Identity Delegation in Policy Based Systems

Rajeev Gupta, Shourya Roy and Manish Bhide
IBM Research
India Research Lab

Presented by: Shourya Roy <rshourya@in.ibm.com>

Introduction

- Policies are rules governing the choices in the behaviour of a system. [Sloman]
- Systems are managed by dynamically adapting themselves in accordance with policies.
- Policy execution requires suitable authenticated identity on the target managed resource
  - In this dynamically adapting environment how can one be sure that policy will be executed successfully?
- Managed system has its own access control, how to dynamically identify the '**most suitable**' identity to execute the policy?

Example

- Business head of *YouGetEveryThingHere* supermarket chain decides on a policy
  "*All preferred customers should be given 10% Christmas discount*"
- Sales department head Mr. Smith converts the policy into an XML like format and determines that the policy should be executed on all systems dealing with ***billing***.
- This policy is distributed to all the branches of the chain. Each branch would have a bunch of system administrators who would have access rights on the billing systems.
- With whose credentials will the policy finally execute?
  - More critical, how to know this at the time of policy definition!!

People involved in policy lifecycle…

Policies are acted upon by three different kinds of entities

- **Policy Owner**
  - Decides high level goals of the policy
  - Ex. Business head of *YouGetEveryThingHere*
- **Domain Expert**
  - Technical head of the department
  - Writes the policy in machine understandable language
  - Ex. Sales department head Mr. Smith
- **Policy Enforcer**
  - Actually executes the policy
  - Ex. bunch of system administrators

Need to execute policies at various systems on behalf of policy owner as decided by domain expert

Need for Delegation

Identity delegation

- Delegation of access rights to enforce policy from Policy owner to Policy Enforcer
- Policy enforcer can be hard-coded while writing the policy, we call it, *explicit delegation*, but
  - Does policy author always know the name of policy enforcer?
  - Policies are written once, say for whole enterprise, and are executed on multiple systems.
  - System access rights (e.g. access control list) change making it difficult to have static policy enforcer.
  - Organizational responsibilities change, changing access rights of individual enforcers.

Automatically Determining Policy Enforcers Depending on Various Parameters

Implicit Identity Delegation

- Algorithm
  - To generate policy enforcer's identity at policy definition time
  - Multiple enforcers can be there for complex policies
  - Needs to be efficient
- Implementation
  - Inside resource manager
  - Plug-in based architecture

Guiding Principles

- Implicit Identity Delegation algorithm is governed by two intuitive principles
  - **Hierarchy Principle** – Concerning users of the system assuming hierarchical relationship between them
  - **Containment Principle** – Concerning objects in the system assuming containment relationship between them

Hierarchy principle

- Users lower in the hierarchy have better understanding of system details.
- Policy should be executed by lowest person in the organization hierarchy having sufficient access rights.

