Verify Network Topology using packet injection

Mike Korshunov, TME, Web Solutions, Cisco.

October, 2016

Improperly connected wires?

- Dangled cables caused problems that we spend hours debugging at the Network layer. Let’s solve this problem with software approach.

Topologies

We have topology in one of the popular format, such as JSON/YAML. Orchestrator spins up the topology. Ansible hosts are automatically filled out from the topo file.

YAML

```
name: 10 nodes topo

orchestration: vagrant

nodes:

  - name: server_1
    type: tgen
    os: linux_ubuntu
    box: ubuntu/trusty64
    mgmt_ip: localhost
    ports:
      - type: ssh
        value: 2521
    interfaces:
      - interface: eth1
        link-name: link1
```

JSON


```
{
  "nodes":[
    {
      "box":"ubuntu/trusty64",
      "name":"server_1",
      "mgmt_ip":"localhost",
      "os":"linux_ubuntu",
      "interfaces":[
        {
          "interface":"eth1",
          "link-name":"link1"
        }
      ],
      "type":"tgen",
      "ports":[
        {
          "type":"ssh",
          "value":2521
        }
      ]
    },
```

Solution: Packet Injection

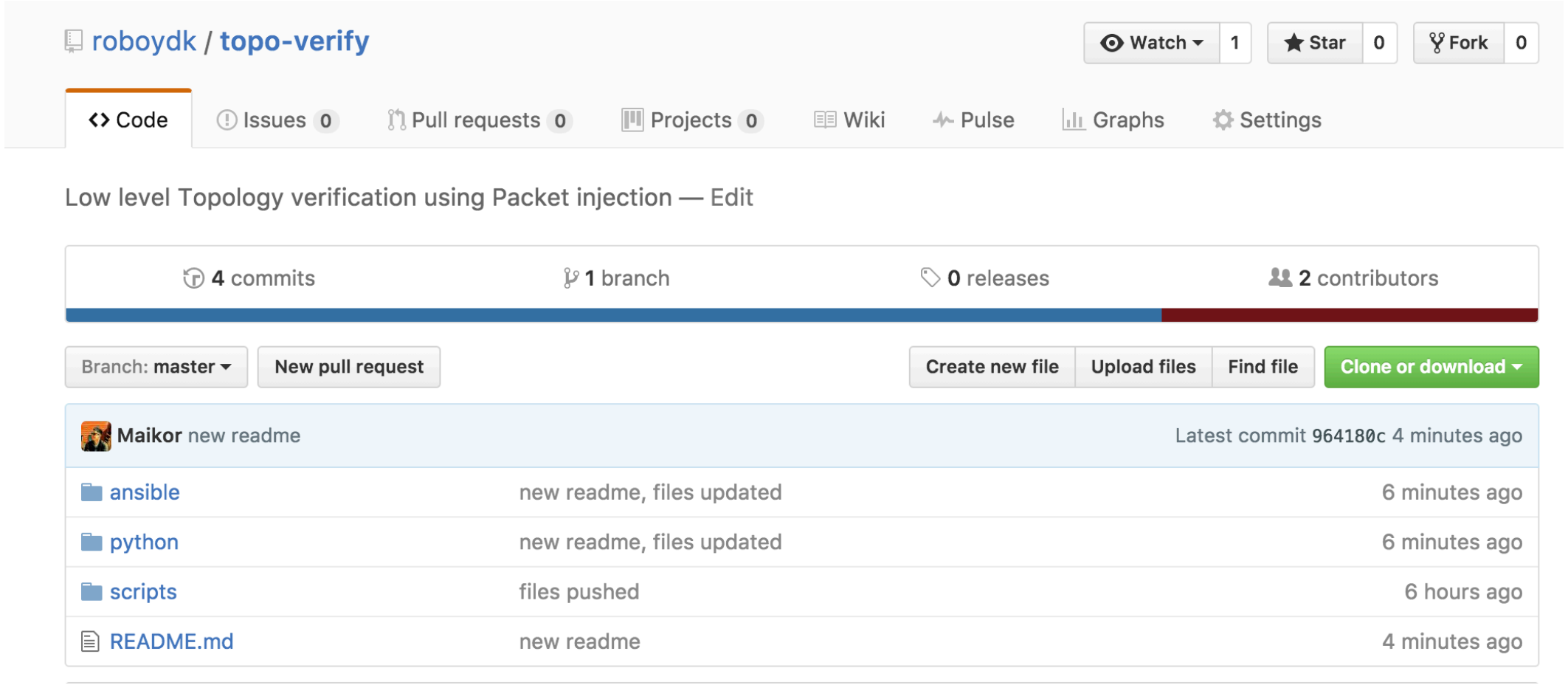
- Packet Injection doesn't care about routing state or any discovery protocols. Works without IP address assigned on port, uses raw sockets to inject packets.
- Required data: **port** to which send packet, destination **mac** address
- Python script will connect to every available device in topology and will retrieve destination macs.
- As second step: script will send packets from every node to it's neighbors and will do vice-versa. So each connection on node will be verified as packet sender and receiver.

