Risk-Aware Role and Attribute Based Access Control Models

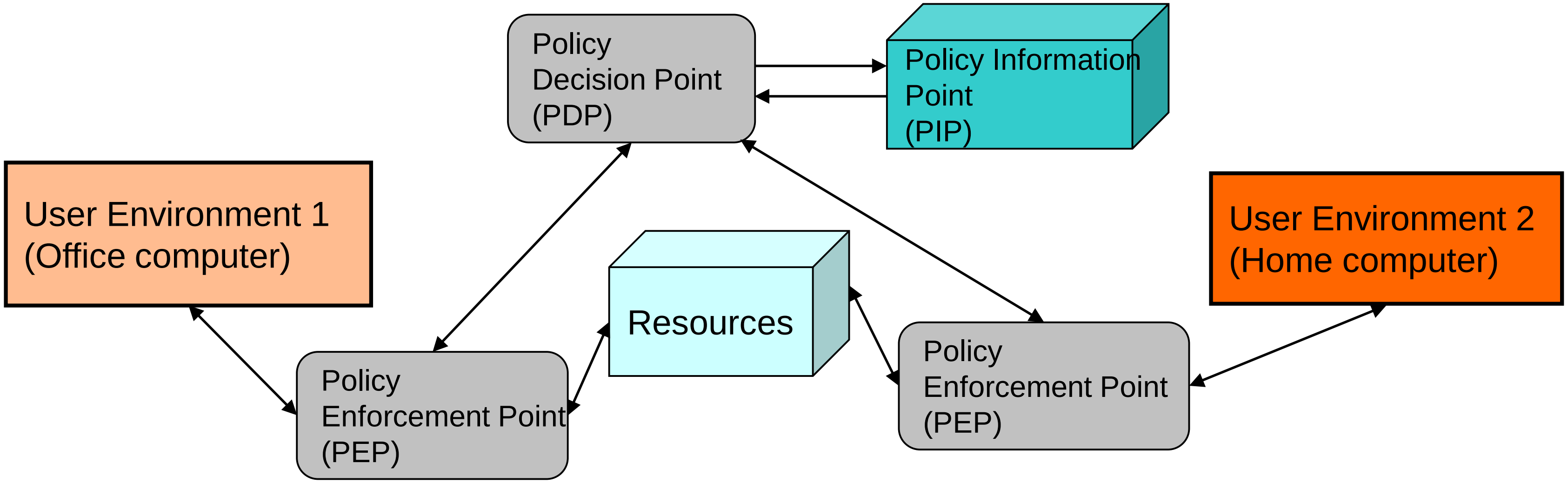
Khalid Zaman Bijon

Intro. & Motivation

Overall Strategy

- Risk-Awareness in Access Control Systems
  - Quantified Approach (Risk is represented as a metric)
  - Calculate risk value, involved in every situation
  - Grant access accordingly based on the estimated risk value

Motivational Scenario


Policy Decision Point (PDP)
Policy Information Point (PIP)
User Environment 1 (Office computer)
Resources
User Environment 2 (Home computer)
Policy Enforcement Point (PEP)
Policy Enforcement Point (PEP)

A simple PDP/PEP based Access Control Enforcement Model

Intro. & Motivation (cont.)

Scope

- A risk-aware access control system should have following two properties[1]:
  - Proper risk-estimation technique suitable to a particular context
  - Appropriate mechanism to utilize risk for access decision making
- Risk-estimation is context dependent
  - Out of scope of my research
  - Several approaches are already proposed (E. Celikel et al (2009), F.Salim et al (2011), L. Chen et al (2011), N. Baracaldo et al (2011) , Q Ni (2010) , H. Khambhammettu et al (2013))
- Focused on Risk-utilization process
  - How estimated risk can influence decision making process
  - Assume risk is somehow computed and readily available

[1] MITRE Corporation Jason Program Office. Horizontal integration: Broader access models for realizing information dominance. *Technical Report JSR-04-132, MITRE Corporation*, 2004