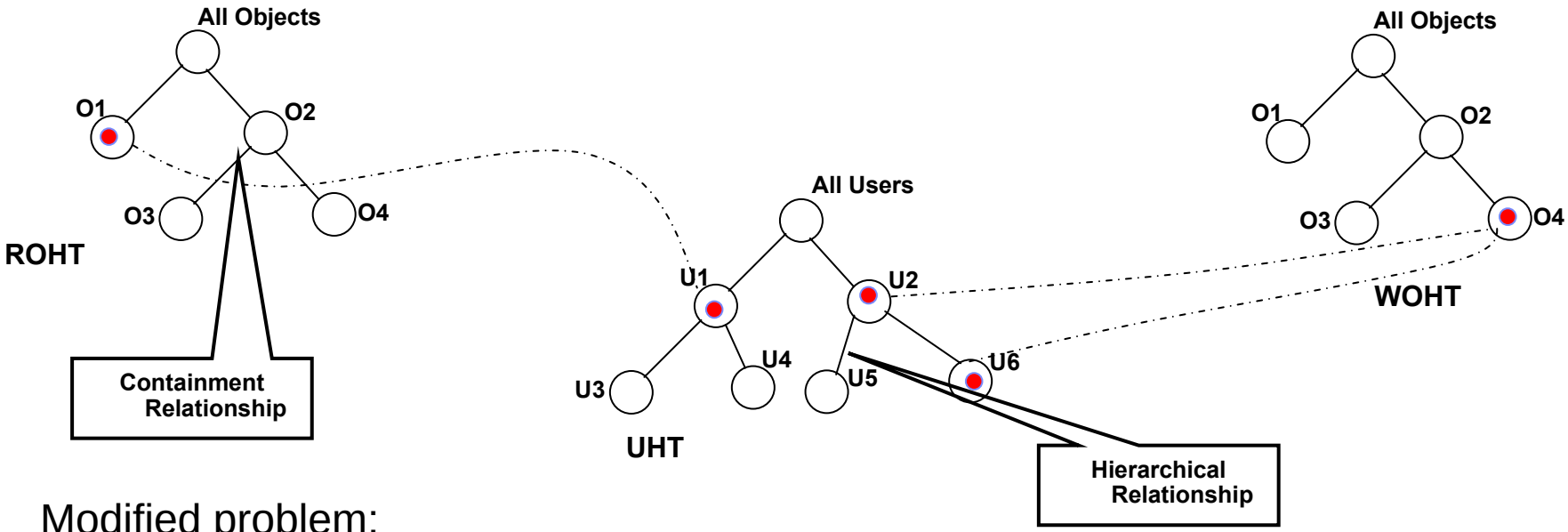
Containment principle

- Access rights are inherited from a system object to its sub-objects.
- If a user u_1 has access over a database D_1 then it, by default, also has access over all the tables of D_1.
- Another user u_2 has access over a table T_1 of the database D_1
  - Policy requires access over the table T_1 .
