An Attribute-Based Access Control Extension for OpenStack and its Enforcement Utilizing the Policy Machine

Prof. Ravi Sandhu
Executive Director and Endowed Chair

2nd IEEE International Conference on Collaboration and Internet Computing (CIC)
November 1-3, 2016

Smriti Bhatt, Farhan Patwa and Ravi Sandhu
Department of Computer Science

ravi.sandhu@utsa.edu
www.profsandhu.com
www.ics.utsa.edu

Outline

- Introduction
- RBAC and ABAC
- Simplified OpenStack Access Control (OSAC) Model
- An ABAC Extension for OpenStack
- Policy Machine (PM) and its Architecture
- ABAC Enforcement Architecture
- Authorization using AE and PM
- Use Cases
- Evaluation
- Discussion and Analysis
- Conclusion and Future Work

Introduction

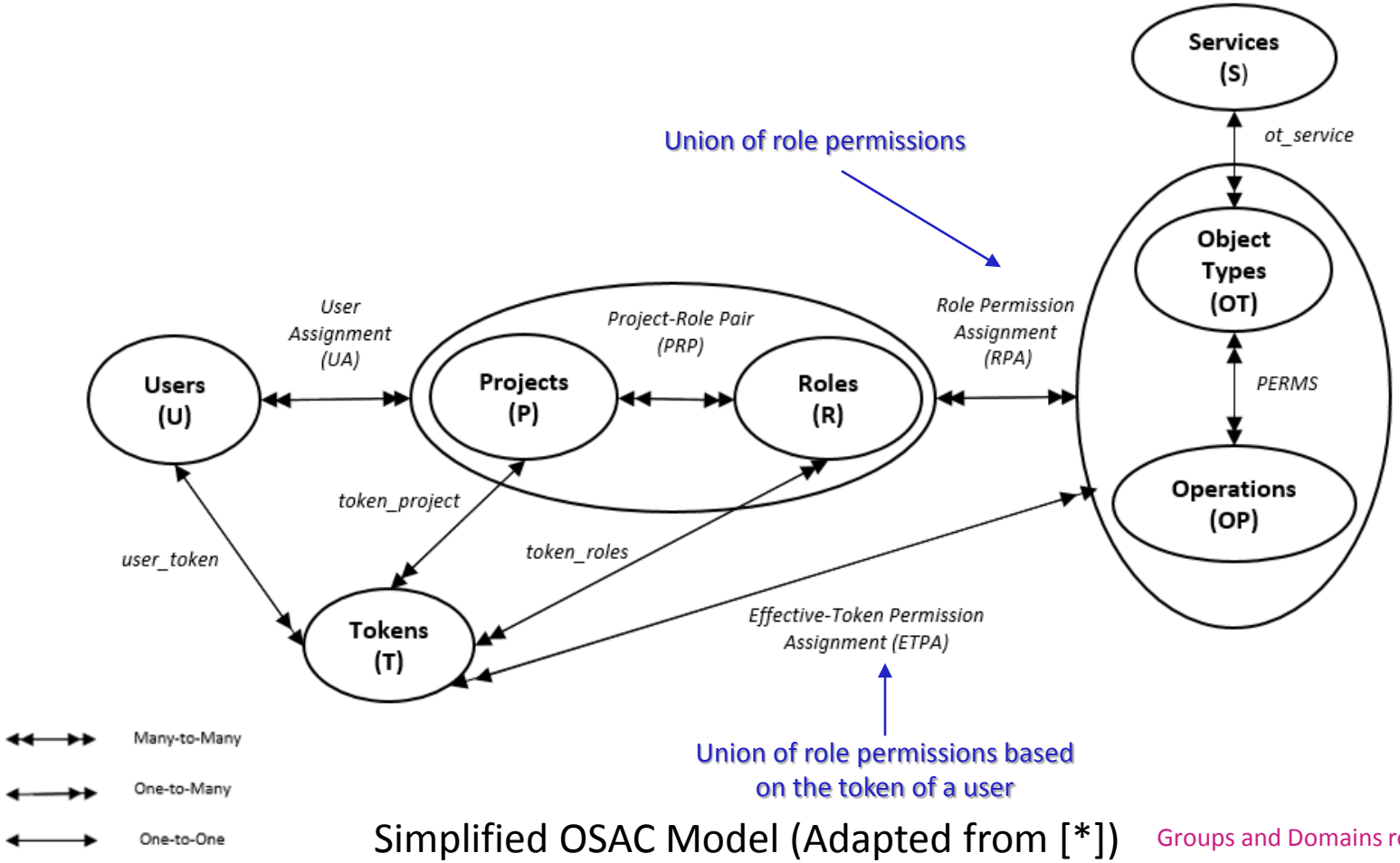
- **RBAC:**
  - Most dominant access control model
  - Major cloud computing platforms:
    - OpenStack
    - AWS
    - Microsoft Azure

