprj}(u). \text{project} = \text{proj}(o))
\]
### Access Checking

Apply policy and get final available permissions in session

<table>
<thead>
<tr>
<th>Functions</th>
<th>Updates</th>
</tr>
</thead>
<tbody>
<tr>
<td>FilteredSessionPerm (se: SESSIONS)</td>
<td><code>perset = avail_session_perm(se);</code></td>
</tr>
<tr>
<td></td>
<td><code>For each (ops, obs) ∈ perset do</code></td>
</tr>
<tr>
<td></td>
<td><code>if TargetFilter(obs) = {} break;</code></td>
</tr>
<tr>
<td></td>
<td><code>For each function ∈ TargetFilter(obs) do</code></td>
</tr>
<tr>
<td></td>
<td><code>if ~function(se, ops, obs)</code></td>
</tr>
<tr>
<td></td>
<td><code>perset = perset \ {ops, obs}; break;</code></td>
</tr>
<tr>
<td></td>
<td><code>return perset;</code></td>
</tr>
</tbody>
</table>

| CheckAccess (se: SESSIONS, ops: OPS, obs: OBS, result: BOOLEAN) | result = ((ops, obs) ∈ FilteredSessionPerm(se)); |

Check against user request
Package Building Path

Select core RBAC
Option: Advanced Review

Core RBAC

Hier. RBAC
a. Limited
b. General

Choose a or b.
Option: Advanced review

Adhere to Dependency

SSD Relations
a. w/hier.
b. wo/hier

DSD Relations

Permission Filter Policy (PFP)

Requirement package

World-Leading Research with Real-World Impact!
Two role definitions are enough.
OUTLINE

- Motivation
- Proposed Model
- XACML Profile
- Conclusion

World-Leading Research with Real-World Impact!
XACML-Profile for RABAC

World-Leading Research with Real-World Impact!

XACML Profile for RBAC

XACML express permission filtering policy
OUTLINE

- Motivation
- Proposed Model
- Use Case
- XACML Profile
- Conclusion

World-Leading Research with Real-World Impact!
Conclusion

➢ Main contribution
   ➢ RABAC model: Extension to RBAC with filtering policy
   ➢ Define languages for specifying policy
   ➢ Modify functions for access checking

➢ Advantages
   ➢ Without modification to original deployment while mitigating role explosion problem.
   ➢ Retains the administration convenience of RBAC
   ➢ Offer flexibility and administration convenience.

➢ Future work
   ➢ Distinguish user attribute and session attribute.
   ➢ Enhance policy language.

World-Leading Research with Real-World Impact!
Thanks
Any Questions?