ECA YANG DATA Model

IETF 106

Nov. 16-17, 2019

Singapore

Hackathon Plan

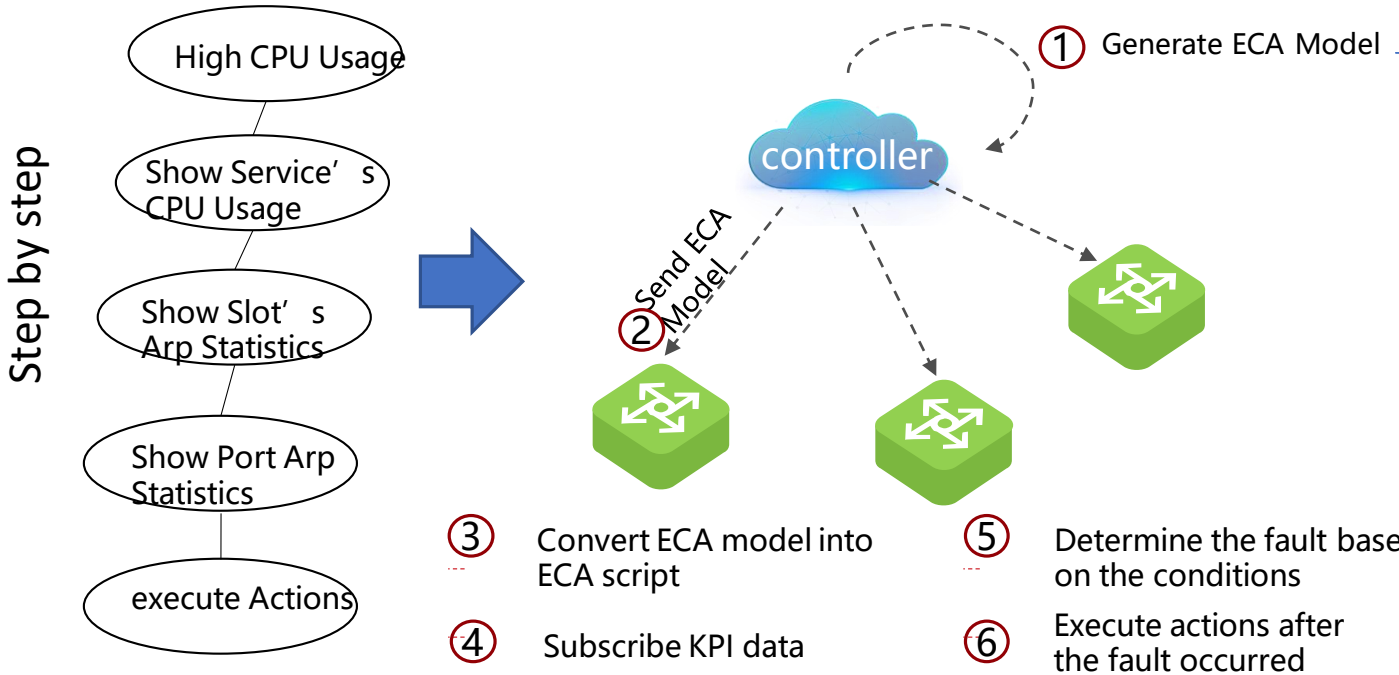
- Verify the validity and feasibility of YANG Data model for Policy based Event Management:
  - Implementation and testing of an ECA fault localization and self-healing application in the network device,
    - i.e. Automatic detect ARP Attack.
  - Implementation and testing of chain reaction of coordinated events
    - One event can trigger another event, i.e., the action output in the first event can be input to target in the second event and a set of events can be grouped together and executed in a coordinated manner.
- Specifications:
  - https://tools.ietf.org/html/draft-wwx-netmod-event-yang-04
  - https://tools.ietf.org/html/draft-bwd-netmod-eca-framework-00

Usage Example

Scenarios: Automatic detect ARP Attack

Before:
Relying on expert experience, manual troubleshooting, time-consuming

After:
Based on ECA YANG Model, realize fault self-healing



Event 1:
Condition:
The average CPU utilization of the slot is greater than threshold.
Action:
- Output the specific slot id which system's cpu usage is high
- Take a snapshot of current system's cpu usage for a specific slot