    Authorization based on RBAC model
  - Limitations:
    - Role explosion
- **ABAC:**
  - Access control based on attributes
  - Enhanced flexibility and fine grained access control
  - Implement ABAC models in real-world applications
- A *gradual* shift from RBAC to ABAC models

RBAC and ABAC

- Three different ways of combining RBAC and ABAC by NIST –
  - *dynamic roles:* uses user and context attributes to dynamically assign roles to users
  - *attribute-centric:* roles just another attribute of users with no special semantics
  - *role-centric:* constrains role permissions based on user attributes
- Proposed a role-centric ABAC extension for OpenStack
- Combining advantages of both RBAC and ABAC

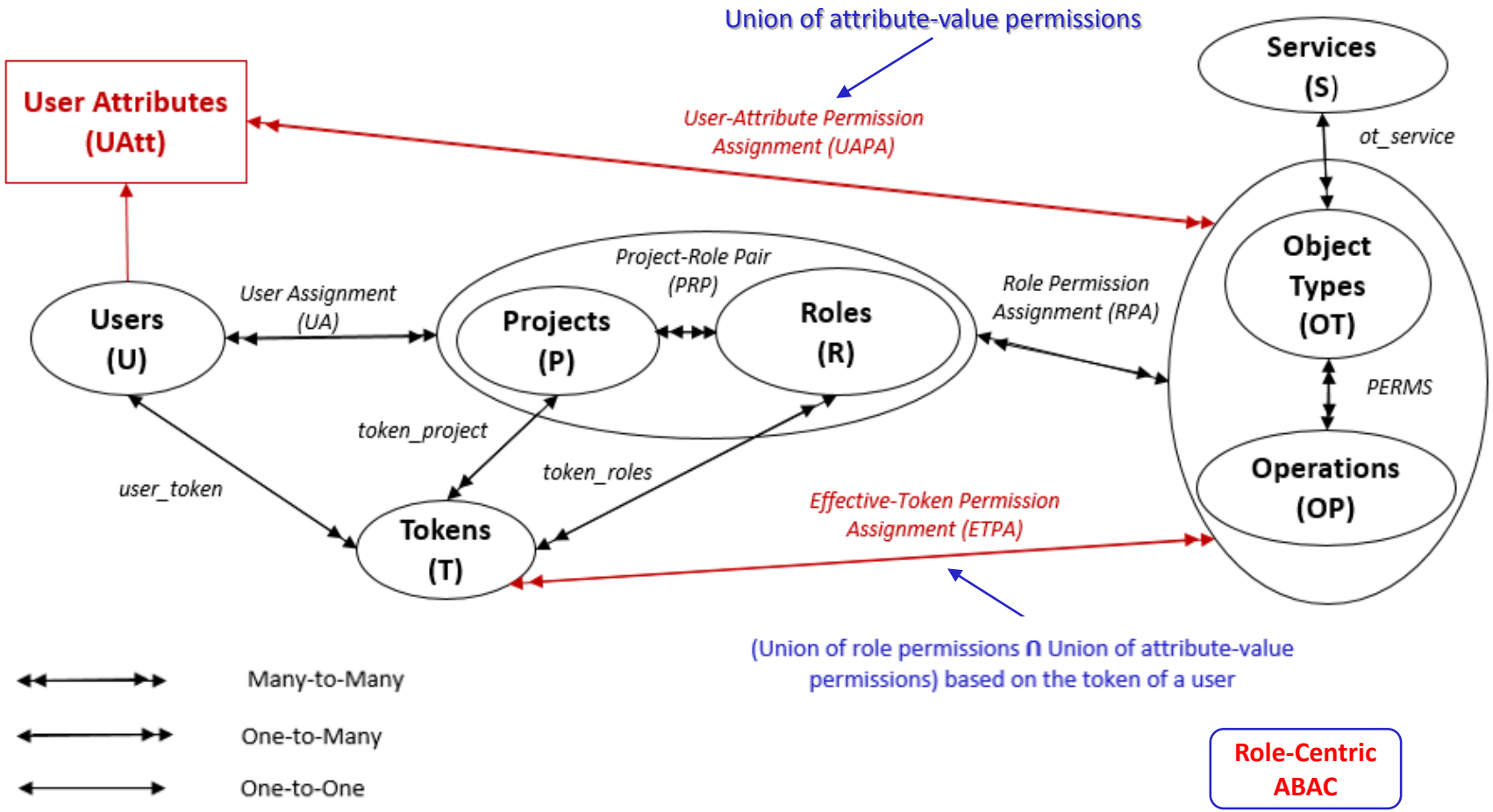
Simplified OpenStack Access Control (OSAC) Model