  - Policy should be executed with u_2's credentials rather than u_1.

Problem statement

Find the set of users who have sufficient access rights to execute the policy such that the hierarchy and containment principles are satisfied.

Illustration (assuming Read and Write access rights)



Modified problem:

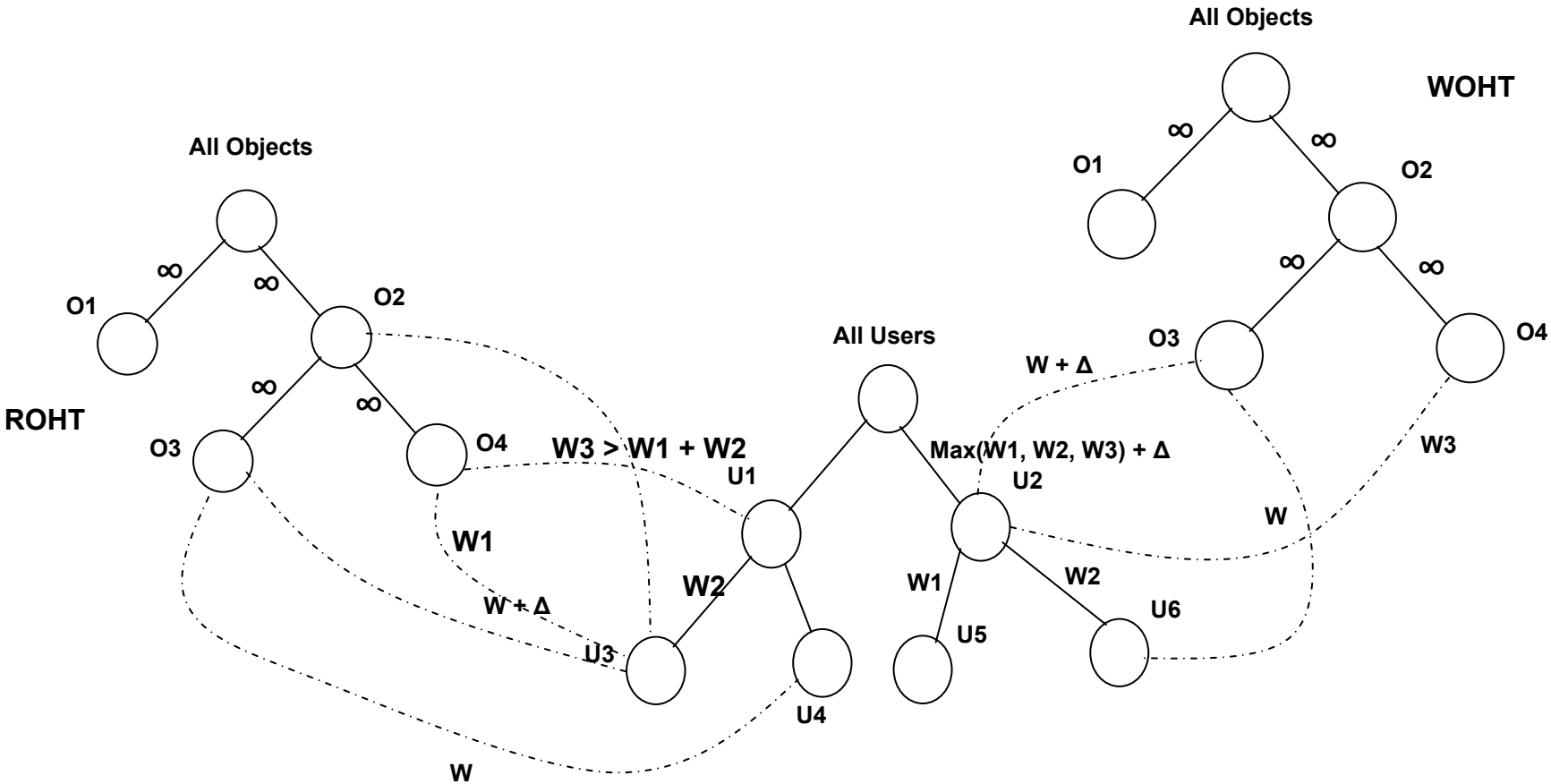
-- Find a tree that connects nodes O1 and O4 in ROHT and WOHT respectively

