Multi-Tenant Access Control for Collaborative Cloud Services

CS6393 Spring 2014
PhD Seminar

Bo Tang

April 11, 2014
What is Multi-Tenancy?
Cloud & Multi-Tenancy

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

- **Multi-Tenancy**
  - Virtually dedicated resources
    - E.g.: rent-a-car

- **Problems:**
  - Who owns the data?
  - How to collaborate across tenants?
    - Even if across my own tenants?

Characteristics of Cloud

- **Distributed Authority**
  - Each tenant manages its own authorization

- **Centralized Facility**
  - Resource pooling

- **Agility**
  - Tenants, users and resources are temporary

- **Homogeneity**
  - Identical or similar architecture and system settings

- **Out-Sourcing Trust**
  - Built-in collaboration spirit
Define Tenant

- All deployment models are multi-tenant
  - E.g.: public cloud, private cloud and community cloud.

- From Cloud Service Provider (CSP) perspective
  - A billing customer
  - Manages its own users and cloud resources

- From consumer perspective
  - An individual, an organization or a department in an organization, etc.
  - Virtually dedicated space with on-demand self-service
DevOps in the Cloud

Andy@DEV
Tenant Administrator

Andy
Administrator

Andy@PRD
Tenant Administrator

Development (DEV)

Teresa
Tester

Dennis
Developer

HR.DEV
Sales.DEV

Production (PRD)

Omar
Operator

HR.PRD
Sales.PRD

Normal Case (Intra-Tenant Access)

DevOps Case (Cross-Tenant Access)
Multi-org/dom A/C Literature

- **RBAC**
  - CBAC, GB-RBAC, ROBAC
  - No cross-organization interaction
  - Require central authority managing collaborations

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

- **Grids**
  - CAS, VOMS, PERMIS
  - Heavy authorization overhead due to the absence of homogeneous infrastructure (which the cloud has)
Role-based Trust

- RT, Traust, RMTN AND RAMARS_TM
- Calero et al: towards a multi-tenancy authorization system for cloud services
  - Implementation level PoC
  - Open for extensions in trust models
- Suits the cloud (out-sourcing trust)

Challenge:
- Trust relation
- Finer-grained models
- Administration
Scope and Assumptions

➢ Standardized APIs
  - Cross-tenant accesses are functionally available

➢ Proper authentication of users

➢ Removable assumptions:
  - One Cloud Service
    - Of a kind: IaaS, PaaS or SaaS etc.
  - Two-Tenant Trust (rather than community trust)
  - Unidirectional Trust Relations (like follow in Twitter)
  - Unilateral Trust Relations (trustor or trustee)
Multi-Tenant Access Control

Ontology (O)

CTTM

Policy (P)

MTAS
MT-RBAC
MT-RT
MT-ABAC

Enforcement (E)

MTAaaS Framework

Implementation (I)

Domain Trust in OpenStack
Tenant Trust (TT) relation is not partial order

It is

- Reflexive: $A \preceq A$
- But not transitive: $A \preceq B \land B \preceq C \nrightarrow A \preceq C$
- Neither symmetric: $A \preceq B \nleftrightarrow B \preceq A$
- Nor anti-symmetric: $A \preceq B \land B \preceq A \nleftrightarrow A \equiv B$
CTTM Trust Types

Four potential trust types:

❖ Type-α: trustor can give access to trustee.
❖ Type-β: trustee can give access to trustor.
❖ Type-γ: trustee can take access from trustor.
❖ Type-δ: trustor can take access from trustee.

