Constraints Specification for Virtual Resource Orchestration in Cloud IaaS

Khalid Bijon, Ram Krishnan and Ravi Sandhu
University of Texas at San Antonio

5th ACM Conference on Data and Applications Security and Privacy (CODASPY 2015)
World-Leading Research with Real-World Impact!

Outline

- Introduction
- Motivation
- Goal
- Methodology
- Enforcement (in Cloud IaaS)
- Implementation (in OpenStack)
- Conclusion
Introduction

Three Different Mapping Types
- Shared Responsibility
- Only Consider Type-3 Mappings
- Complex Management Process

World-Leading Research with Real-World Impact!
Motivation

- Inefficient and Tedious Management Plane
  - Manual Identification
  - User Centric (*unnecessary indirection*)

- No Direct Misconfiguration Detection/Prevention
  - Elevate Security Vulnerability

**Figure 1**

Manual Detection

Solution??

Credit: www.iconarchive.com
www.consulting.ky
www.acm.icpc.org

World-Leading Research with Real-World Impact!
Goal

- Easily Manageable Type 3 Mapping
  - High-level Policy
  - Configure Diverse Requirements

- Establish Direct Relations
  - Keep Users Out of Loop

- Automatically Prevent Misconfiguration

World-Leading Research with Real-World Impact!
Methodology

Constraint Policy
- For Each Type-3 Mappings

Satisfied By
- Individual Virtual Resources

Figure 2

Figure 3
An Attribute Based Approach

- Attribute Specifies Virtual Resource Properties

- A name:value Pair

- Designed as Functions

World-Leading Research with Real-World Impact!
A Constraint

- Logical Formula
- Compares Certain Attribute Values

Simple but Powerful

- Hadoop Cluster
- 3-tier business application

Constraint Policy

Constraint Format

If tier=database

Then ioType='fast'

True False
UseCase (3-Tier System)

Constraint 1: If a VM is for presentation layer, attaching storage’s ioType cannot be fast.

Constraint 2: Only an application layer VM can connect to a virtual network which is created for passing application layer data.

Constraint 3: If a router is for connecting to out-side internet, only presentation layer network or web front network can connect to it.
Two Components
- Specifier and Enforcer

**Constraint Specifier**

1. Execute operation to configure two virtual resources
2. Retrieve attribute values of respective virtual resources
3. Retrieve constraints for requested operations
4. Evaluate

**Constraint Enforcer**

5. Allow/Deny
Implemented in OpenStack

Execution of "attribute-creation" operation

Similarly,

- Attribute-value specification
- Constraint Specification
- Attribute-value assignment

World-Leading Research with Real-World Impact!
## API Specification

**Rest API**

<table>
<thead>
<tr>
<th>Name</th>
<th>URL</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>att-create</td>
<td>/v2/{tenant_id}/attributes</td>
<td>POST</td>
</tr>
<tr>
<td>att-delete</td>
<td>/v2/{tenant_id}/attributes/{id}</td>
<td>DELETE</td>
</tr>
<tr>
<td>att-list</td>
<td>/v2/{tenant_id}/attributes</td>
<td>GET</td>
</tr>
<tr>
<td>att-value-set</td>
<td>/v2/{tenant_id}/scopes</td>
<td>POST</td>
</tr>
<tr>
<td>att-value-delete</td>
<td>/v2/{tenant_id}/scopes/{id}</td>
<td>DELETE</td>
</tr>
<tr>
<td>att-value-get</td>
<td>/v2/{tenant_id}/scopes/</td>
<td>GET</td>
</tr>
<tr>
<td>constraint-add</td>
<td>/v2/{tenant_id}/constraints</td>
<td>POST</td>
</tr>
<tr>
<td>constraint-delete</td>
<td>/v2/{tenant_id}/constraints/{id}</td>
<td>DELETE</td>
</tr>
<tr>
<td>constraint-get</td>
<td>/v2/{tenant_id}/constraints</td>
<td>GET</td>
</tr>
<tr>
<td>meta</td>
<td>/v2/{tenant_id}/servers/</td>
<td>POST</td>
</tr>
</tbody>
</table>

**Attribute Name**

**Attribute Value**

**Attribute Value Assignment**

World-Leading
Validation Check

- Validity of Attribute Name and Value
- Tenant-Specific Attribute
- System or Inter-Tenant Attribute
Enforcer Implementation