-- All the users adjacent to the OHTs are responsible for policy execution

Solution

- Multiple trees may exist connecting the set of nodes on which access is required
- Can we assign weights to the edges of the graph such that Sub-tree which satisfies the containment and hierarchy principles has the least weight?
- YES! IDA Weight assignment algorithm

Weight Assignment Rules



Rule 1: As per the hierarchy principal, there should be a penalty for traversing the higher in the hierarchy

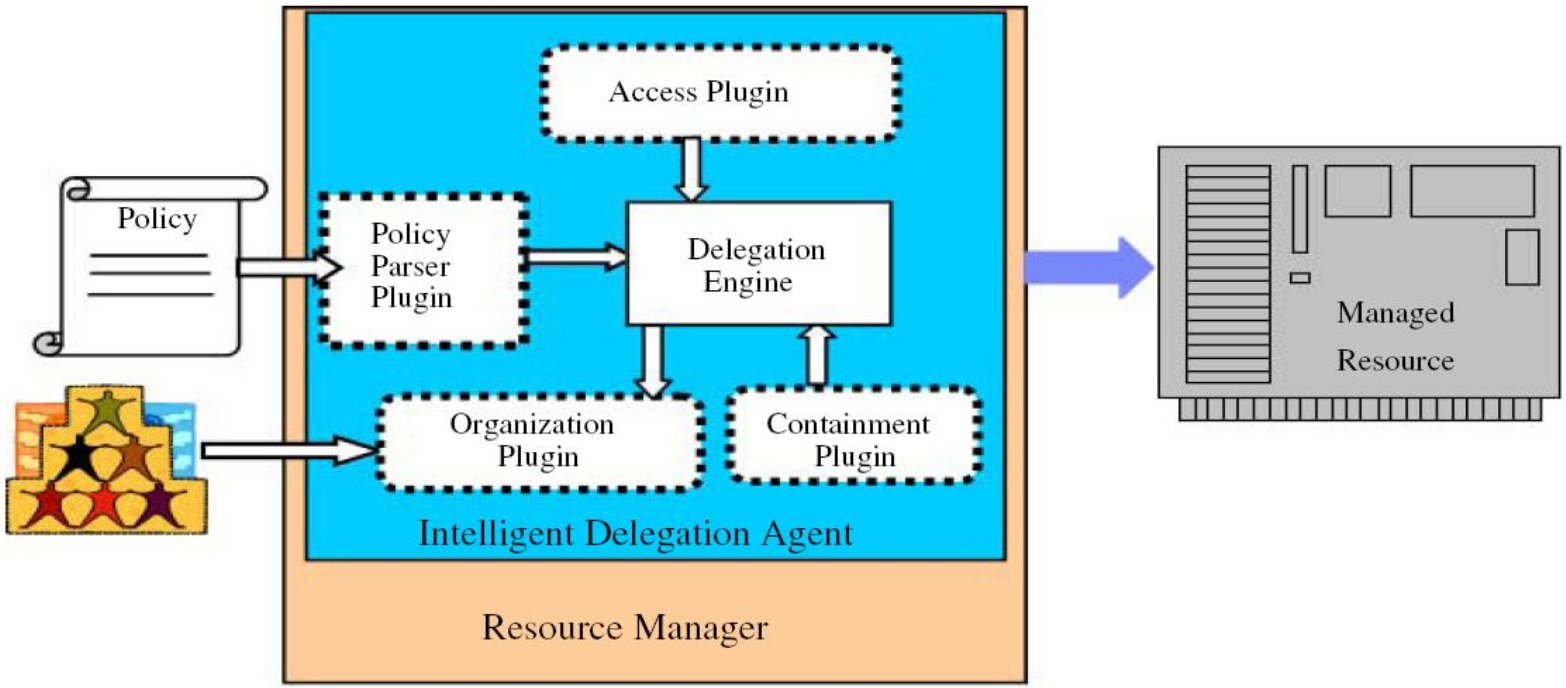
Rule 2: The policy edge directly connected to the user is preferred over an access edge which is not inherited

Minimization Problem

- Problem of finding policy executors mapped to finding the ***shortest*** path connecting a bunch of nodes in a graph
  - **Input**: A graph G=(V,E). A subset of vertices T ⊆ V.
  - **Problem**: Find the sub-tree connecting all the vertices of T having the least weight.
- This is the Steiner tree[1] problem
  - NP Hard
- We use a greedy heuristic to find the path
  - Very efficient in practice
  - Gives sufficiently good results

1: ***Find a minimum-weight tree connecting a designated set of vertices, called terminals, in an undirected, weighted graph or points in a space.***

Implicit Delegation Framework



Plug-in Based Framework

- Algorithm requires various information such as access rights in a system, relationship between users.
- The architecture is designed as loosely-coupled plugin based manner
  - Inputs are modeled as plug-ins which can applied to different domains
  - The plugins can be implemented as sensors on the managed resource

Plugins and Delegation Engine

Policy Parser Plugin

- Responsible for parsing a policy and extract relevant fields such as *scope*
- We have implemented a parser for a subset for Autonomic Computing Policy Language (ACPL)

Access Plugin

- Retrieves information about direct and inherited access rights of users present in a system
- We have implemented this plugin for DB2 using *systabauth* table which stores access rights of DB2 users

Containment Plugin

- Finds if an object o_1 contained in another object o_2
- We have implemented a simple containment plugin for filesystem

Organization Plugin

- Captures hierarchical relationship between users in the organization
- We used an LDAP client to extract information from the LDAP directory

Delegation Engine

- Constructs the UHT-OHT graph based on output from plugins and assigns edge weights as per the rules
- Once the policy enforcers have been found the delegation engine notifies the users and uses their credential if the user approves the use of their identity for executing the policy
- Typically done once at the time of system initiation and in case of changes in organization hierarchy or access rights etc.

Conclusion

- There is a need for Identity Delegation in Policy Management Infrastructure
- Difficult task to choose the policy enforcer identity manually
- Presented an ‘autonomic’ approach to Identity Delegation
  - Modeled the problem to that of finding a path in a graph
  - Novel weight assignment algorithm
- Novel plug-in based architecture and implementation

My co-authors!

Rajeev

Manish

Thank you very much for your attention

Questions?