A Multi-Tenant RBAC Model for Collaborative Cloud Services

Bo Tang, Qi Li and Ravi Sandhu

Presented by Bo Tang at The 11th International Conference on Privacy, Security and Trust (PST) July 12, 2013 Tarragona, Spain
OUTLINE

- Introduction and Background
- A Family of Multi-Tenant RBAC (MT-RBAC) Models
  - MT-RBAC\textsubscript{0,1,2}
  - Administrative MT-RBAC (AMT-RBAC) model
  - Constraints
- Prototype Implementation and Evaluation
- Related Work
- Conclusion and Future Work
OUTLINE

Introduction

A Family of Multi-Tenant RBAC (MT-RBAC) Models

- MT-RBAC_{0,1,2}
- Administrative MT-RBAC (AMT-RBAC) model
- Constraints

Prototype Implementation and Evaluation

Related Work

Conclusion and Future Work
Cloud Computing

- **Shared infrastructure**
  - [$$$] -----> [$|$|$]

- **Multi-Tenancy**
  - Virtually dedicated resources

- **Drawbacks:**
  - **Data Locked-in**
    - Collaborations can only be achieved through desktop.
    - E.g.: create/edit Word documents in Dropbox.
  - **How to collaborate in the cloud?**

C1. *Charlie* as a *developer* in OS has to access the source code stored in Dev.E to perform his out-sourcing job;

C2. *Alice* as an *auditor* in AF requires read-only access to financial reports stored in Acc.E; and

C3. *Alice* needs read-only accesses to Dev.E and Dev.OS in order to audit the out-sourcing project.
Industry Solutions

- **Microsoft and IBM**: Fine-grained data sharing in SaaS using DB schema
  - Only feasible in DB
- **NASA**: RBAC + OpenStack (Nebula)
  - Lacks ability to support multi-org collaborations
- **Salesforce (Force.com)**: Single Sign-On + SAML
  - Focus on authentication and simple authorization
  - Heavy management of certificates

Source:
- [http://nebula.nasa.gov/blog/2010/06/03/nebulas-implementation-role-based-access-control-rbac/](http://nebula.nasa.gov/blog/2010/06/03/nebulas-implementation-role-based-access-control-rbac/)
OUTLINE

- Introduction

- A Family of Multi-Tenant RBAC (MT-RBAC) Models
  - $\text{MT-RBAC}_{0,1,2}$
  - Administrative MT-RBAC (AMT-RBAC) model
  - Constraints

- Prototype Implementation and Evaluation

- Related Work

- Conclusion and Future Work
If B (resource owner) trusts A then A can assign

- B’s permissions to A’s roles; and
- B’s roles as junior roles to A’s roles.