Event 2:
Condition:
The average CPU utilization of the slot is greater than threshold.
Action:
- Output the specific slot id which system's cpu usage is high
- Limit the cpu committed access rate by sending netconf command

Event 3:
Condition:
The average CPU utilization of the ARP is greater than threshold.
Action:
- Output the specific slot id which system's cpu usage is high
- Trigger another event with the specific slot id

Event 4:
Condition:
1- Event 3 triggered;
2- Monitoring the number of ARP packets received for the slot that be report by event 3, if it greater than threshold
Action:
- Output the slot id which the number of ARP pkts received is high
- Logging the ARP attack

What Got Done

- Achievements
  - Develop an ECA fault localization and self-healing application in the network device
    - To provide automatic fault localization and self-healing, i.e. ARP attack, based on ECA YANG Model
  - Develop an network visibility application to display the event changing over time.

Instances: ECA Scripts generated by ECA model

```
def eca_condition():
    fmc.Event("system_cpu") \
        .set_kpi(fmc.kpi.system_cpu_usage(), "alias_a", fmc.range.all_slots()) \
        .set_compute_function(fmc.compute.avg_lastN("alias_a", 2), "reta") \
        .set_relation_function("reta", fmc.relation.greater(), 40, fmc.period.frequency(2, 1)) \
        .add_output("slot", "reta", fmc.output.slot()) \
        .register()
    # arp_check.py  huawei-devm:devm/serviceCpuInfos/serviceCpuInfo key position 3 serviceName:ARP  kpi:serviceCpuUsage

    fmc.Strategy("cpu_snapshot") \
        .set_strategy_formula("system_cpu") \
        .set_valid_time(30) \
        .add_output("slot", "system_cpu.slot") \
        .set_action(fmc.action.system_log(), "Cpu usage of slot %s is high", "system_cpu.slot") \
        .set_action(fmc.action.kpi_snapshot(), fmc.kpi.system_cpu_usage(), "system_cpu.slot") \
```

```
def eca_condition():
    fmc.Event("system_cpu") \
        .set_kpi(fmc.kpi.system_cpu_usage(), "alias_a", fmc.range.all_slots()) \
        .set_compute_function(fmc.compute.avg_lastN("alias_a", 3), "reta") \
        .set_relation_function("reta", fmc.relation.greater(), 40, fmc.period.frequency(1, 1)) \
        .add_output("slot", "reta", fmc.output.slot()) \
        .register()

    # arp_check.py  huawei-devm:devm/serviceCpuInfos/serviceCpuInfo key position 3 serviceName:ARP
    fmc.Strategy("cpu_recovery") \
        .set_strategy_formula("system_cpu") \
        .set_valid_time(40) \
        .add_output("slot", "system_cpu.slot") \
        .set_action(fmc.action.system_log(), "Cpu usage of slot %s is high", "system_cpu.slot") \
        .set_action(fmc.action.lower_cpucar(), "system_cpu.slot") \
        .register()
```

