Policy Canvas
Draw your policies for OpenStack services

Joon-Myung Kang, Mario Ant Sánchez, Sujata Banerjee, Hewlett Packard Labs
{firstname.lastname}@hpe.com

Jeongkeun “JK” Lee
Barefoot Networks
jklee@barefootnetworks.com
Talk Outline

Policy management in OpenStack

Policy Canvas for OpenStack

How to define policies

Policy Canvas Demo
Policy Management in OpenStack
Policy Management in OpenStack

- Cloud operator
- Network Administrator
- Tenant Administrator
- Application Developer
- User
- Application
- Keyston
- Cinder
- Swift
- Murano
- Tacker
- Ironic

Multiple Users

- openstack™

Hewlett Packard Enterprise
Policy Management in OpenStack

Multiple Policies

Multiple Users

User

Performance

QoS

Application

ACI

Cloud

Network Administrator

Tenant Administrator

Load Balancer

DPI

IDS

Application Developer

Keystone

Cinder

Swift

Murano

Tacker

Ironic

Application

Developer

User

ACL

Performance

QoS

Tenant Administrator

Multiple Policies

Multiple Users

User

Performance

QoS

Application

Developer

Tenant Administrator

ACL

QoS

Performance

User

Application

Developer

Security

Policy Management

OpenStack Components

Nova
Neutron
Keystone
Heat
Glance
Swift
Cinder
Ironic
Murano

Cloud Computing

Infrastructure

Virtualization
Network Policy Management in OpenStack

– Security groups and rules
  – Allow/Deny protocols and ports
  – Associate to VMs or ports

– Automation (Heat, Murano, etc.)
  – Pre-define templates and application-specific policies

Low-level commands
Fragmented interfaces
Static configurations
Network Policy Management in OpenStack

- Security groups and rules
  - Allow/Deny protocols and ports
  - Associate to VMs or ports
- Automation (Heat, Murano, etc.)
  - Pre-define templates and application-specific policies
- Group-based Policy (GBP)
  - High-level Intent based management for OpenStack
  - Policies between Endpoint Groups (EPG)

Additional API Support
Multiple-writer problem
Policy Management in OpenStack

- Security groups and rules
  - Allow/Deny protocols and ports
  - Associate to VMs or ports

- Automation (Heat, Murano, etc.)
  - Pre-define templates and application-specific policies

- Group-based Policy (GBP)
  - High-level Intent based management for OpenStack
  - Policies between Endpoint Groups (EPG)

- Congress
  - Global policy framework for OpenStack
  - SQL-like language (Datalog) defining policies across services

Steep learning curve
Multiple-writer problem

Example:
P3: Have an error, if VM that is connected to Internet is not using the security group called "secure_private_VM".
=> Equivalent Congress policy:

\[\text{error\_secure(vm)} ::=\]
\[\text{novaservers(vm, name, host, status, tenant\_id, user\_id, image\_id, flavor\_id),}\]
\[\text{connected\_to\_internet(vm, port),}\]
\[\text{not port\_security\_group(port, 'secure\_private\_VMs')}.\]
Policy Management in OpenStack

Challenges

– Decouple high-level intent (the “what”) from the underlying implementation (the “how”)

– Easy to use, intuitive, high-level interfaces

– Automated deployment mechanisms

– Scalable
Policy Canvas
Draw network policies for OpenStack
Policy Management for OpenStack
Idea from Real World
Policy Canvas
Draw your policies for OpenStack services
Policy Canvas: Graph Abstraction & Composition

Policy sources

Graph Abstraction

Unified, conflict-free policy graph

Deploy

Intuitive to human, readable to system

Proactive policy composition
1) Resolve policy conflicts prior to runtime deployment
2) Simplify policy enforcement & runtime trouble-shooting
PGA: Policy Graph Abstraction & Composition
PGA: Graph Model & Composition

Graph model

Node: End-Point Group (EPG)
Edge: communication intent

Logical labels

Decouple policy from target specifics
Defined from existing data sources

Graph composition

Compute the union of input policies
Resolve conflicts in the intersection
Normalize into disjoint sets

Label relations
Mktg ⊂ Campus
Web ⊂ Cloud

Venn-diagram merging

App admin

P1: Mktg. (Exclusive to Mktg)

Cloud operator

P2: Campus

Node: End-Point Group (EPG)
Edge: communication intent

Decouple policy from target specifics
Defined from existing data sources

Compute the union of input policies
Resolve conflicts in the intersection
Normalize into disjoint sets

App admin

P1: Mktg.