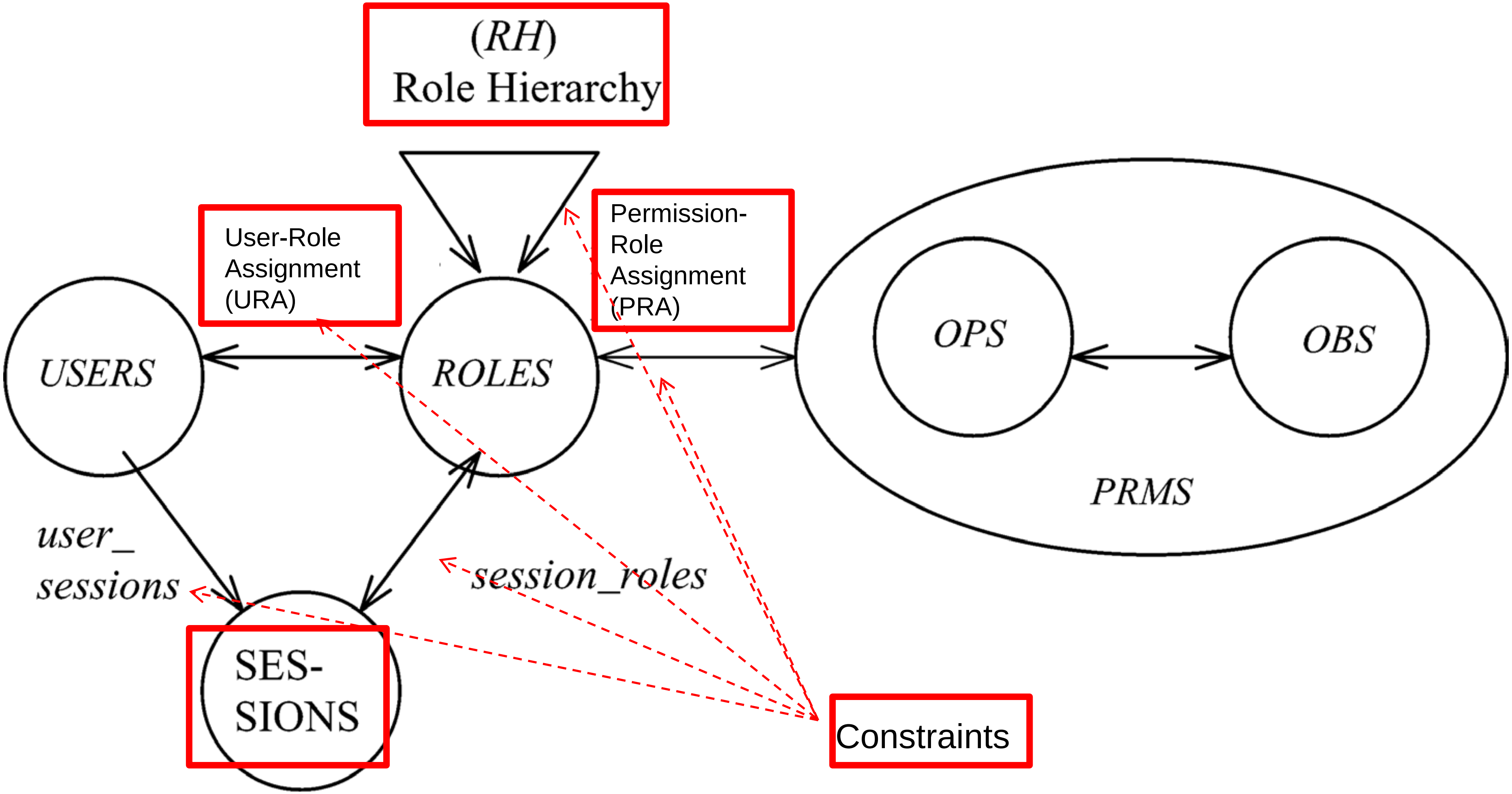
Problems To Solve

- What should it take to make a RBAC system risk-aware?
  - Identify the components that could be risk-aware.
  - Identify the risk-awareness types, if any.
  - How a particular type of risk-awareness affects the present functionalities of a risk-aware component?
  - What additional functionalities that component requires for that risk-awareness?
  - In conventional RBAC, is there any risk-awareness?
  - What are the differences and boundary between quantified and traditional approaches?
  - Overall, A proper guideline to develop a risk-awareness around present RBAC system in order to provide dynamism in decision making process
- Similar problems need to be addressed for a risk-aware ABAC system.

The Framework

- Identify the Risk-Aware RBAC Components
  - Faces different types of security risk while performing their operations
  - Need to develop additional functionalities to support a risk-awareness
- Different Types of Risk-Awareness
  - Traditional Approaches
  - Quantified Approaches
    - Non-adaptive approach
    - Adaptive approach

Risk-Aware RBAC Components

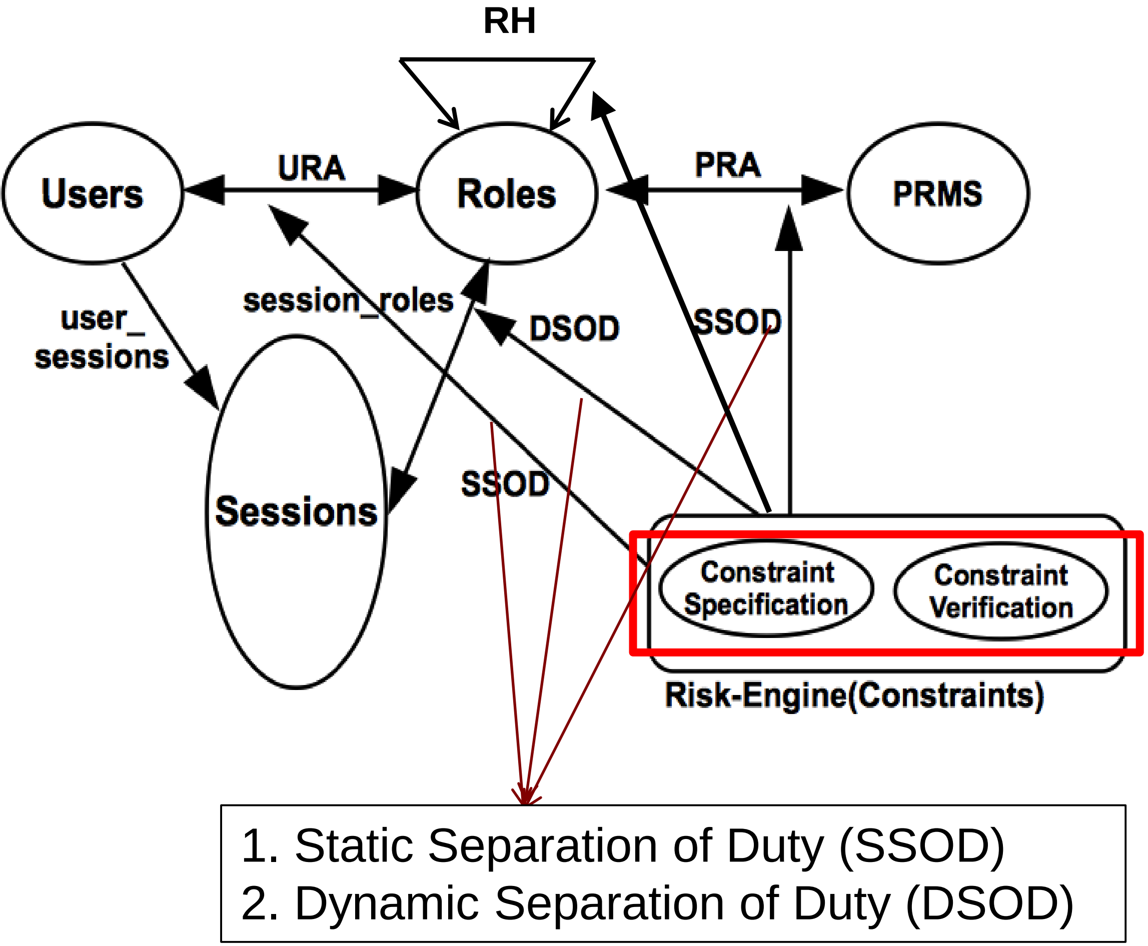


Risk-Awareness Types

- Traditional Approaches
  - Constraints driven risk mitigation
  - No explicit notion of risk value
- Quantified Approaches
  - Risk is explicitly represented as a metric
  - Risk is mitigated based on the estimated value