Simplified OSAC Model (Adapted from [*]) Groups and Domains removed

* B. Tang and R. Sandhu, "Extending OpenStack access control with domain trust," in International Conference on Network and System Security. Springer, 2014, pp. 54–69

An ABAC Extension for OpenStack



User-Attribute Enhanced OSAC in Single Tenant

An ABAC Extension for OpenStack

- Extended simplified OSAC with user attributes –
  - *UAtt* – a finite set of user attribute functions
  - For each *uatt* in *Uatt*, *Range(uatt)* is a finite set of atomic value
  - *UAPA* – user-attribute value permission assignment
- For any user –
  - maximum permissions determined based on roles in the token,
  - further constrained by permissions associated to its user-attribute values
- Currently, model designed for atomic valued attributes only
- Object attributes next challenge to explore
- Enforced this model in OpenStack utilizing the Policy Machine (PM)

Policy Machine (PM)

- General-purpose attribute-based access control framework
- Express and enforce arbitrary access control policies
- Provide a unified platform supporting:
  - Commonly known and implemented access control policies
  - Combinations of policies
  - New access control policies

PM Core Elements

- Users
- Objects
- User Attributes
- Object Attributes
- Operations, Access Rights
- Processes
- Policy Classes

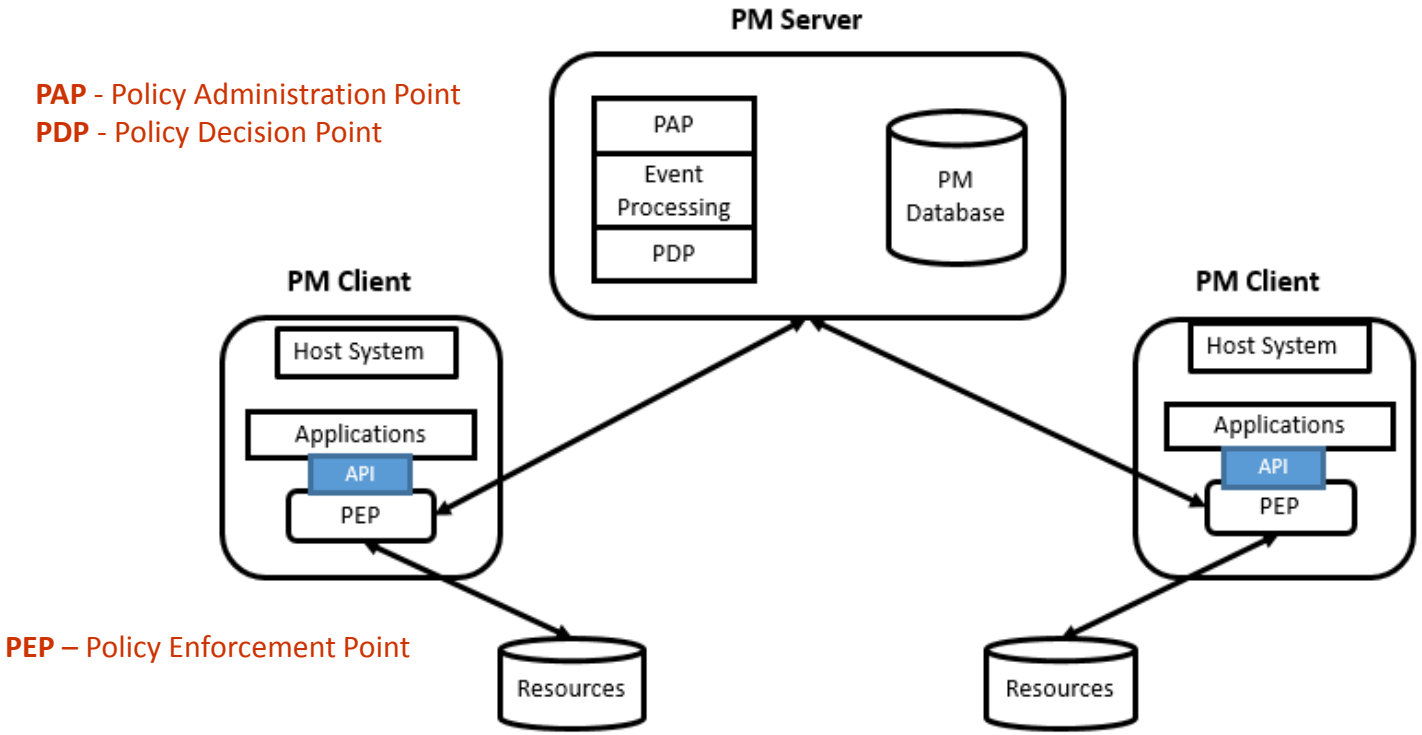
PM Relations

- Assignment
- Association
- Prohibition
- Obligation

✓ **assignment**—for specifying relationships between policies, users, and user attributes, objects and object attributes