- No meaningful use case, since the trustor holds all the control of the cross-tenant assignments of the trustee’s permissions.
Cross-Tenant Trust in DevOps

Example: Temporary DevOps access

- [\$]: grant Dennis@DEV access to HR.PRD
- **Trust-α:**
  - PRD trusts DEV so that PRD can say [\$].
- **Trust-β:**
  - DEV trusts PRD so that PRD can say [\$].
- **Trust-γ:**
  - PRD trusts DEV so that DEV can say [\$].
Formalized CTTM Model

- **User Ownership (UO)**
- **Permission Ownership (PO)**
- **Authorization Assignment (AA)**

**Tenant Trust (TT)**

**Tenants (T)**

**Users (U)**

**Permissions (P)**
Role-Based CTTM

- Users (U)
- Roles (R)
- Permissions (P)
- Tenants (T)

Diagram Relationships:
- Tenant Trust (TT)
- Role Ownership (RO)
- Permission Ownership (PO)
- User Assignment (UA)
- Permission Assignment (PA)
- Role Hierarchy (RH)
Role-Based CTTM

OpenStack Identity + RH v2.0
Multi-Tenant Authz. as a Service (MTAaaS)
Cross-Tenant Trust in DevOps

Example: Temporary DevOps access

- [\$]: grant Dennis@DEV access to HR.PRD
- Trust-\(\alpha\) (RT):
  - PRD trusts DEV so that PRD can say [\$].
- Trust-\(\beta\) (MTAS):
  - DEV trusts PRD so that PRD can say [\$].
- Trust-\(\gamma\) (MT-RBAC):
  - PRD trusts DEV so that DEV can say [\$].
Tenants: issuers

MTAS

Users (U) -> Roles (R)
- User Ownership (UO)
- User Assignment (UA)

Roles (R) -> Permissions (P)
- Role Ownership (RO)
- Role Assignment (PA)

Users (U) -> Roles (R)
- roles(s)

Sessions (S) -> roles(s)

Issuers (I) -> Trust
- Issuer Trust (IT)

Permission Ownership (PO)
-

Role Hierarchy (RH)
Issuers are administered by the CSP

Each issuer administer:

- trust relations with other issuers
- entity components: users, roles and permissions
- UA, PA and RH assignments
  - Cross-tenant assignments are issued by the trustee
    - UA: trustor users to trustee roles
    - PA: trustee permissions to trustor roles
    - RH: trustee roles junior to trustor roles
Finer-grained Trust Models

- Problem of MTAS trust model
  - Over exposure of trustor’s authorization information

- Trustor-Centric Public Role (TCPR)
  - Expose only the trustor’s public roles

- Relation-Centric Public Role (RCPR)
  - Expose public roles specific for each trust relation
Example MTAS policy structure

- **i1 policysets**
  - Trust <PolicySet> (TPS:i1)
    - <Target>
      - subject-tenant = i1
  - <PolicySetIdReference>
  - Role <PolicySet> (RPS:i1:...)
    - <Target>
      - subject-role = i1:...
  - <PolicySetIdReference>
  - Perm. <PolicySet> (PPS:i1)
    - <Target>
      - resource-tenant = i1
  - UA
  - RH

- **i2 policysets**
  - Trust <PolicySet> (TPS:i2)
    - <Target>
      - subject-tenant = i2
  - <PolicySetIdReference>
  - Role <PolicySet> (RPS:i2:...)
    - <Target>
      - subject-role = i2:...
  - <PolicySetIdReference>
  - Perm. <PolicySet> (PPS:i2)
    - <Target>
      - resource-tenant = i2
  - Cross-Issuer UA
  - Cross-Issuer RH
  - RH
  - UA
  - RH
  - Cross-Issuer PA
  - PA
MTAaaS Platform Prototype

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

➤ Authorization Service
  ❖ Centralized PDP
  ❖ Distributed PEP
MTAS introduces \( \approx 12 \text{ ms} \) overhead in average.

Scalable

- Capability proportional to throughput

![Performance](chart1.png)

![Scalability](chart2.png)
Example: Temporary DevOps access

- [$] grant Dennis@DEV access to HR.PRD

- Trust-α (RT):
  - PRD trusts DEV so that PRD can say [$].

- Trust-β (MTAS):
  - DEV trusts PRD so that PRD can say [$].