Traditional Risk-Awareness

1. Administrative user needs to identify risky operations and generate constraints accordingly. (For example, a constraints can restrict two risky roles from assigning to same user (SSOD).
2. Static in nature (a constraint always gives same outcome, unless modified)

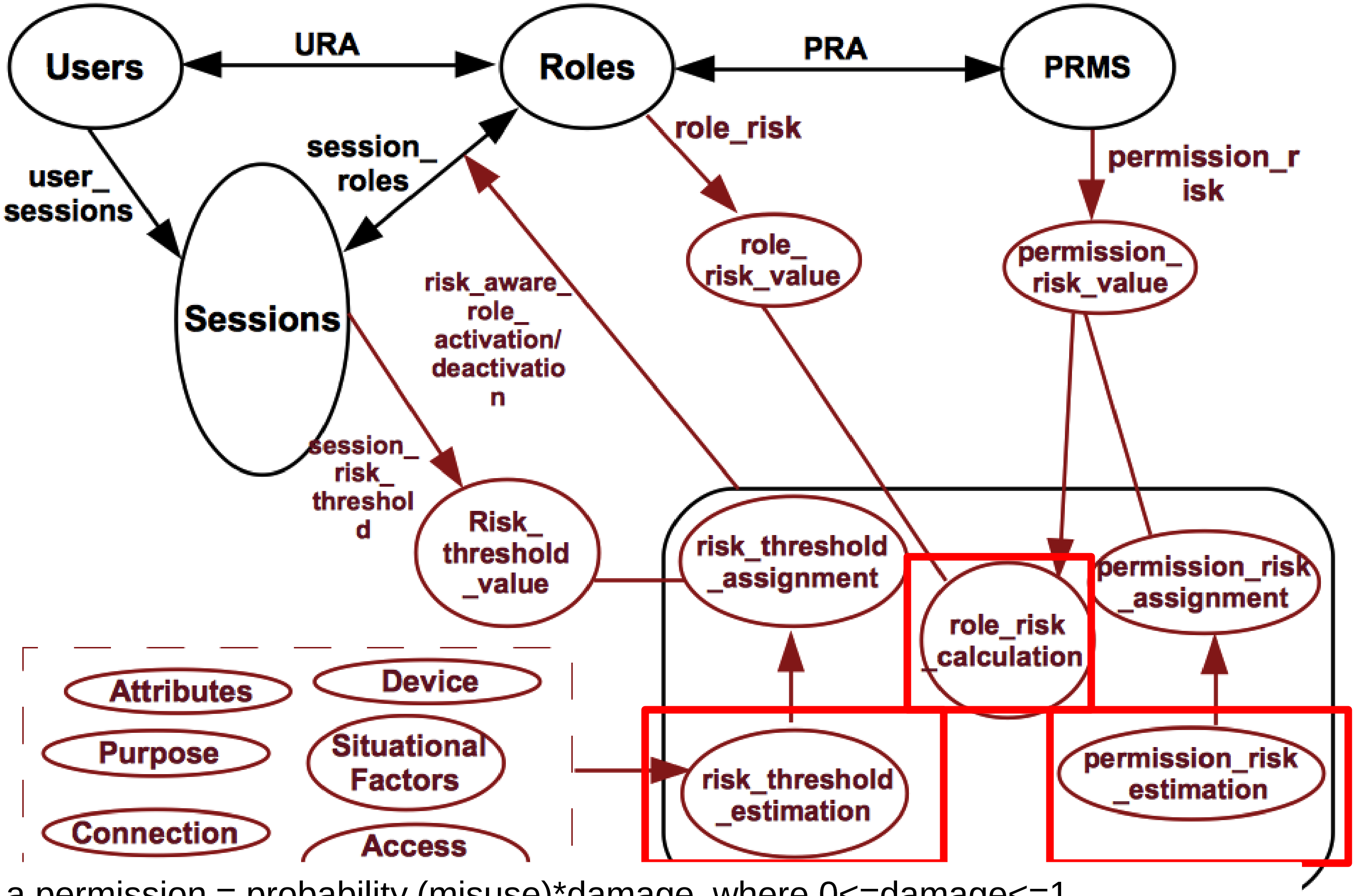


1. Static Separation of Duty (SSOD)
2. Dynamic Separation of Duty (DSOD)

Quantified Risk-Awareness
(Non-Adaptive)

1. Risk-threshold should vary across sessions (e.g. a session from office vs. session from home pc)

2. Risk-threshold limits user activities by restricting role-activation



An Example: Risk of a permission = probability (misuse)*damage, where 0<=damage<=1
Risk of a role = ∑Risk of assigned permission/number of assigned permissions
Now, Risk of permission p1 = 0.5 and p2 = 0.7.
Risk(r1) = 0.6, p1 and p2 assigned to r1.
Lets say, a session s1 risk threshold value is = 0.55. Hence, r1 can not be activate in s1.

A Framework for Risk-Aware RBAC

- To Summarize the framework:
  - The Risk-Aware RBAC Components are indentified
    - Sessions, User-Role assignments, Permission-Role assignments, Role Hierarchy, Constraints
    - Each components have different functionalities (need to be developed to support a Risk-Awareness)
  - Different Types of Risk-Awareness Approaches
    - Traditional Approaches
      - Constraints specific (implicit risk and static in nature)
    - Quantified Approaches
      - Non-adaptive approach (explicit notion of risk that varies across different situations)
      - Adaptive approach (need run-time monitoring capabilities and additional system functions for automatic response)