- Implemented in OpenStack
- A Constraint Parser
- Invoked by Resource Mapping Operations (e.g., volume-attach)

1. Get User Token
2. Request volume-attach with VM Id, Storage Id and Token
3. Token Revoked?
4 5 7 8 9
10. Evaluate Constraint
11. Allow/Deny
6. Verify Project of VM, Storage and User

World-Leading Research with Real-World Impact!
Automated Constraint Construction

- Helps the tenants to find policy
- From Previous Configurations

Construct Relation between values of two attributes

Figure 7

Figure 8
Approach

- **Frequent-ItemSet Mining**
  - Apriori Algorithm
  - with customization for IaaS (CVRM-Apriori)
Policy for VM-Network Connectivity Mapping

From VM-Network Table (table `virtual_interfaces` in Nova, OpenStack)

- 10 Attributes each with 10 values
- 10 Virtual Networks
- At least three Networks per VM
- Mine relations between every two pair of attribute values
Conclusion

- A Constraint Specification Framework
- Easily manageable and generic
- Can be applied for Misconfiguration Prevention
- Also, for detection (flag-generator)
- Automatic Generation of Constraints

- Flag Generator System
- Semantic meaning of mined Attribute Relation
- Improve mining (incorporate noise)

World-Leading Research with Real-World Impact!
\[\begin{align*}
\text{<Quantifier>} & : = \forall (v_{r1}, v_{r2}) \in R_{<\text{Cls}>},<\text{Cls}> \cdot <\text{Stmt}> \\
\text{<Stmt>} & : = <\text{Stmt}> <\text{connector}> <\text{Stmt}> \mid (<\text{rule}>)
\end{align*}\]

\[\begin{align*}
\text{<rule>} & : = \text{<Token>} \rightarrow \text{<Token>}
\text{<Token>} & : = (<\text{Token}> <\text{connector}> <\text{Token}>)|(<\text{Term}>)
\text{<Term>} & : = <\text{Attribute}>(<\text{resource}>)<\text{comperator}> <\text{Scope}>
\text{<Attribute>} & : = <\text{letter}> \mid <\text{digit}> \mid <\text{Attribute}>
\text{<Scope>} & : = <\text{letter}> \mid <\text{digit}> <\text{Scope}>
\text{<connector>} & : = \land \mid \lor
\text{<comperator>} & : = = \mid \neq
\text{<Cls>} & : = c_1 \mid c_2 \mid \ldots \mid c_n
\text{<resource>} & : = v_{r1} \mid v_{r2}
\text{<digit>} & : = 0|1|2\ldots|8|9
\text{<letter>} & : = a|b|\ldots|y|z|A|B\ldots|Y|Z
\end{align*}\]
<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Integer</td>
</tr>
<tr>
<td>name</td>
<td>String(255)</td>
</tr>
<tr>
<td>project_id</td>
<td>String(255)</td>
</tr>
</tbody>
</table>

(a) The attribute Table

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Integer</td>
</tr>
<tr>
<td>name</td>
<td>String(255)</td>
</tr>
<tr>
<td>value</td>
<td>String(255)</td>
</tr>
<tr>
<td>project_id</td>
<td>String(255)</td>
</tr>
</tbody>
</table>

(a) The scope Table

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Integer</td>
</tr>
<tr>
<td>relation_name</td>
<td>String(255)</td>
</tr>
<tr>
<td>expression</td>
<td>String(255)</td>
</tr>
<tr>
<td>project_id</td>
<td>String(255)</td>
</tr>
</tbody>
</table>

(c) The constraints Table

(d) The instance-metadata Table
3-Tier Application System Configurations

A: Virtual Resources, Attributes and Constraints

- **ATTR\textsuperscript{tier}={tier,versionVM,status}**
- **SCOPE\textsubscript{tier}={presentation,application,database}**
- **SCOPE\textsubscript{versionVM}={psV1,psV2,app1,dbV1,dbV2}**
- **SCOPE\textsubscript{status}={running,stop,suspended}**

- **ATTR\textsuperscript{img}={tier,versionIMG}**
- **SCOPE\textsubscript{tier}={presentation,application,database}**
- **SCOPE\textsubscript{versionIMG}={psI1,psI2,appI1,dbI1}**

- **ATTR\textsuperscript{net}={netType}**
- **SCOPE\textsubscript{netType}={outerNet,psNet,appNet,dbNet}**

- **ATTR\textsuperscript{route}={route}**
- **SCOPE\textsubscript{route}={outerR,psToappR,appTodbR}**

- **ATTR\textsuperscript{io}={ioType,volumeSize}**
- **SCOPE\textsubscript{ioType}={regular,fast,fastest}**
- **SCOPE\textsubscript{volumeSize}={regular,large,huge}**

C: Constraint Specification

Constraints for router-network connection mapping:

- **Const 1:** If route attribute of a router is outerRoute then only network with netType outerNet can connect to it.