Cloud operator

P2: Campus

https, ssh

LB

https

Web

Mktg & Campus

~ Mktg & Campus

https, ssh

~ Web & Cloud

BC

ARA

Venn-diagram merging
Current solutions: manual composition

**App admin P1:**
Allow Marketing employees **exclusive** https access to Web servers through Load Balancer (LB)

**Cloud operator P2:**
Allow https and ssh traffic from Campus to Cloud through Byte Counter (BC)

**Placements:**
Marketing in Campus
Web in Cloud

---

Group-based policy (GBP)

**Overlap and conflict**

**Clause (prioritized):**
1. Mktg \(\rightarrow\) Web: Web-Access
2. * \(\rightarrow\) Web: Web-Block
3. Campus \(\rightarrow\) Cloud: Cloud-Access

---

**Composition Algorithm**

**Subject1: Web-Access**
tcp, dstport=https : Permit, LB-BC chain

**Subject2: Web-Block**
* : Deny

**Subject3: Cloud-Access**
tcp, dstport=https|ssh : Permit, BC chain
PGA continued

- Formal graph model, data structure, algorithm
- Label relations in Label Trees

(a) Enterprise IT admin
- Engg
- Mktg

(b) Application admin
- Empl
- HTTP
- LB
- Web
- SQL
- DB

(c) SDN app: Net Protector
- Normal
- DNS
- DPI
- DNS

(d) Cloud operator
- Campus
- *FW
- BC
- Cloud

- Scalable: 13 mins to compose 20K ACL + service policies
- Intuitive reasoning & troubleshooting by graph walk

Label Namespace

<table>
<thead>
<tr>
<th>Tenant</th>
<th>Label Trees</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empl</td>
<td>App</td>
<td>Location</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Status</td>
</tr>
<tr>
<td>Engg</td>
<td>Mktg</td>
<td>Net protector</td>
</tr>
<tr>
<td>Cmp-A</td>
<td>Cmp-B</td>
<td>Normal</td>
</tr>
<tr>
<td>Normal</td>
<td>Qn</td>
<td></td>
</tr>
</tbody>
</table>

Label Mappings
- Engg: Campus-A
- Mktg: Campus-B
- App : Cloud
- Empl: Net protector
- Engg&Cmp-A &Qn
- Mktg&Cmp-B &Qn

Hewlett Packard
Enterprise
Intent APIs to Express Hidden Intents

**Whitelisting**

WL = \{HTTPS\}

\[
\begin{aligned}
&\text{src} \xrightarrow{\text{WL}} \text{dst} \\
\{^*\} - \text{WL} : \text{deny or don't care?}
\end{aligned}
\]

Default edge: MUST

No edge: Conditional

**Blacklisting**

BL = \{SSH, ICMP\}

\[
\begin{aligned}
&\text{src} \xrightarrow{\text{BL}} \text{dst} \\
\{^*\} - \text{BL} : \text{allow or don't care?}
\end{aligned}
\]

Default edge: Block

No edge: CAN communicate

**Intent ACL edges**

Reduce %conflict: 50% → 12.5%

Enable systematic composition
PGA Adoption and Publicity

- **Adoption**
  - ODL Network Intent Composition (NIC) project adopted graph compiler and intent ACL
    [https://wiki.opendaylight.org/view/Network_Intent_Composition:Graph](https://wiki.opendaylight.org/view/Network_Intent_Composition:Graph)

- **Paper, Demo**

- **Talks**
  - OpenStack Summit Vancouver #vBrownBag session, [https://www.youtube.com/watch?v=jDCxUQ5D4Y4](https://www.youtube.com/watch?v=jDCxUQ5D4Y4)
  - OpenDaylight Summit 2015, Santa Clara, CA, [https://www.youtube.com/watch?v=EHMSuWz3LHw](https://www.youtube.com/watch?v=EHMSuWz3LHw)
Policy Canvas System
Policy Canvas
System overview

Policy Canvas (Horizon module)

Python-pgaclient
(python bindings to PGA API)

PGA service

Label drivers
Compilation drivers
Enforcement drivers

PGA extension
- Input policy graph
- Composed policy graph
- Deployed policy graph
- Label tree
- Function box