✓ **association** – for defining policies through associations between user attributes and object attributes or objects through some operations

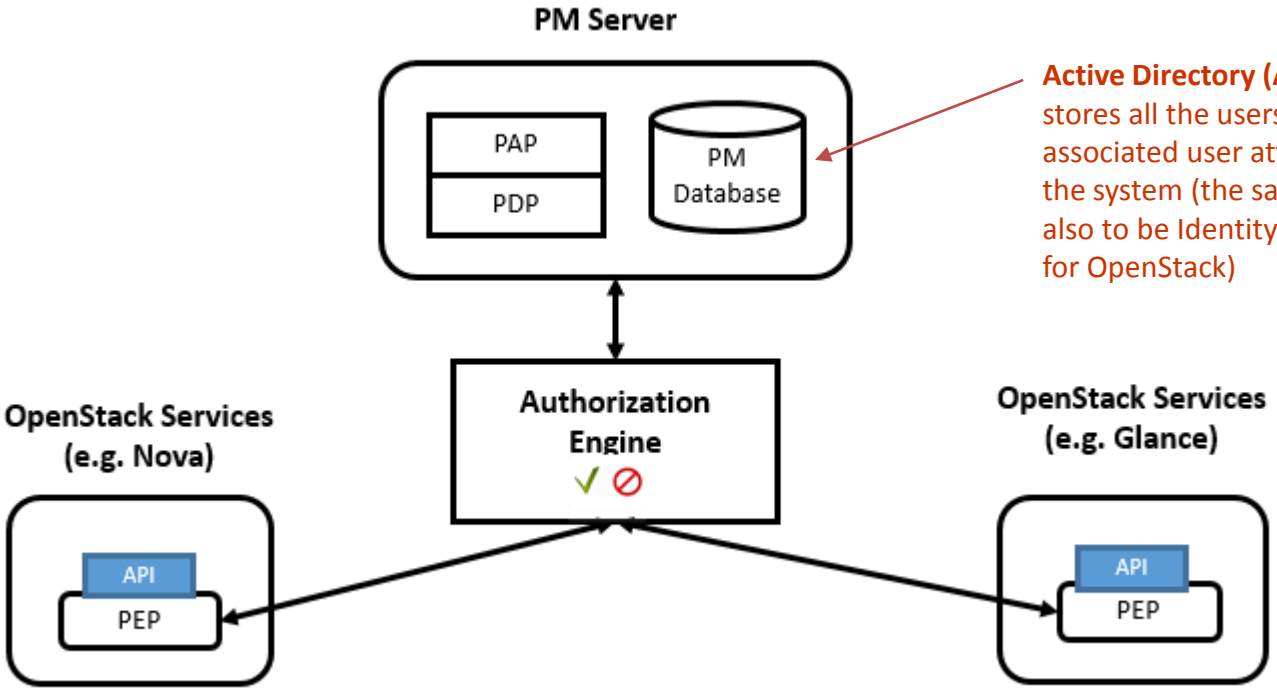
PM Architecture



Policy Machine Architecture (Adapted from [*])

