Auditing as a Service Part 2

CS528 - Cloud Computing

Mentors: Sahil Tikale - Naved Ansari

Ali Raza - Rushi Patel - Chenxi Li - Kevin Liang

Background

- **Bare Metal Clouds**
  - Bare metal machines instead of virtual machines
  - Current bare metal clouds also do provisioning
  - Each tenants may have different requirements
  - How do we ensure different tenants can share the cloud?
  - Tenant does not have the flexibility

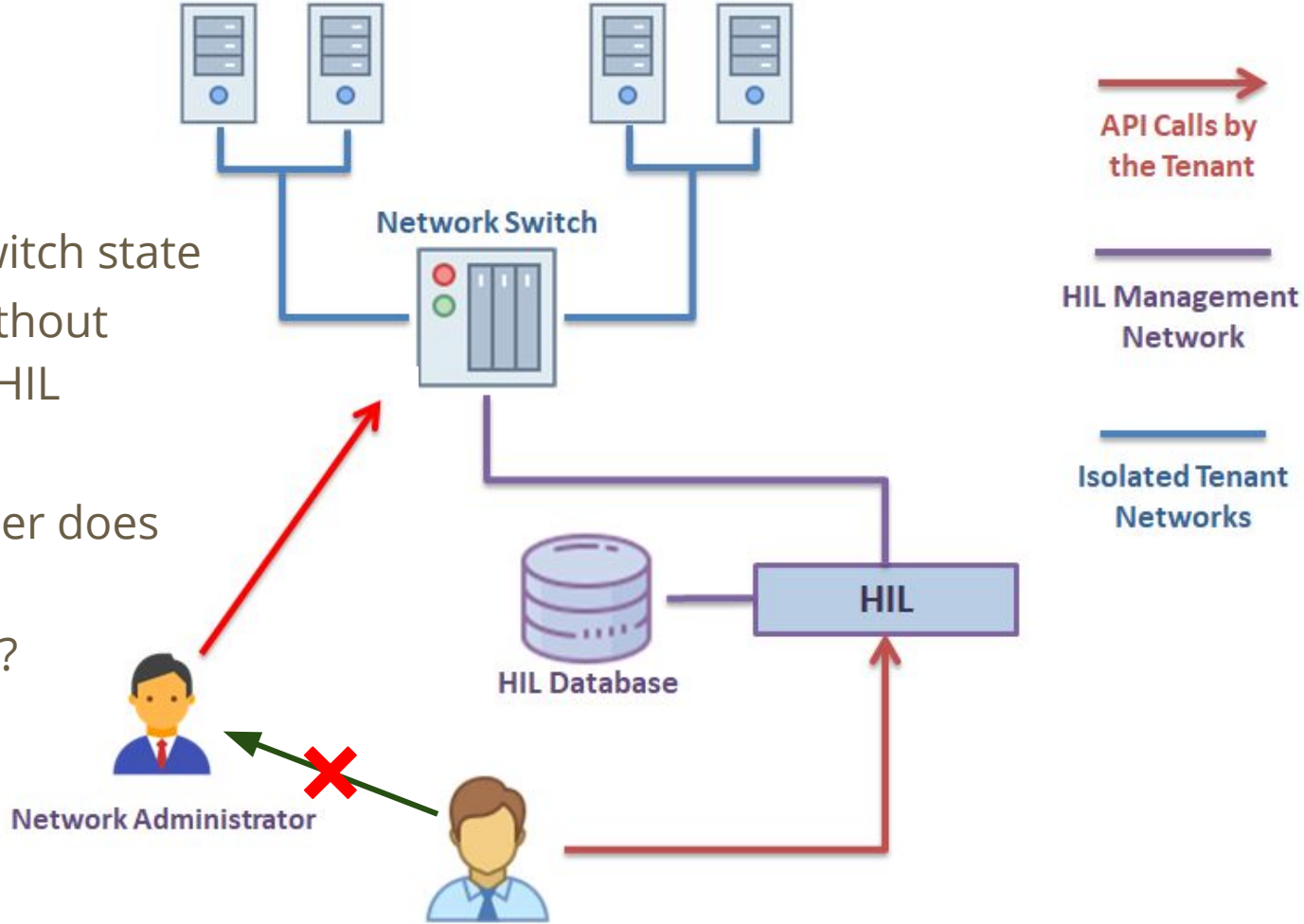
Background

- **Hardware Isolation Layer (HIL)***
  - Provides isolated network of nodes
  - Exokernel for the cloud
  - Provides as little abstractions as possible
  - Allows tenants to choose provisioning system and other services
  - Move resources between multiple clusters
  - Allows security sensitive applications to use public clouds

* Hennessey, Jason, et al. "HIL: designing an exokernel for the data center." *Proceedings of the Seventh ACM Symposium on Cloud Computing*. ACM, 2016.