RBAC Sessions

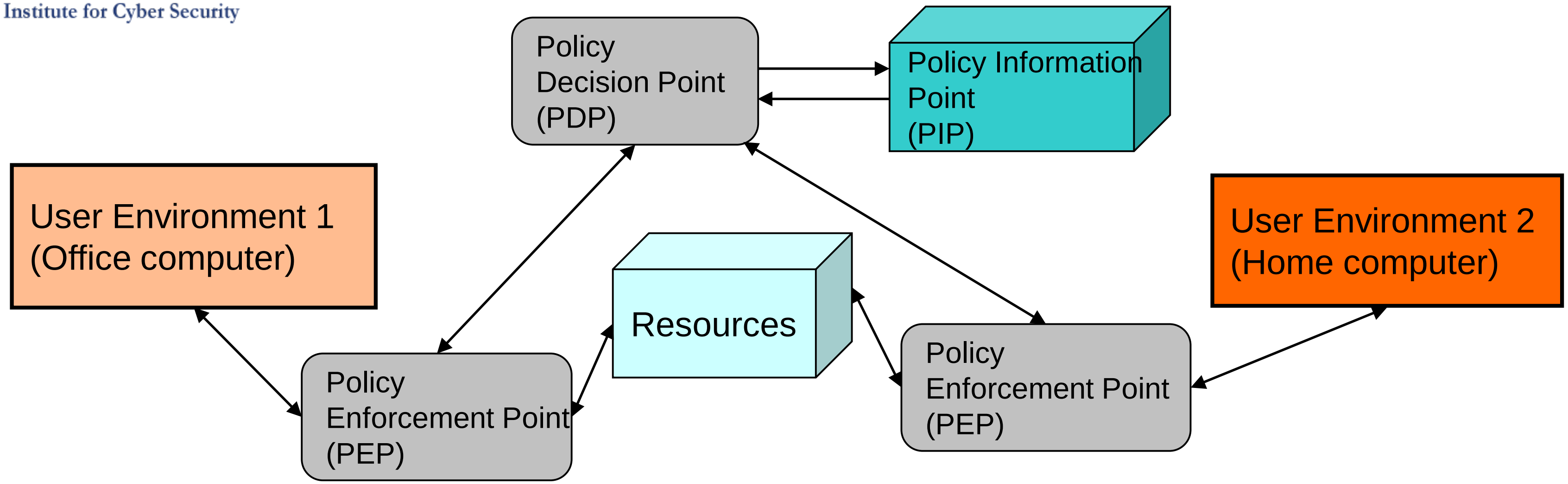
- Motivation for Session in Classical RBAC
  - Least Privilege
  - Dynamic Separation of Duty (DSOD)

- Functionalities:
  - Role Activation: Activate a role **(Increase the session's access capability)**
  - Role Deactivation: Deactivate a role **(Decrease the session's access capability)**

Concern:

1. User's complete discretion on activation and deactivation
2. No differentiation of sessions

Motivational Scenario


Policy Decision Point (PDP)
Policy Information Point (PIP)
User Environment 1 (Office computer)
User Environment 2 (Home computer)
Resources
Policy Enforcement Point (PEP)
Policy Enforcement Point (PEP)

A simple PDP/PEP based Access Control Enforcement Model

- Environment 2 might be less secure than Environment 1
  - Thus, user sessions from them should not be equally secure
- A user session can also be compromised
  - E.g. by malware running in user's computer (environment)
- Attacker could completely impersonate the user in a compromised session
  - Activating all the roles assigned to the user **(role activation is entirely at user's discretion in every session)**

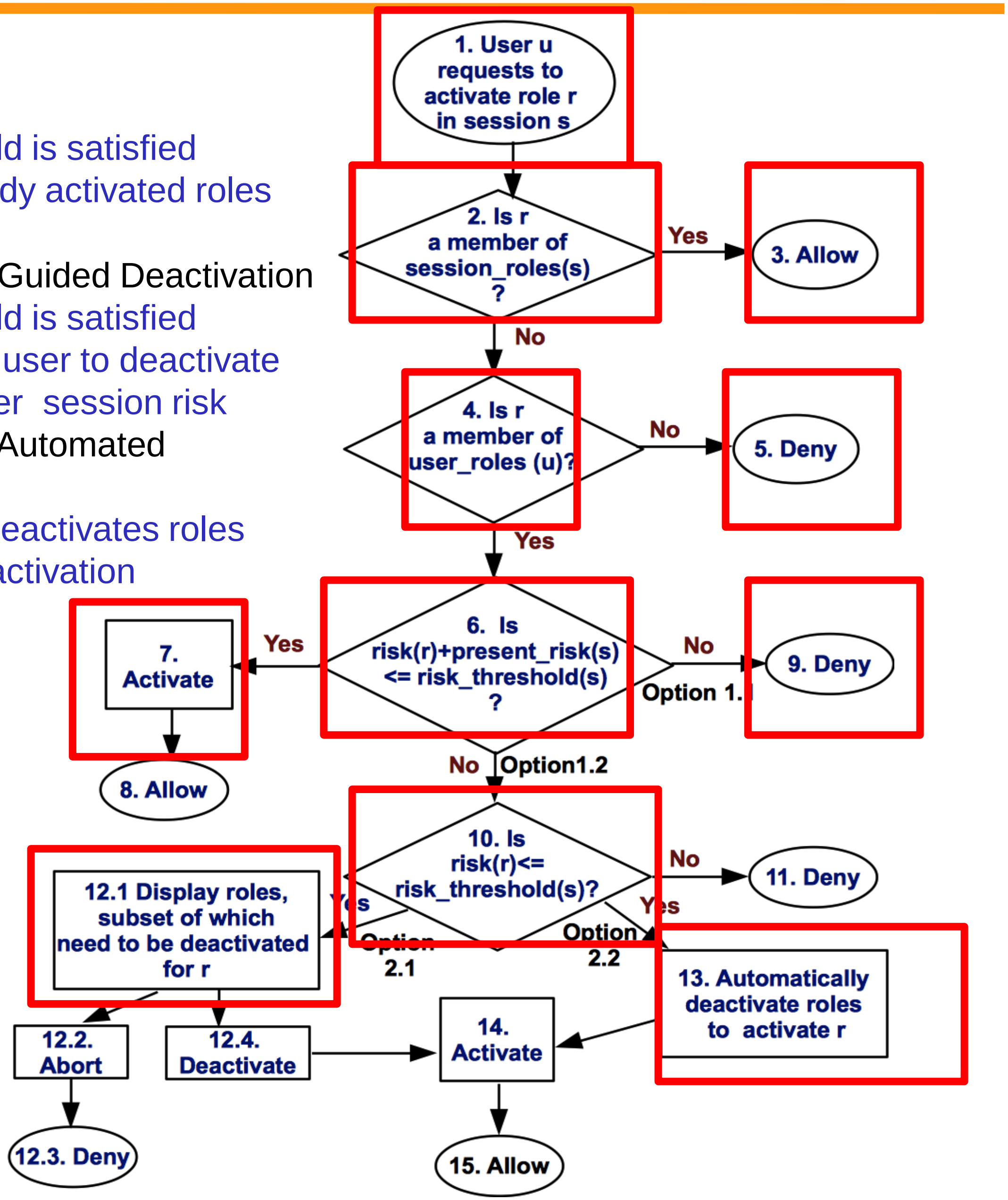
World-Leading Research with Real-World Impact!