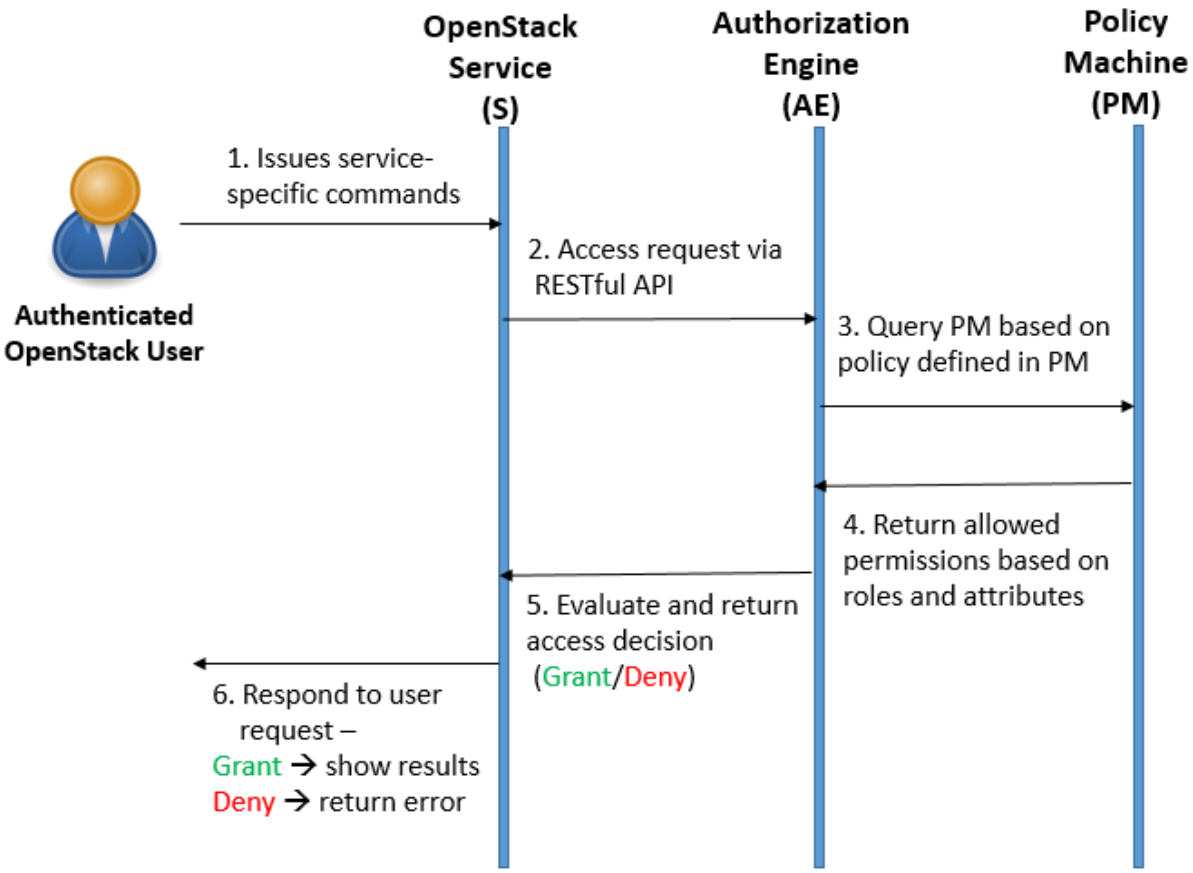
* D. Ferraiolo, V. Atluri, and S. Gavrila, "The Policy Machine: A novel architecture and framework for access control policy specification and enforcement," J. of Sys. Architecture, vol. 57, no. 4, pp. 412–424, 2011