REST APIs

Python-pgaclient
(python bindings to PGA API)

Policy Canvas
(Horizon module)

- Input policy graph
- Composed policy graph
- Deployed policy graph
- Label tree
- Function box

PGA service

Label drivers
Compilation drivers
Enforcement drivers

**Hewlett Packard Enterprise**
Policy Canvas
Policy graph creation/composition

User/App\textsubscript{1} \quad User/App\textsubscript{2} \quad \ldots \quad User/App\textsubscript{n}

Draw policy graphs

Policy Canvas (Horizon module)

Label tree(s) \quad Input policy graphs \quad Composed policy graph

PGA service

Data sources (nova, keystone, \ldots)

Label drivers

Compilation drivers

PGA graph composer

OpenDaylight NIC graph compiler

Hewlett Packard Enterprise
Policy Canvas
Policy graph deployment

Admin

Deploy

Policy Canvas (Horizon module)

Composed policy graph

PGA service

Nodes and edges

classifier + action

enforcement drivers

Congress driver

Neutron driver

Endpoint monitoring rules

security groups, SFC, LB/FW ...

action

classifier

Congress

Data sources
(nova, keystone, ...)

Neutron

clients

HTTP

LB

web

Hewlett Packard
Enterprise

24
Policy Canvas
Automatic policy mapping

Admin

Policy Canvas (Horizon module)

PGA service

enforcement drivers

Congress driver  Neutron driver

notify membership change  Associate security group

Congress

Data sources (nova, keystone, ...)

Neutron

clients

HTTP

LB

web

VM

Hewlett Packard Enterprise
Policy Canvas
1. Demo Scenario
2. Demo
Demo Scenario

Availability Zone 1 (AZ1)

- VM
  - cnc32
  - cnc27

Availability Zone 2 (AZ2)

- VM
  - cnc3
  - cnc4

SSH
PING
HTTP

Global Policy

Host Manager
Zone Manager
Policy Canvas Demo
Future Directions
Extending Policy Canvas

- Importing already-expressed policies directly from neutron or other services for composition and analysis
- Modeling and composing newly-added neutron features QoS
- Port-level connectivity
Policy Enforcement and Verification

Policy is eventually enforced at various network dataplanes

Need abstraction to capture capabilities and limitations
  Can this network implement my policy?

Need programmable dataplane to implement complex policies
  Tailor the network, rather than retrofit your policy into it

Need complete network visibility to verify policy
  Does my policy get correctly enforced?

Industry-wide open-source efforts
  OpenFlow Data Plane Abstraction (OF-DPA)
  OpenCompute Switch Abstraction Interface (SAI)
  P4: Programming Protocol-independent Packet Processor
P4: Programming Protocol-independent Packet Processor

High-level language to define and modify dataplane
Protocol/target independent
Open source, Apache 2.0 CLA
Adopted by OpenSwitch, OpenCompute SAI

Benefits in policy enforcement
  Pick the best protocol for your policy, remove the others
Custom protocol carrying SFC metadata
Embed service functions (e.g., L4 LB) in the switch
Policy verification by In-Network Telemetry

```
#define MAX 100

/* Generator for action vector */

action rmac_hit() {
  modify_field(l3_metadata.rmac_hit, TRUE);
}

action rmac_miss() {
  modify_field(l3_metadata.rmac_hit, FALSE);
}

table rmac {
  reads {
    l3_metadata.rmac_group : exact;
    l2_metadata.1kp_mac_da : exact;
  }
  actions {
    rmac_hit;
    rmac_miss;
  }
  size : ROUTER_MAC_TABLE_SIZE;
}
```
Policy down to Dataplane

Composed Policy

Program

Compile

Auto-Generated runtime API

Target Binary

Neutron

Network controller

SBI Agent

Driver

Programmable Data Plane (hw, sw)

Northbound API

Southbound API (SBI)
Conclusions

– Policy Canvas
  – Simple and intuitive abstraction (Graph model)
  – Portable policies, decoupled from infrastructure specifics
  – Automatic and proactive composition of independently specified policies

– Status
  – Horizon GUI
  – PGA service and API
  – Neutron/Congress drivers
  – Graph compiler in OpenDaylight NIC

– Collaboration with OpenStack community
  – More use cases and feedback
  – Code contribution to OpenStack
Thank you