  \((\text{route}(\text{router})=\text{outerR}) \rightarrow ((\text{netType}(\text{network})=\text{outerNet}) \vee (\text{netType}(\text{network})=\text{psNet})))\)

Constraints for network-vm connection mapping:

- **Const 2:** Only presentation layer vm can connect to a psNet network. Similar, constraints can be generated for other layer vms and networks.

  \((\text{netType}(\text{network})=\text{psNet}) \rightarrow (\text{tier}(\text{vm})=\text{presentation}))\)

- **Const 3:** A network can be removed from a vm if the vm status is stop.

  \((\text{status}(\text{vm})=\text{stop}))\)

Constraints for image-vm connection mapping:

- **Const 4:** If tier of IMG is presentation and versionImg is psI1, it cannot be used by presentation VM with versionVM psV2.

  \(((\text{tier}(\text{image})=\text{presentation}) \land (\text{versionIMG}(\text{image})=\text{psI1})) \rightarrow (((\text{tier}(\text{vm})=\text{presentation}) \land (\text{versionVM}(\text{vm})\neq \text{psV2})) \vee (\text{tier}(\text{vm})\neq \text{presentation})))\)

Constraints for storage-vm connection mapping:

- **Const 5:** A presentation vm cannot get fastest ioType storages.

  \(((\text{tier}(\text{vm})=\text{presentation}) \rightarrow (\text{ioType}(\text{storage})\neq \text{fastest})))\)
## Hadoop Cluster Configurations

### A: Virtual Resources, Attributes and Constraints

![Diagram showing virtual resources, attributes, and constraints]

**Constraints**

1. **Constraint 1**
   - If `route` attribute of a router is `outerRoute` then only network with `netType` `outerNet` and `clientNet` can connect to it and if `route` attribute is `taskRoute` then it cannot be connected with `nameNet`, `outerNet` and `clientNet`.
   
   \[
   \begin{align*}
   &((route\ (router)=outerRoute) \rightarrow \nonumber \\
   &((netType\ (network)=outerNet) \lor \ (netType\ (network)=clientNet))) \\
   \land \nonumber \\
   &((route\ (router)=taskRoute) \rightarrow \nonumber \\
   &((netType\ (network)\neq nameNet) \land \ (netType\ (network)=outerNet)) \\
   \land &\ (netType\ (network)=clientNet))
   \end{align*}
   \]

2. **Constraint 2**
   - In a `nameNet` network only `nameNode` and `jobTracker` `vm` can be connected.
   
   \[
   \begin{align*}
   &((netType\ (network)=nameNet) \rightarrow \nonumber \\
   &((nodeName\ (vm)=nameNode) \lor \ (nodeName\ (vm)=jobTracker)))
   \end{align*}
   \]

### B: Scopes of the Attributes

- **ATTR**
  - **ATTR\^VM\_hadoop** = `{nodeName}`
  - **SCOPE**
    - **SCOPE\_{nodeName}** = \{clientNode, nameNode, jobTracker, mapTask, reduceTask\}
  - **ATTR\^NET\_hadoop** = `{netType}`
  - **SCOPE**
    - **SCOPE\_{netType}** = \{outerNet, clientNet, nameNet, jobNet, mapNet, reduceNet\}
  - **ATTR\^RT\_hadoop** = `{route}`
  - **SCOPE**
    - **SCOPE\_{route}** = \{outerRoute, nameRoute, jobRoute, taskRoute\}

### C: Constraint Specification

**Constraints for router-network connection mapping:**

- **Constraint 1**
  - If `route` attribute of a router is `outerRoute` then only network with `netType` `outerNet` and `clientNet` can connect to it and if `route` attribute is `taskRoute` then it cannot be connected with `nameNet`, `outerNet` and `clientNet`.

- **Constraint 2**
  - In a `nameNet` network only `nameNode` and `jobTracker` `vm` can be connected.