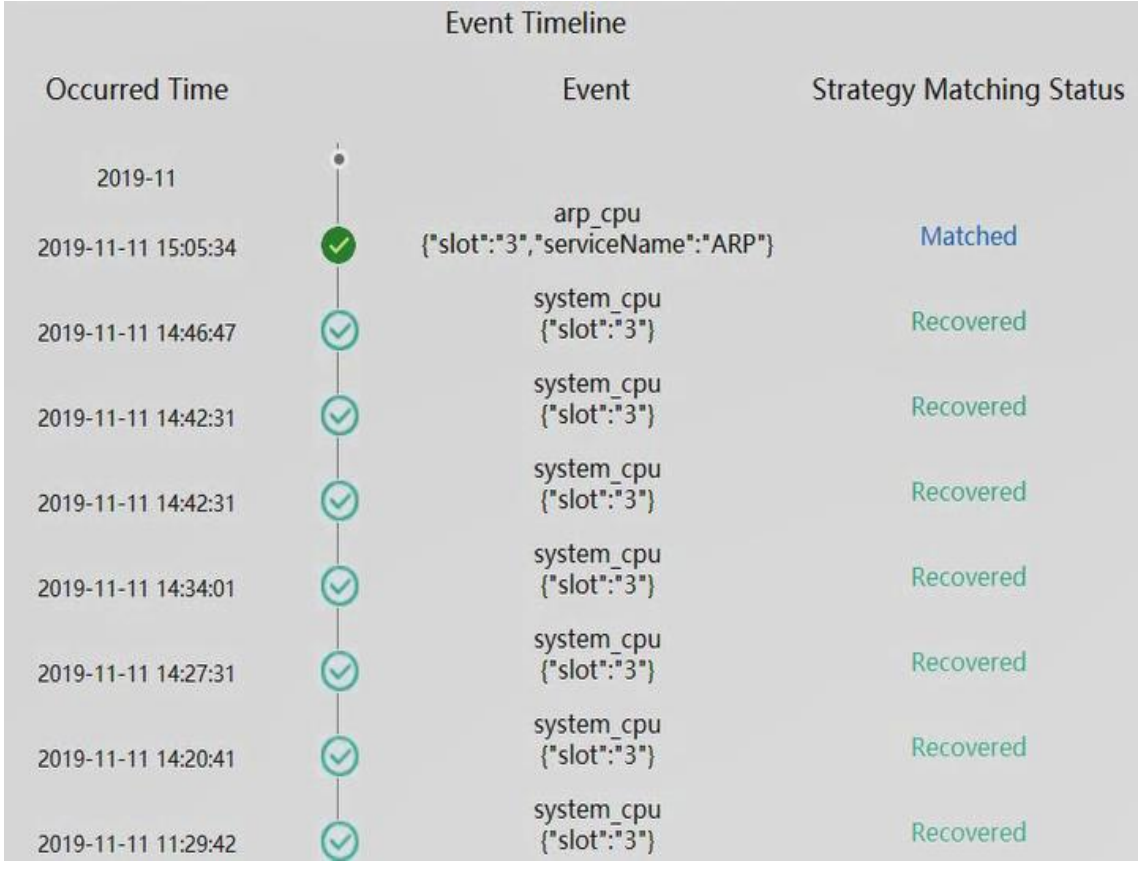
```
def eca_condition():
    fmc.Event("arp_cpu") \
        .set_kpi(fmc.kpi.service_cpu_usage(), "alias_a", fmc.range.all_slots(), fmc.range.specified_service("ARP")) \
        .set_compute_function(fmc.compute.avg_lastN("alias_a", 3), "reta") \
        .set_relation_function("reta", fmc.relation.greater(), 5, fmc.period.frequency(1, 1)) \
        .add_output("slot", "reta", fmc.output.slot()) \
        .register()

    fmc.Strategy("arp_cpu_handle") \
        .set_strategy_formula("arp_cpu") \
        .set_valid_time(60) \
        .add_output("slot", "arp_cpu.slot") \
        .set_action(fmc.action.system_log(), "Cpu usage of ARP service on slot %s is high.", "arp_cpu.slot") \
        .set_action(fmc.action.nested_script(), "arp_attack.py", 120) \
        .register()
```

```
def eca_condition():
    fmc.Event("arp_pkt_received") \
        .set_kpi(fmc.kpi.arp_pkt_received(), "alias_c", fmc.range.get_input("slot")) \
        .set_compute_function(fmc.compute.minus_last("alias_c"), "retc") \
        .set_relation_function("retc", fmc.relation.greater(), 900, fmc.period.frequency(1, 1)) \
        .add_output("slot", "retc", fmc.output.slot()) \
        .register()

    fmc.Strategy("arp_attack_handle") \
        .set_strategy_formula("arp_pkt_received") \
        .set_valid_time(20) \
        .add_output("slot", "arp_pkt_received.slot") \
        .set_action(fmc.action.system_log(), "Slot %s has received a large amount of ARP packets. The device is suffering from ARP
attacks.", "arp_pkt_received.slot") \
        .register()
```

System view



What We Learned

- IETF ECA YANG can be deployed to not limited to support automatic fault locating and self-healing.
- There are many other valuable use cases such as:
  - Smart Filter;
  - TE Path Computing.
- ECA YANG is the key to network management automation and more input and suggestions are welcome.

The Crews

- Champion(s)
  - Zitao Wang <wangzitao@huawei.com>
  - Daniel King <d.king@lancaster.ac.uk>
  - Chongfeng Xie <xiechf.bri@chinatelecom.cn>
  - Xiaopeng Qin <qinxiaopeng@huawei.com>
- Members:
  - Yu Yang;
  - Fengwei Qin;
  - Lan Xu;
  - Jingjing Yu;

Thanks!