- Trust-γ (MT-RBAC):
  - PRD trusts DEV so that DEV can say [$].
MT-RBAC

Issuers: Real-world admins
Issuers administer tenants

Each issuer administer:

- Trust relations from owned tenants
- Entity components: tenants, users, roles and permissions
- UA, PA and RH assignments
  - Cross-tenant assignments are issued by the trustee’s owning issuer
    - UA: trustee users to trustor roles
    - PA: trustor permissions to trustee roles
    - RH: trustor roles junior to trustee roles
Finer-grained Trust Models

➢ Trustee-Independent Public Role (TIPR)
  ◆ Expose only the trustor’s public roles

➢ Trustee-Dependent Public Role (TDPR)
  ◆ Expose public roles specific for each trustee
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.

Diagram:

- Tenant 1: M1 -> E1
- Tenant 2: M2 -> E2
- M1 -> E2
- M2 -> E1

© ICS at UTSA World-Leading Research with Real-World Impact!
Policy Specification of MT-RBAC

Role <PolicySet>
  <Target>
  subject-role = tr....
  subject-tenant = te
  <PolicySetIdReference>

Cross-Tenant UA

Trust <PolicySet> tr
  <Target>
  subject-tenant = tr
  <PolicySetIdReference>

TDPR <PolicySet> tr
  <Target>
  subject-tenant = tr
  <PolicySetIdReference>

Permission <PolicySet>
  <Target>
  resource-id = tr....
  <PolicySetIdReference>

Role <PolicySet>
  <Target>
  subject-role = te....
  <PolicySetIdReference>

UA

Trust <PolicySet> te
  <Target>
  subject-tenant = te
  <PolicySetIdReference>

TDPR <PolicySet> te
  <Target>
  subject-tenant = te
  object-role = tr....
  <PolicySetIdReference>

Permission <PolicySet>
  <Target>
  resource-id = te....
  <PolicySetIdReference>

Cross-Tenant RH

PA

RH
MT-RBAC vs RBAC

More policy references incur more decision time

MT-RBAC₂ introduces \textbf{16 ms} overhead in average.
_scalable by either

- Enhancing PDP capability; or
- Increasing PEP amount.
Example: Temporary DevOps access

- [\$]: grant Dennis@DEV access to HR.PRD

- **Trust-α (RT):**
  - PRD trusts DEV so that PRD can say [\$].

- **Trust-β (MTAS):**
  - DEV trusts PRD so that PRD can say [\$].

- **Trust-γ (MT-RBAC):**
  - PRD trusts DEV so that DEV can say [\$].
MT-RT: “P” layer model of RT with MT features

- No certificate is required (centralized facility)
- Trust (delegation) in OpenStack identity?
DevOps in OpenStack

DOMAINS

PROJECTS

USER / GROUPS

ROLES

DEVELOPMENT DOMAIN

PRODUCTION DOMAIN

DOMAINS

токен

用户/组

角色

用户-角色-项目

权限

项目

租户

服务

角色

用户

组

权限

创建虚拟机

添加映像

添加IP

上传文件

创建虚拟机

添加映像

添加路由器

列出文件

删除虚拟机

删除端口

读取文件

政策

服务
Cloud Admin

Domain A Admin
  - Project A1 Admin
  - Project A2 Admin

Domain B Admin
  - Project B1 Admin
  - Project B2 Admin

rule:add_user_to_tenant -> (role:keystone_admin ||
  (role:tenant_admin && tenant_id:%(target_tenant_id)s) ||
  (domain_role:domain_admin && domain_id:%(target_domain_id)s))

rule:add_tenant_to_domain -> (role:keystone_admin ||
  (domain_role:domain_admin && domain_id:%(target_domain_id)s))

Source: https://wiki.openstack.org/wiki/Domains
Domain Trust in OpenStack

- **Enhanced security**
  - Limit visibility in the specific domain
  - Prevent malicious / dumb assignments

- **Better management with Dtrust**
  - Specified by domain admin available for project admin
  - Automatic revocation of cross-domain assignments
  - Finer-grained control enabled
    - Only expose users with certain roles
    - May specify collaborating users and projects
Conclusion

- **Completed Work**
  - CTTM
  - MTAaaS
  - MTAS
  - MT-RBAC

- **On-going research**
  - MT-RT
  - MT-ABAC
  - Domain Trust in OpenStack
Thank You!