Mitigation Strategy

- A procedure to identify how risky a session is
  - risk-estimation of a session
- Limit session's access capability based on its estimated risk
  - a risk-threshold restricts certain roles activation
  - session risk-threshold vs. combined risk of activated roles
- Reduce User's discretion on Role activation and deactivation
  - involve system to select a role to activate or deactivate

Role Activation Framework (Role-Level Interaction)

- Strict Activation
  - activates if risk-threshold is satisfied
  - no deactivation of already activated roles from session
- Activation with System Guided Deactivation
  - activates if risk-threshold is satisfied
  - if not, system suggests user to deactivate certain activated roles to lower session risk
- Activation with System Automated Deactivation
  - system automatically deactivates roles
  - need a specific role deactivation
  - algorithm (e.g. LRU, heuristics)


1. User u requests to activate role r in session s
2. Is r a member of session_roles(s) ?
Yes
3. Allow
No
4. Is r a member of user_roles (u)?
No
5. Deny
Yes
6. Is risk(r)+present_risk(s) <= risk_threshold(s) ?
Yes
7. Activate
8. Allow
No
Option 1.1
9. Deny
No
Option1.2
10. Is risk(r)<= risk_threshold(s)?
No
11. Deny
Yes
Option 2.1
Yes
Option 2.2
12.1 Display roles, subset of which need to be deactivated for r
12.2. Abort
12.3. Deny
12.4. Deactivate
13. Automatically deactivate roles to activate r
14. Activate
15. Allow

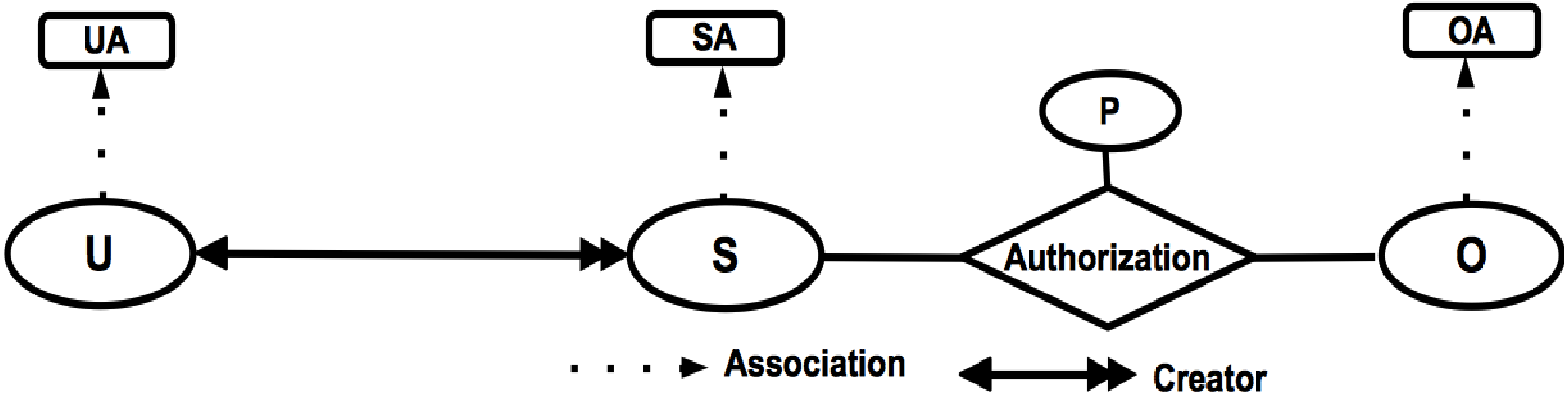
World-Leading Research with Real-World Impact!

Constraint Specification in ABAC

- Traditional Risk-Awareness in RBAC has a very rich literature
  - A role based constraint specification language (RCL-2000)
  - Can specify several SSOD, DSOD and other constraints in RBAC

- There is no such constraints specification process in ABAC
  - Except ABACα (2012) limitedly addressed some constraints on ABAC

Overview of an ABAC Model



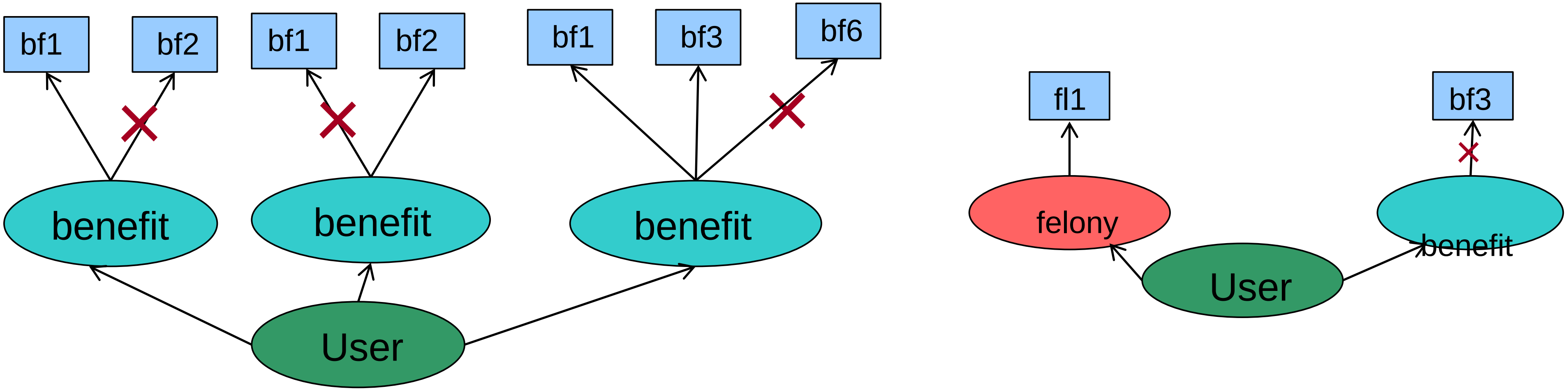
- User (U), Subject (S) and Object (O) are associate with a set of attributes UA, SA and OA respectively.
- An attribute is a key:value pair. For example, *role* is an attribute and the value of role could be {‘president’, ‘vice-president’, ‘manager’, etc. }
- An attribute can be set-valued or atomic.
  - Clearance vs. Role
- A User needs to create a subject to exercise privileges in the system.
- Each permission is associated with an authorization policy that verifies necessary subject and object attributes for authorization.