\[ \text{CanUse}(r_B) = \{A, B, \ldots\} \]
<table>
<thead>
<tr>
<th>Function</th>
<th>Precondition</th>
<th>Update</th>
</tr>
</thead>
<tbody>
<tr>
<td>assignUser (t, r, u)</td>
<td>((t, i) \in TO \land (u, t) \in UO \land t \in canUse(r))</td>
<td>(UA' = UA \cup {(u, r)})</td>
</tr>
<tr>
<td>revokeUser (t, r, u)</td>
<td>((t, i) \in TO \land (u, t) \in UO \land t \in canUse(r) \land (u, r) \in UA)</td>
<td>(UA' = UA \setminus {(u, r)})</td>
</tr>
<tr>
<td>assignPerm (t, r, p)</td>
<td>((t, i) \in TO \land (r, t) \in RO \land (p, t) \in PO)</td>
<td>(PA' = PA \cup {(p, r)})</td>
</tr>
<tr>
<td>revokePerm (t, r, p)</td>
<td>((t, i) \in TO \land (r, t) \in RO \land (p, t) \in PO \land (p, r) \in PA)</td>
<td>(PA' = PA \setminus {(p, r)})</td>
</tr>
<tr>
<td>assignRH (t, r_asc, r_desc)</td>
<td>((t, i) \in TO \land (r_asc, t) \in RO \land t \in canUse(r_desc)) \land \neg(r_asc \gg r_desc) \land \neg(r_desc \gg r_asc))</td>
<td>(\geq' = \geq \cup {(r, q : R</td>
</tr>
<tr>
<td>revokeRH (t, r_asc, r_desc)</td>
<td>((t, i) \in TO \land (r_asc, t) \in RO \land t \in canUse(r_desc)) \land r_asc \gg r_desc)</td>
<td>(\geq' = (\gg \setminus {(r_asc, r_desc)})^*)</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>Equations</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| assignTrust(t, t₁)       | Assign trust to t₁ for tenant t.                                            | \[ t₁ \in T \Rightarrow \quad \quad \le' = \le \cup \{(t, t₁)\} \]
| revokeTrust(t, t₁)       | Revoke trust from t₁ for tenant t.                                          | \[ t₁ \in T \land t \neq t₁ \land t \le t₁ \Rightarrow \quad \quad \le' = \le \setminus \{(t, t₁)\} \]
| addTenant(t)             | Add tenant t to system.                                                     | \[ i \in \mathcal{I} \land t \notin T \Rightarrow \quad \quad T' = T \cup \{t\} \]
| deleteTenant(t)          | Delete tenant t from system.                                                | \[ (t, i) \in \mathcal{TO} \land t \in T \Rightarrow [\forall t₁ : T \Rightarrow \text{revokeTrust}(t, t₁)] \]  
                        |                                                                             | \[ [\forall t₂ : T \Rightarrow \text{revokeTrust}(t₂, t)] \]
|                          |                                                                             | \[ UA' = UA \setminus \{(u, r) | (u, t) \in UO \land (r, t) \in RO\} \]
|                          |                                                                             | \[ PA' = PA \setminus \{(p, r) | (p, t) \in PO \land (r, t) \in RO\} \]
|                          |                                                                             | \[ RH' = RH \setminus \{(r, r') | (r, t) \in RO \land (r', t) \in RO\} \]
|                          |                                                                             | \[ U' = U \setminus \{(u, t) \in UO\} \]
|                          |                                                                             | \[ UO' = UO \setminus \{(u, t) | u \notin U\} \]
|                          |                                                                             | \[ R' = R \setminus \{(r, t) \in RO\} \]
|                          |                                                                             | \[ RO' = RO \setminus \{(r, t) | r \notin R\} \]
|                          |                                                                             | \[ P' = P \setminus \{(p, t) \in PO\} \]
|                          |                                                                             | \[ PO' = PO \setminus \{(p, t) | p \notin P\} \]
|                          |                                                                             | \[ T' = T \setminus \{t\} \]
|                          |                                                                             | \[ TO' = TO \setminus \{(t, i)\} \]

† The notation “⇒” represents an immediate inheritance relation.
‡ This condition avoids the creation of role cycles.
§ The notation “∗” represents recursive updates for the entire RH assignments. Implied RH relations are preserved after revocation.
¶ The revocation of a trust relation automatically triggers updates in the canUse() function of all t’s roles and then corresponding UA and RH accordingly.
Constraints

- Cyclic Role Hierarchy: lead to implicit role upgrades in the role hierarchy
- SoD: conflict of duties
  - Tenant-level
    - E.g.: SOX compliance companies may not hire the same company for both consulting and auditing.
  - Role-level
    - across tenants
- Chinese Wall: conflict of interests among tenants
  - E.g.: do not share infrastructure with competitors.
OUTLINE

- Introduction
- A Family of Multi-Tenant RBAC (MT-RBAC) Models
  - MT-RBAC_{0,1,2}
  - Administrative MT-RBAC (AMT-RBAC) model
  - Constraints
- Prototype Implementation and Evaluation
- Related Work
- Conclusion and Future Work
user = “Charlie”; permission = “(read, /root/)%Dev.E”

tr = “Dev.E”; te = “Dev.OS”
MTAaaS Platform Prototype

- **Experiment Settings**
  - CloudStorage: an open source web based cloud storage and sharing system.
  - Joyent, FlexCloud

- **Authorization Service**
  - Centralized PDP
  - Distributed PEP
MT-RBAC vs RBAC

- More policy references incur more decision time
- MT-RBAC₂ introduces **16 ms** overhead on average.