ABAC Enforcement Architecture



An ABAC Enforcement Architecture for OpenStack using PM

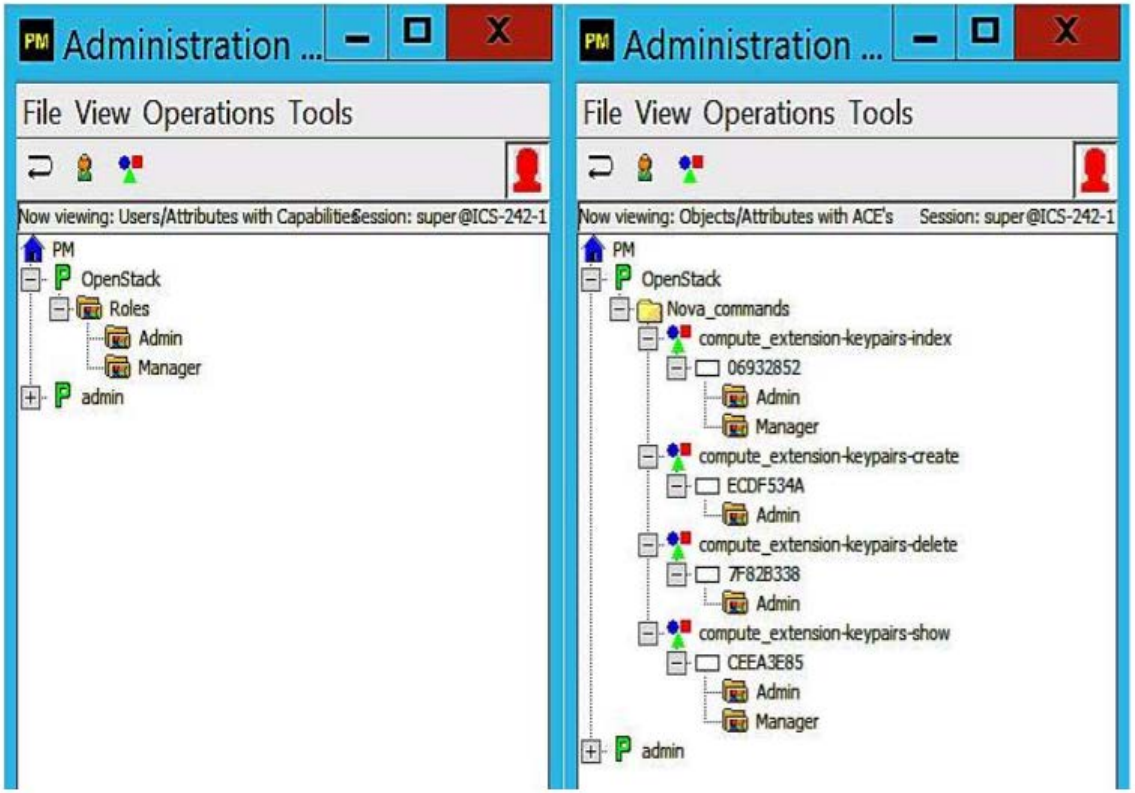
Authorization Sequence Diagram



OpenStack Authorization using AE and PM

Use Case I

- A Simplified OSAC RBAC Policy
  - Roles: *{Admin, Manager}*
  - Commands (c):
    - *compute extension-keypair-index*
    - *compute extension-keypair-create*
    - *compute extension-keypair-delete*
    - *compute extension-keypair-show*
  - Authorization rules for any user u:
    - *compute extension-keypair-create → Role(u) = Admin*
    - *compute extension-keypair-delete → Role(u) = Admin*
    - *compute extension-keypair-index → (Role(u) = Admin ᵛ Role(u) = Manager)*
    - *compute extension-keypair-show → (Role(u) = Admin ᵛ Role(u) = Manager)*



A Role-Based Access Control Policy in PM

Use Case I

Role = Admin

```
stack@opm-1:/opt/stack/nova/nova$
stack@opm-1:/opt/stack/nova/nova$ nova --os-username user1 --os-password ****** --os-tenant-name test
keypair-add test4 >test4.pem
stack@opm-1:/opt/stack/nova/nova$ nova --os-username user1 --os-password ****** --os-tenant-name test
keypair-list
+-------+-------------------------------------------------+
| Name  | Fingerprint                                     |
+-------+-------------------------------------------------+
| test  | ************************************************* |
| test1 | ************************************************* |
| test2 | ************************************************* |
| test3 | ************************************************* |
| test4 | ************************************************* |
+-------+-------------------------------------------------+
stack@opm-1:/opt/stack/nova/nova$ nova --os-username user2 --os-password ****** --os-tenant-name test
keypair-add test5 >test5.pem
ERROR (Forbidden):  Policy doesn't allow [compute_extension:keypairs:create] to be performed for role
[Manager] due to role (HTTP 403) (Request-ID: req-88a0af9a-d6ae-46d5-b308-3b59a2fa2908)
stack@opm-1:/opt/stack/nova/nova$
```