Solution: Packet Injection

- Packet Injection doesn't care about routing state or any discovery protocols. Works without IP address assigned on port, uses raw sockets to inject packets.
- Required data: **port** to which send packet, destination **mac** address
- Python script will connect to every available device in topology and will retrieve destination macs.
- As second step: script will send packets from every node to it's neighbors and will do vice-versa. So each connection on node will be verified as packet sender and receiver.

Workflow

- Github repo: https://github.com/roboydk/topo-verify/



Workflow #2

- 2 Pre step, spin up vagrant configuration and play ansible playbooks.

$ **vagrant status**
Current machine states:

server_1	running (virtualbox)
server_2	running (virtualbox)
server_3	running (virtualbox)
tor_1	running (virtualbox)
tor_2	running (virtualbox)
tor_3	running (virtualbox)
spine_1	running (virtualbox)
spine_2	running (virtualbox)
edge	running (virtualbox)

$ **ansible-playbook playbooks/eline.yml -i ansible_hosts**
PLAY [network-nodes]
****

TASK [copy public part of key]
****

changed: [server_3]
changed: [tor_2]
changed: [tor_1]
changed: [server_2]
changed: [server_1]
changed: [spine_2]
changed: [tor_3]
changed: [edge]
changed: [spine_1]

Workflow #3

- Manual example:

Used port to sent packet

vagrant@server-1:~$ **sudo ./send-raw -i eth1 -s 255.255.255.254 -d 255.255.255.255 -m 08:00:27:67:5b:04**
Tx interface: eth1
Source IP: 255.255.255.254
Dest IP: 255.255.255.255pkt
len = 42 bytes
Got ifindex 3
Src mac: 08:00:27:23:6b:67
Dest mac: 8:0:27:67:5b:4

Destination Mac

tx packet:08 00 27 67 5b 04 08 00 27 23 6b 67 08 00 45 00 00 1c 12 34 00 00 40 01 68 af ff ff ff fe ff ff ff ff 08 00 5e 66 99 99 00 00
total bytes = 42

vagrant@server-1:~$

vagrant@tor-1:~$ sudo tcpdump -i eth1
tcpdump: WARNING: eth1: no IPv4 address assigned
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 65535 bytes

21:08:56.267735 IP 255.255.255.254 > 255.255.255.255: ICMP echo request, id 39321, seq 0, length 8

Workflow #4

- Automated example. Shut port on device tor_2 and check output of script:

```
$ python topo_verifier.py
Checking link server_1 ---> tor_1
  Link  server_1 --->  tor_1 ✓
Checking link server_2 ---> tor_2
  Sorry, there is some problem
Checking link server_3 ---> tor_3
      ... Omitted output...
Checking link edge ---> spine_2
  Link  edge --->  spine_2 ✓
```

Online and reachable devices ['server_1', 'server_2', 'server_3', 'tor_1', 'tor_2', 'tor_3', 'spine_1', 'spine_2', 'edge']

Device with connection problems in between: [['server_2', 'tor_2']]

Links:

- Source code for talk: https://github.com/roboydk/topo-verify
- Site with tutorials/docs: https://xrdocs.github.io/
- Follow us on twitter: https://twitter.com/xrdocs
- Catch us tomorrow as we present open source test framework that uses this tool on Wednesday 5:15pm

Thanks!