---

**PDP Performance**

![PDP Performance Graph](chart1)

**Client-End Performance**

![Client-End Performance Graph](chart2)
Evaluation: Scalability

 Scalable by changing either

 PDP capability; or
 Number of PEPs.
OUTLINE

- Introduction
- A Family of Multi-Tenant RBAC (MT-RBAC) Models
  - MT-RBAC_{0,1,2}
  - Administrative MT-RBAC (AMT-RBAC) model
  - Constraints
- Prototype Implementation and Evaluation
- Related Work
- Conclusion and Future Work
Characteristics of Cloud

- **Agility**
  - Collaboration and collaborators are temporary

- **Centralized Facility**
  - No need to use cryptographic certificates

- **Homogeneity**
  - Same access control model in each tenant

- **Out-Sourcing Trust**
  - Collaboration spirit
Literature in Multi-org/dom

- **RBAC**
  - CBAC, GB-RBAC, ROBAC (e.g.: player transfer in NBA)
  - Require central authority managing collaborations

- **Delegation Models**
  - dRBAC and PBDM (e.g.: allowing subleasing)
  - Lacks agility (which the cloud requires)

- **Grids**
  - CAS, VOMS, PERMIS
  - Absence of centralized facility and homogeneous architecture (which the cloud has)
Role-based Trust

- RT, Traust, RMTN AND RAMARS_TM
- Calero et al: towards a multi-tenant authorization system for cloud services
  - Implementation level PoC
  - Coarse-grained trust model
- MTAS
- Suits the cloud (out-sourcing trust)

Critical: trust model
Table 3.3: Trust Model Comparison. $A$ and $B$ represent two entities, issuers and tenants respectively in RT, MTAS and MT-RBAC. $A$ represents the resource owner and $B$ the requester.

<table>
<thead>
<tr>
<th></th>
<th>RT</th>
<th>MTAS</th>
<th>MT-RBAC</th>
</tr>
</thead>
<tbody>
<tr>
<td>trust relation required</td>
<td>$A$ trust $B$</td>
<td>$B$ trust $A$</td>
<td>$A$ trust $B$</td>
</tr>
<tr>
<td>trust assigner</td>
<td>$A$</td>
<td>$B$</td>
<td>$A$</td>
</tr>
<tr>
<td>authorization assigner</td>
<td>$A$</td>
<td>$A$</td>
<td>$B$</td>
</tr>
<tr>
<td>User Assignment (UA)</td>
<td>$U \rightarrow \mathcal{A.R}$</td>
<td>$U \rightarrow \mathcal{A.R}$</td>
<td>$B.U \rightarrow B.R \cup \mathcal{A.R}$</td>
</tr>
<tr>
<td>Permission Assignment (PA)</td>
<td>$\mathcal{A.P} \rightarrow \mathcal{A.R}$</td>
<td>$\mathcal{A.P} \rightarrow \mathcal{A.R} \cup \mathcal{B.R}$</td>
<td>$\mathcal{B.P} \rightarrow \mathcal{B.R}$</td>
</tr>
<tr>
<td>Role Hierarchy (RH)</td>
<td>$\mathcal{A.R} \leq \mathcal{B.R}$</td>
<td>$\mathcal{A.R} \leq \mathcal{B.R}$</td>
<td>$\mathcal{A.R} \leq \mathcal{B.R}$</td>
</tr>
<tr>
<td>require common vocabulary</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>require centralized facility</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
OUTLINE

- Introduction
- A Family of Multi-Tenant RBAC (MT-RBAC) Models
  - MT-RBAC_{0,1,2}
  - Administrative MT-RBAC (AMT-RBAC) model
  - Constraints
- Prototype Implementation and Evaluation
- Related Work
- Conclusion and Future Work
Conclusion

- Collaboration needs among cloud services
- MT-RBAC model family
  - Formalization
  - Administration
  - Constraints
- MTAaaS architecture viable in the cloud
- Overhead $\approx 16\text{ms}$ and scalable in the cloud
- Comparison of role-based trust models
Future Work

- Cross-tenant trust models in cloud computing
- Other multi-tenant access control models
  - MT-ABAC
  - MT-RT
  - MT-PBAC and more.
- Implementation MT-RBAC in OpenStack API.
Thank You!
Multi-Tenant Authorization as a Service (MTAaaS)

- User IDs
- Resource Catalogs
- Authz Policies

MTAaaS

- Multi-Tenant Access Control
- Cross-Tenant Access

Cloud services and applications:
- Flickr
- Dropbox
- Google
- Rackspace Hosting
- Amazon Web Services
- Windows Azure
- Salesforce

© ICS at UTSA
World-Leading Research with Real-World Impact!