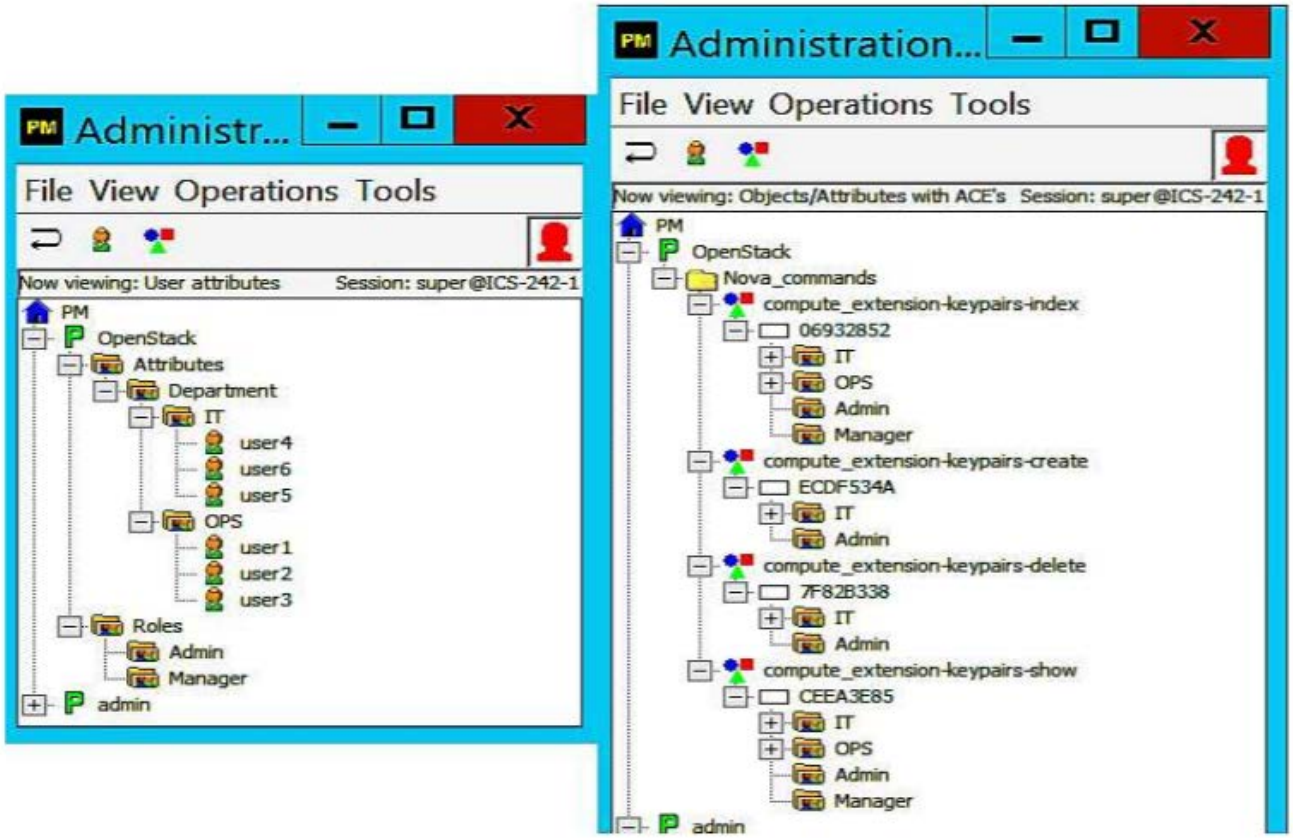
Role = Manager

OpenStack Enforcement Results

Use Case II

- A Role-Centric ABAC Policy
  - Roles: *{Admin, Manager}*
  - Department: *{IT, OPS}*
  - Commands (c):
    - *compute extension-keypair-index*
    - *compute extension-keypair-create*
    - *compute extension-keypair-delete*
    - *compute extension-keypair-show*
  - Authorization rules for any user u:
    - *compute extension-keypair-create → (Role(u) = Admin ^ Dep(u) = IT)*
    - *compute extension-keypair-delete → (Role(u) = Admin ^ Dep(u) = IT)*
    - *compute extension-keypair-index → ((Role(u) = Admin ᵛ Role(u) = Manager) ^ (Dep(u) = IT ᵛ Dep(u) = OPS))*
    - *compute extension-keypair-show → ((Role(u) = Admin ᵛ Role(u) = Manager) ^ (Dep(u) = IT ᵛ Dep(u) = OPS))*

Use Case II



A User-Attribute Enhanced OSAC Policy in PM

Use Case II

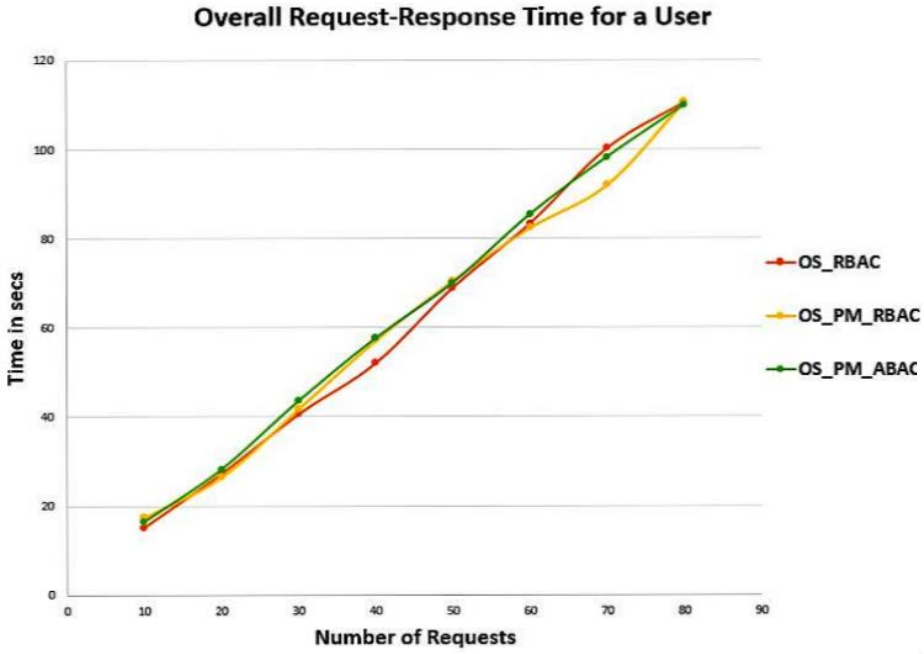
Role = Admin and
Department = OPS

```
stack@opm-1:/opt/stack/nova/nova$
stack@opm-1:/opt/stack/nova/nova$ nova --os-username user1 --os-password ***** --os-tenant-name test
keypair-add test3 >test3.pem
ERROR (Forbidden):  Policy doesn't allow [compute_extension:keypairs:create] to be performed for role
[admin] due to user attribute (HTTP 403) (Request-ID: req-be5b53dc-e81b-4f23-8e15-724a6b29b5ee)
stack@opm-1:/opt/stack/nova/nova$ nova --os-username user4 --os-password ***** --os-tenant-name test
keypair-add test45 >test45.pem
stack@opm-1:/opt/stack/nova/nova$ nova --os-username user4 --os-password ***** --os-tenant-name test
keypair-list
+--------+-------------------------------------------------+
| Name   | Fingerprint                                     |
+--------+-------------------------------------------------+
| test4  |************************************************ |
| test41 |************************************************ |
| test42 |************************************************ |
| test43 |************************************************ |
| test44 |************************************************ |
| test45 |************************************************ |
+--------+-------------------------------------------------+
stack@opm-1:/opt/stack/nova/nova$
```