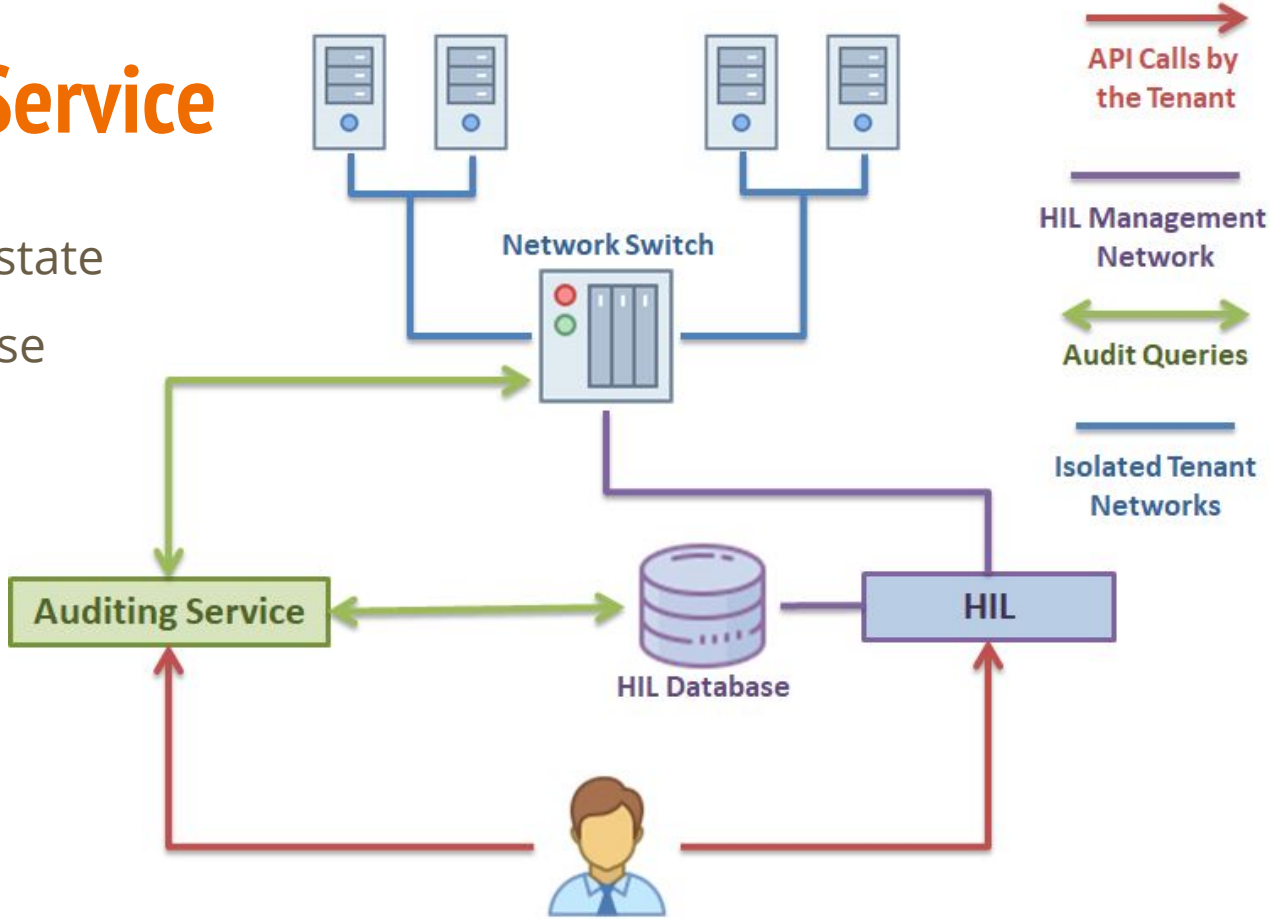
Motivation

- What if the switch state is changed without updating the HIL database?
- What if the user does not trust the administrator?



Auditing as a Service

- Query the switch state
- Query HIL database
- Check and report discrepancies



Updated Epics

- **Epic 1**
  - Minimum Viable Product
  - Current state of HIL and network switches - Simple Yes or No at the start
- **Epic 2**
  - History and database of audit system
  - Automated service to maintain status of system

Votes	Name	Project	Sprint	Assigned	Status	Progress
▲ 0	#41 Audit system returns yes or no (Current State) EPIC				New	
▲ 0	#50 Audit system history and automated EPIC				New	

View options

Related user stories

#4 Setup HIL simulation environment	Done	Not assigned
#8 Design REST API interface - Allow for Communication with CURL and CLI	Done	Chenxi Li
#10 Create CLI application to interact with audit system	In progress	Kevin Liang
#33 Create Back-end Audit System - Simple function to network switch - Part 1	Done	Not assigned
#57 Create Back-end Audit System - Simple function to network switch - Part 2	New	Not assigned
#62 Connect CLI with REST API	New	Kevin Liang
#65 Connect REST API with backend Audit code	New	Chenxi Li
#69 Test system with simulation environment	New	Not assigned

Second Sprint - MVP

- Setup HIL simulation environment (All) - Week 1
- Create Backend audit system
  - Simple calls to HIL database (Rushi)