Motivation

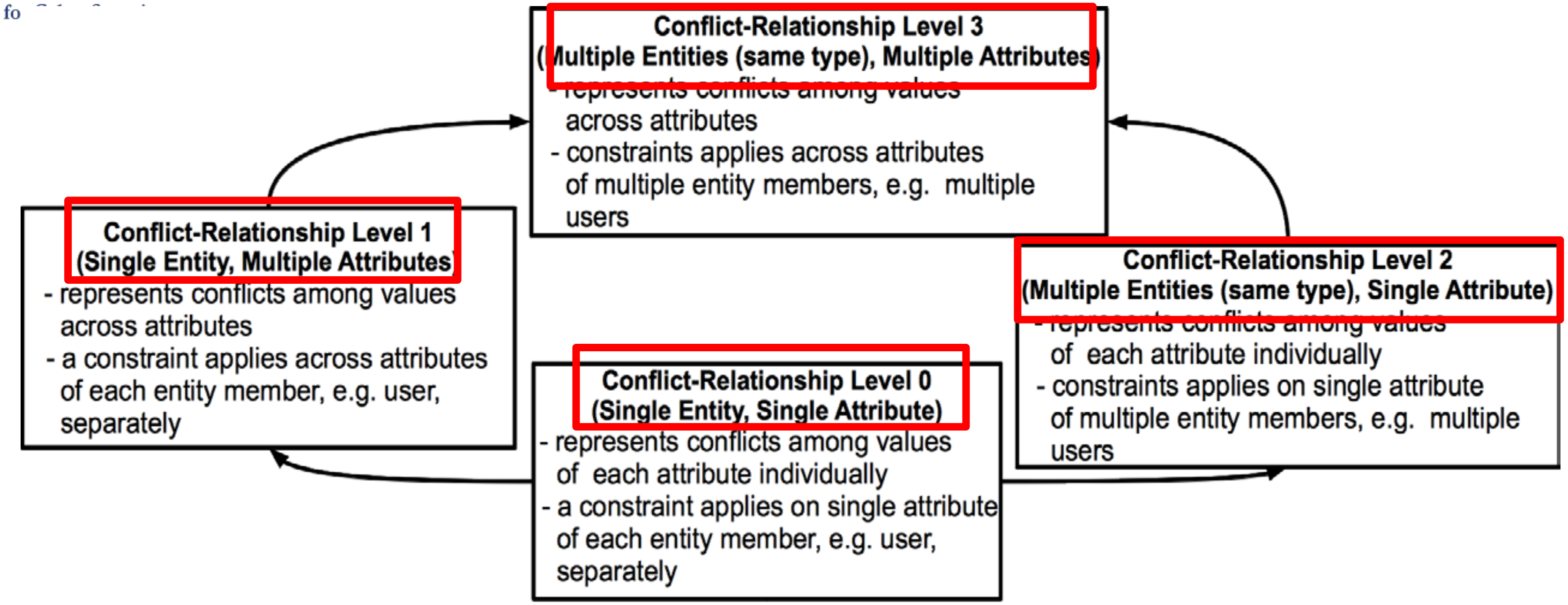
- ABAC is famous for its policy neutral and dynamic decision making capability
  - Authorization decision of each permission are made by comparing respective attributes of the involved subjects and objects
  - A subject with required attribute can access to an object
- Security policies are necessary to assign attributes to right entities (user, subject, etc.) for avoiding unauthorized access
  - Similar to correct role assignment to users in RBAC
- Proper constraints specification process can configure required security policies of an organization

Attribute Based Constraints Specification Language (ABCL)

- Develop an attribute based constraints specification language (ABCL)
    - Identify that attributes preserve different types of conflict-relationship with each other such as mutual exclusion, precondition, etc.
    - A particular conflict-relation restricts an entity to get certain values of an attribute.
        - *Benefit* attribute represents customers' assigned benefits in a Bank
        - A customer cannot get both *benefits* 'bf1' and 'bf2' (mutual exclusion)
        - Cannot get more than 3 benefits from 'bf1', 'bf3' and 'bf6' (cardinality on mutual exclusion)



Attribute Based Constraints Specification Language (ABCL) (cont.)



Conflict-Relationship Level 3
(Multiple Entities (same type), Multiple Attributes)
- represents conflicts among values across attributes
- constraints applies across attributes of multiple entity members, e.g. multiple users

Conflict-Relationship Level 1
(Single Entity, Multiple Attributes)
- represents conflicts among values across attributes
- a constraint applies across attributes of each entity member, e.g. user, separately

Conflict-Relationship Level 2
(Multiple Entities (same type), Single Attribute)
- represents conflicts among values of each attribute individually
- constraints applies on single attribute of multiple entity members, e.g. multiple users

Conflict-Relationship Level 0
(Single Entity, Single Attribute)
- represents conflicts among values of each attribute individually
- a constraint applies on single attribute of each entity member, e.g. user, separately

- A constraint can be applied to each entity (one user) independently or across entities (multiple users)
  - *Benefits* ‘bf1’ cannot be assigned to more than 10 users.
- Hierarchical classification of the attribute conflict-relationships
  - Number of attributes and number of entities allowed in a conflict relations

Attribute Based Constraints Specification Language (ABCL) (cont.)

- A mechanism to represent different types of such relationships as a set
  - Mutual-Exclusive relation of the *benefit* attribute values (single attribute conflict)

Attribute_Set$_{U,benefit}$ *UMEBenefit*

UMEBenefit={avset1, avset2} where

avset1=({'bf1','bf2'}, 1) and

avset2=({'bf1','bf3','bf4'}, 2)

  - Mutual-Exclusive relation of the *benefit and felony* (cross attribute conflict)

Cross_Attribute_Set$_{U,Aattset,Rattset}$ *UMECFB*

Here, *Aattset*= {felony} and *Rattset*= {benefit}

UMECFB={attfun1} where

attfun1(felony)=(attval, *limit)*

where *attval={'fl1',* 'fl2'} and *limit=1*

attfun1(benefit)=(*attval, limit)*

where *attval={'bf1'}* and *limit=0*

Attribute Based Constraints Specification Language (ABCL) (cont.)