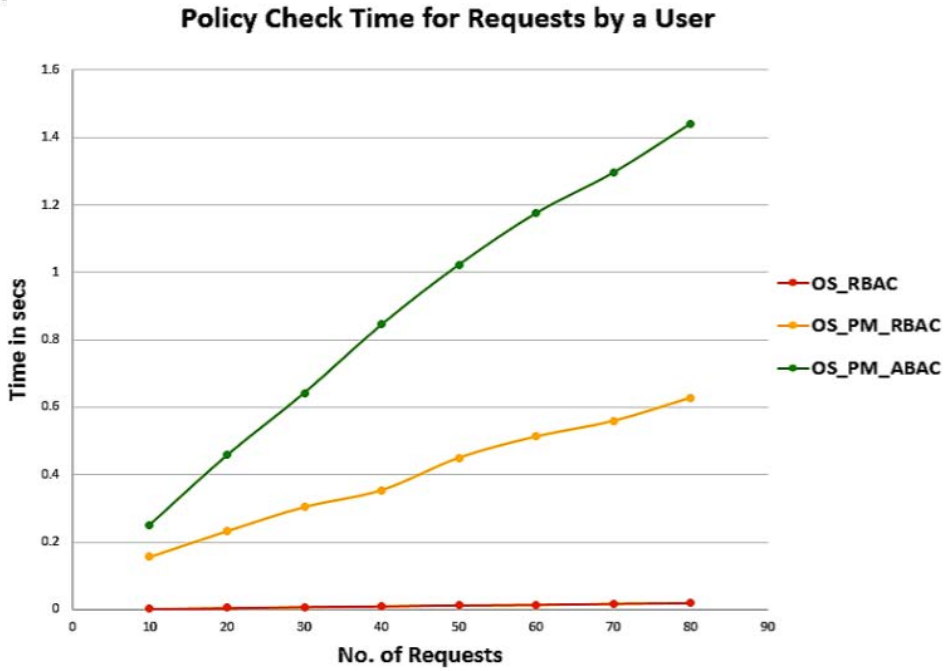
Role = Admin and
Department = IT

OpenStack Enforcement Results

Evaluation



Indications of cost of implementing our ABAC extension in OpenStack using PM



Network latency
Proof-of-concept implementation
Need for Optimization

Discussion and Analysis

- Advantages of an ABAC extension with user attributes:
  - define more fine-grained access control policies
  - significantly reduce number of roles required in a policy
  - avoid problems such as role explosion and role-permission explosion
- Trade-off between performance and enhanced functionality/capability
- Performance improvement techniques:
  - high-performance server to host PM and AE to improve policy evaluation time
  - cache policy evaluation results locally
  - Install PM, AE, and OpenStack services on an isolated subnet

Conclusion and Future Work

- Proposed an ABAC extension with user attributes for OpenStack
- Enforced our model in OpenStack utilizing the PM and AE
- An initial attempt to facilitate transition towards ABAC models in real world applications
- **Future Work:**
  - Explore other capabilities of PM – combination of access control polices, attribute and role hierarchy
  - Apply performance enhancements to enforcement framework
  - Include object attributes

Thank you!!

Questions??