  - Abstract class for switches (Ali)
- Create CLI application to interact with Audit system (Kevin)
- Design REST API (Mark)

HIL Simulation - CentOS

- Create CentOS VM and installed the HIL
- HIL Register Node
  - hil node_register node1 ipmi host1 user1 pass1234;
- HIL Register NICs
  - hil node_register_nic node1 eth0 aa:bb:cc:dd:ee:01;
- HIL Create Projects
  - hil project_create proj1
- Add nodes to projects
  - hil project_connect_node proj1 node1
- List Nodes on project
  - hil list_project_nodes proj1

REpresentational State Transfer(REST) API for Auditing Service

Simple REST API to query switch

1. bash

python

```
(haas_auditing) dhcp-wifi-8021x-155-41-
51-99:haas_auditing markli$ python mana
ge.py runserver
Performing system checks...

System check identified no issues (0 si
lenced).
February 22, 2018 - 17:13:04
Django version 1.11.10, using settings
'haas_auditing.settings'
Starting development server at http://1
27.0.0.1:8000/
Quit the server with CONTROL-C.
[22/Feb/2018 17:13:08] "GET /switch HTT
P/1.1" 200 57
```

bash

```
dhcp-wifi-8021x-155-41-51-99:Desktop ma
rkli$ curl http://127.0.0.1:8000/switch
[{"id":1,"project":"moc","node":"node01
","eth0":"eth01"}]dhcp-wifi-8021x-155-4
rkli$ 9:Desktop mar
```

Calls REST API to query table for switches

Server sends back Javascript Object Notation(JSON) response

REpresentational State Transfer(REST) API for Auditing Service

Used Django framework to create dummy REST API to return “switch” object

Steps:

1. Set up database table “switch”

2. Set up url pattern so using “switch” as suffix for url will call REST API to send back Json ---- url(r'^switch', views.switchList.as_view())

3. Write serializer class to turn database object into JSON object
---- class switchSerializer(serializers.ModelSerializer)

Base CLI to Connect to REST API

```
└[>>> ./switch_auditor_cli.py --help
CLI for auditing system to verify the integrity of the HIL database

Usage:
  switch_auditor_cli.py -h | --help | --version
  switch_auditor_cli.py [check-state | check-diff | show-diff] [options]

Examples:
  switch_auditor_cli.py check-diff -v myproject -o outfile
  switch_auditor_cli.py show-diff --quiet myproject
  switch_auditor_cli.py check-state myproject

Commands:
  check-state PROJECT [PORT | VLAN]  query auditing service to get switch state
  check-diff PROJECT [PORT | VLAN]   check if there's a difference between the switch state versus HIL database
  show-diff PROJECT [PORT | VLAN]    show differences between the switches and auditing system
  check-vlan PROJECT PORT            check for connected VLANs for a given port
  check-port PROJECT VLAN            check for connected ports for a given VLAN

Options:
  -h, --help                         show this
  --quiet                            print less text
  -v, --verbose                      print more text
  --version                          show version
  -o FILE                            print the output to a file

Arguments:
  PROJECT                            currently selected project
  FILE                               output file
  PORT                               currently selected port
  VLAN                               currently selected VLAN
```

Backend Audit - Switch Calls

Two different scenarios

- There are hidden nodes/ports on my network
- My node/port is connected to hidden networks

So the switch drivers need to implement two functions

- get_port_networks
  - What ports are on my network?
- get_network_ports
  - What networks are my ports connected to?
- Input will be a list of ports or networks
- Output will be a dictionary of key value pairs
  - E.g., for ports, key will be ports and value will be a list of networks the port is connected to

Backend Audit - HIL database access

- Currently the access to the HIL database mimics the method that HIL already uses.
- Read-Only calls are used since the audit system will not produce any additional changes to the HIL system.
  - List networks
  - List projects
  - List nodes - all or free
  - List node info
- This requires HIL to be installed on the current system
  - This is ok for the current requirements of the system.

Backend Audit - HIL database access

hil_audit list_nodes
[u'node1', u'node2', u'node3', u'node4', u'node5', u'node6', u'node7', u'node8', u'node9']
hil_audit list_projects
[u'proj1', u'proj2', u'proj3']
hil_audit show_node node1
{u'project': u'proj1', u'nics': [{u'port': None, u'switch': None, u'macaddr': u'aa:bb:cc:dd:ee:01', u'networks': {}, u'label': u'eth0'}], u'name': u'node1', u'metadata': {}}

HIL REST API SERVER OUTPUT
INFO:hil.rest:In request context 1f7e25c0-1551-46e4-ac77-d86bcf934d1b: API call: **list_nodes**(is_free=u'all')
INFO:werkzeug:127.0.0.1 - - [21/Feb/2018 22:03:23] "**GET /nodes/all HTTP/1.1**" 200 -
INFO:hil.rest:In request context a95ca00b-2896-4e0f-b8ab-b354114d78ab: API call: **list_projects**()
INFO:werkzeug:127.0.0.1 - - [21/Feb/2018 22:06:14] "**GET /projects HTTP/1.1**" 200 -
INFO:hil.rest:In request context 5fd75f7e-f83e-4574-b3bd-929268862bf2: API call: **show_node(nodename=u'node1')**
INFO:werkzeug:127.0.0.1 - - [21/Feb/2018 22:07:01] "**GET /node/node1 HTTP/1.1**" 200 -

Next Sprint - MVP cont.

User Stories:

1. Complete backend audit system
    a. Simple functions to perform differences between HIL db and network switches
2. Connect CLI to REST API
3. Connect REST API to audit system backend
4. Test system with simulation environment

Thank you!