- **Examples**

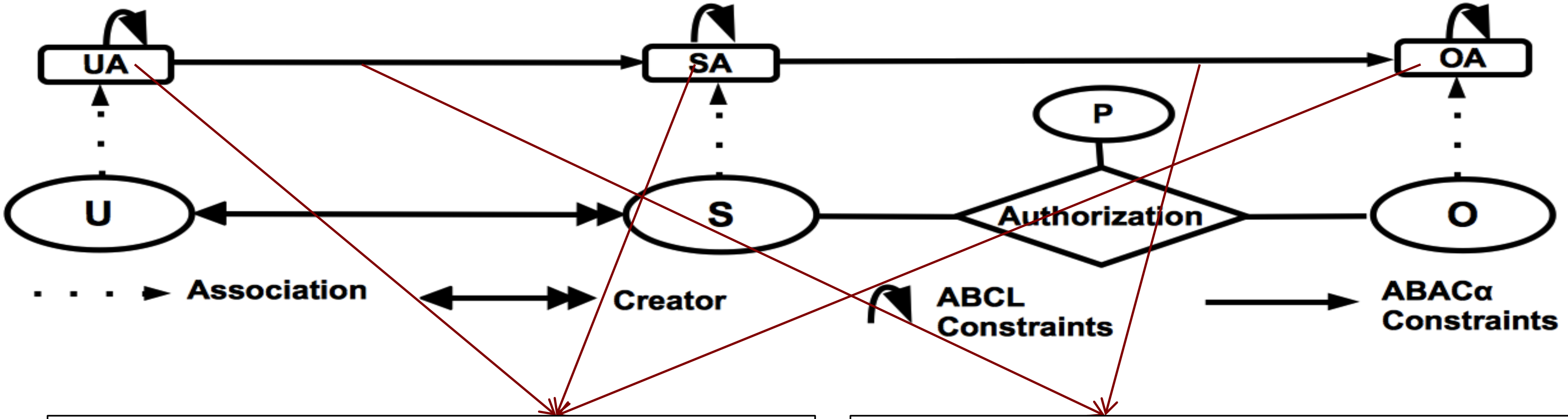
  - A customer cannot get both benefits ‘bf1’ and ‘bf2’

    Expression: $|OE(UMEBenefit).attset \cap benefit(OE(U))| \leq OE(UMEBenefit).limit$

  - A customer committed felony ‘fl1’, She can not get more than one benefit from ‘bf1’, ‘bf2’ and ‘bf3’

    Expression: $OE(UMECFB)(felony).attset \cap felony(OE(U))| \geq OE(UMECFB)(felony).limit$
    $\Rightarrow |OE(UMECFB)(benefit).attset \cap benefit(OE(U))| \leq OE(UMECFB)(benefit).limit$

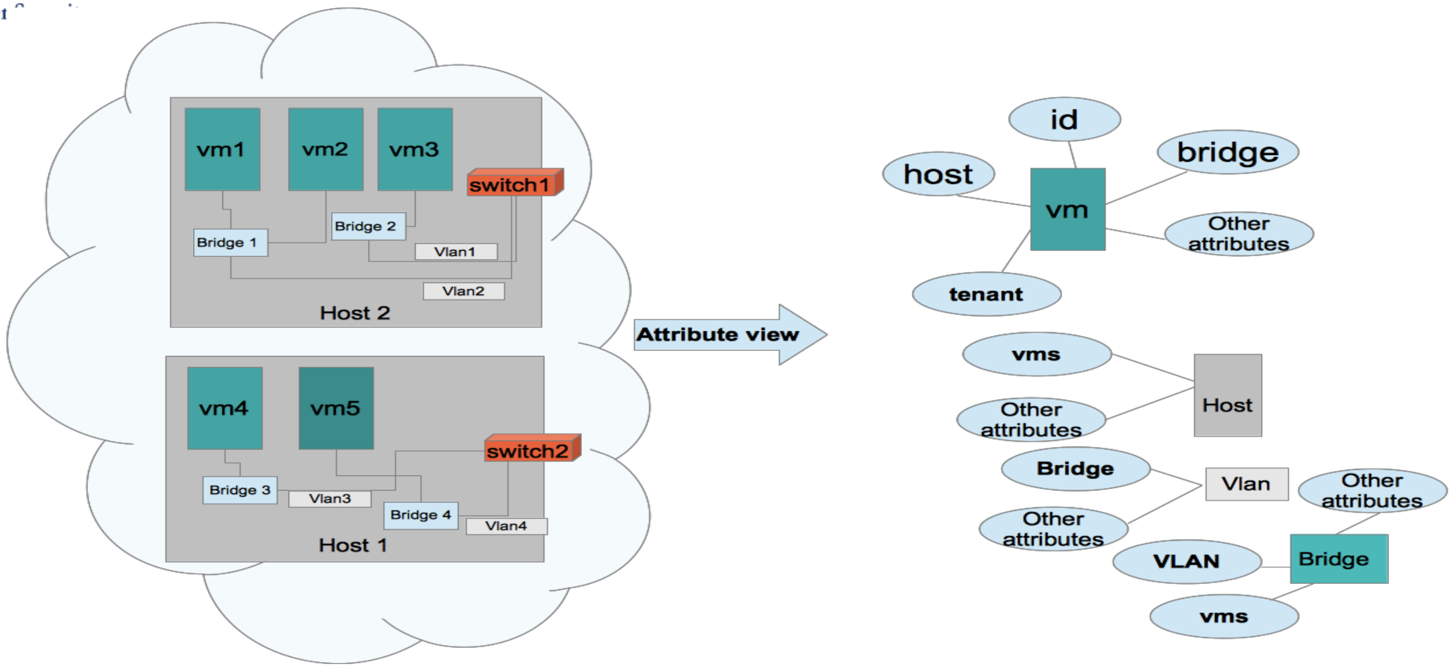
Attribute Based Constraints Specification Language (ABCL) (cont.)



