Securing SDN-based monitoring
Obstacles and learned insights

Ricardo Fonseca
Introduction
An SDN network

- Network Application(s)
  - Open northbound API
  - Open southbound API
- Controller Platform
- Network Infrastructure
  - Data forwarding elements (e.g., OpenFlow switches)
Why network monitoring?

Collects and provides data about the current state of the network

- **Traffic engineering** (QoS-based routing)
- **Failure detection** (e.g., high loss rate)
- **Traffic shaping** (e.g., flow throttling)
- **Load balancing** (e.g., delay-based)
Traditional monitoring technique

Active flow discovery
Traditional monitoring technique

Active flow discovery
Traditional monitoring technique

Active flow discovery

Controller

Switch 1

Switch 2

PACKET_IN
Traditional monitoring technique

Active flow discovery

Controller

Switch 1

Switch 2

PACKET_IN

PACKET_OUT
Traditional monitoring technique

Active flow discovery

Controller

Switch 1

Switch 2
Traditional monitoring technique

**Active probing**

![Diagram of network components]

- Controller
- Switch 1
- Switch 2
Traditional monitoring technique

**Active probing**

Controller

---

Switch 1

---

Switch 2
Traditional monitoring technique

**Active probing**

Switch 1  

Controller  

Switch 2  

PACKET.OUT probe  

forward probe
Traditional monitoring technique

Active probing

Switch 1

Switch 2

Controller

PACKET_OUT probe

forward probe

PACKET_IN probe
Traditional monitoring technique

**Statistics request**

Switch 1 -- regular traffic -- Switch 2

Controller
Traditional monitoring technique

Statistics request

Controller

FLOW_STATS request

Switch 1

regular traffic

Switch 2
Traditional monitoring technique

**Statistics request**

- Flow STATS request
- Flow STATS reply

Switch 1 to Controller:

Switch 2 to Controller:

Controller to Switch 1:

Controller to Switch 2:

Regular traffic:

Switch 1 to Switch 2:

Switch 2 to Switch 1:
Traditional monitoring technique

Statistics request

Controller

FLOW_STATS request

FLOW_STATS reply

Switch 1

regular traffic

Switch 2
Traditional monitoring technique

Packet sampling

Switch 1  regular traffic  Switch 2

Controller
Traditional monitoring technique

Packet sampling

Switch 1

regular traffic

Switch 2

Controller

sampled packets
Traditional monitoring technique

Packet sampling

Switch 1

Controller

Switch 2

regular traffic
Common security pitfalls

• PACKET_IN-based flow discovery is vulnerable to DoS attacks due to slow control path

• Unprotected and detectable probes can be forged, replayed, delayed, etc.

• Switch counters are insufficient for asserting packet integrity
Iterative solution design
Iterative solution design

• Avoid the use of active probes

• Then how do we measure link delay?
  – Sample user packets during unpredictable times by sending them from source and target switches to the controller

• But the same packet must be sampled in both switches, how do we do this reliably?
Iterative solution design

• How do we assert packet integrity in a link?
  – Sample user traffic from source and target switches and compare the two samplings

• Again, the same portion of traffic must be sampled in both switches, how do we do this reliably?
Solving the sampling problem in a link

• Most switches only support random sampling, which do not allow us to sample the same traffic at source and target switches

• We can add a tag to packets at the source switch, periodically modify its value, and sample a slice of traffic during the lifetime of a tag value
Solving the sampling problem in a link
Solving the sampling problem in a link

set tag value to 1

regular traffic (tagged with value 1)
Solving the sampling problem in a link

sampled packets (tagged with value 1)

regular traffic (tagged with value 1)
Solving the sampling problem in a link

Controller

Switch 1

regular traffic (tagged with value 1)

Switch 2
Solving the sampling problem in a link

set tag value to 2

regular traffic (tagged with value 2)
Solving the sampling problem in a link

• Experiments with physical switches show that getting a slice of traffic via the control channel overloads the switches !!!

• The control channel is very slow, so we need an alternative channel to sample packets from switches to the controller
Current solution

• Send packets from the switches to local collector machines via traffic mirroring, which works fine at line rate

• The collectors send time-stamped packet hashes to the controller
  – We still need to resolve the clock synchronization problem
Current solution
Current solution

• Packet delay is measured by getting the difference between the timestamps of the same hash

• Enhanced loss rate is measured by finding the proportion of source traffic that is missing from the target traffic
  – This grants us the assertion of packet integrity
Future optimizations

• Use trajectory sampling at the collectors to reduce the amount of traffic sent to the controller

• Complement sampled probing with active probes, protected with keyed cryptographic hashes
Thank you!