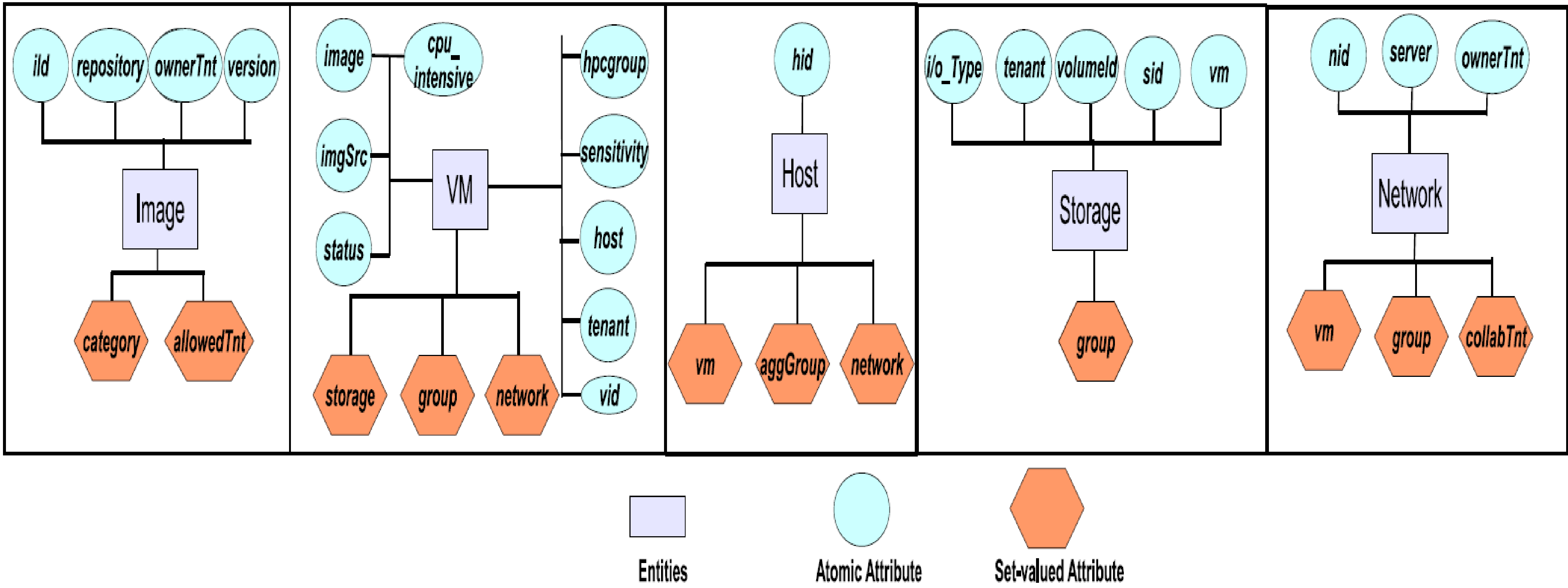
What value of an attribute of user, subject or object can get based on specific relationships with other attribute values of the user, subject and object

Attributes a subject can get from its user or an object can get from the subject (much like what roles a session can activate from the user's roles)

Customized ABCL for Cloud IaaS



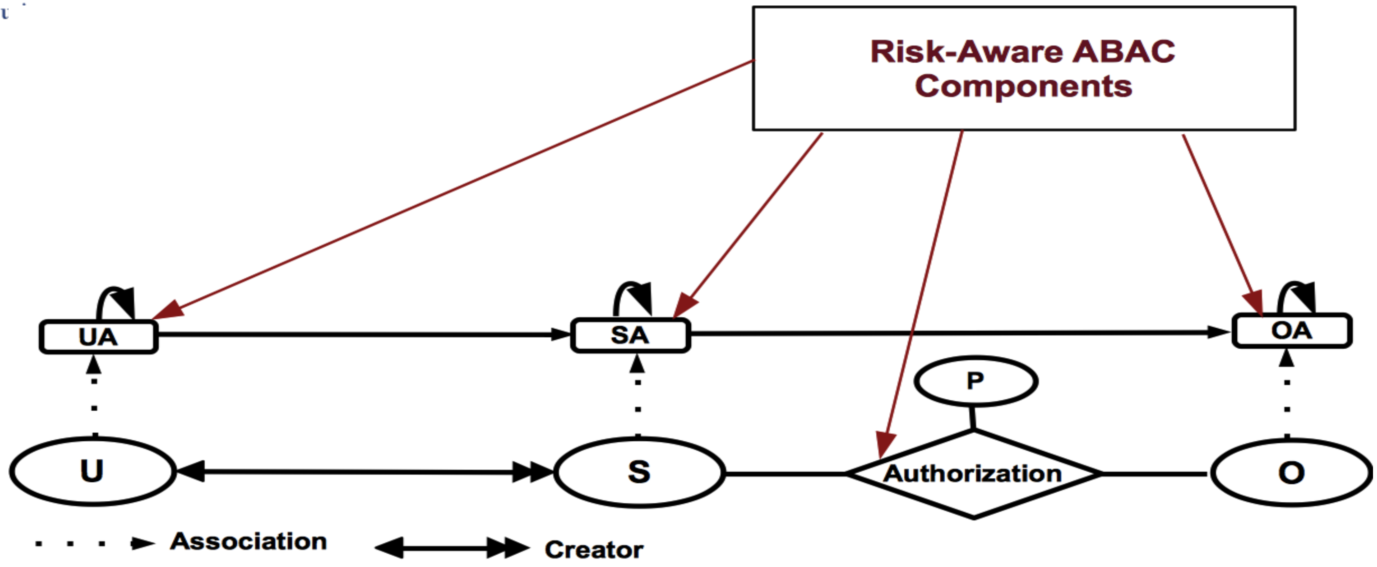
Customized ABCL for Cloud IaaS



Customized ABCL for Cloud IaaS

- Components of the cloud IaaS have different properties or attributes
  - For instance, a VM can have attributes host, tenant, id, bridge, required_comp_power, etc.
- A customized ABCL can restrict certain attributes assignment to a VM
  - If two VMs are from conflicting tenants, they cannot be located in same host
  - Two high hpc VMs cannot be located in same host

Open Discussion



- Developed Traditional Risk-Awareness (ABCL) for ABAC
- Identified Risk-Aware ABAC Components
  - Authorization component, User attribute assignment (UAA), SAA, OAA
- Issues
  - Presently only one authorization policy for each permission (for every risky situation)
  - Depending on risk involved in current situation certain attributes may not be assigned to certain entities

The End

Questions?