Detecting Suspicious Behavior of SDN Switches by Statistics Gathering with Time

Takahiro Shimizu†, Naoya Kitagawa†, Kohta Ohshima‡, Nariyoshi Yamai†
† Tokyo University of Agriculture and Technology
‡ Tokyo University of Marine Science and Technology

APAN Research Workshop 2018
August 6th, 2018
Outline

• Background
• Related Work
• Proposed Method:
  Forwarding state verification with scheduled statistics gathering
• Experiment
• Discussion
• Conclusion
What is Software Defined Network (SDN)?

- Decouples network control and packet forwarding function
  - Enables flexible network control by centralized control plane
  - Controller is programmable
    - Maintains a global view of the entire network, thus enable efficient routing
  - OpenFlow: A protocol to realize SDN

![Diagram of Traditional Network vs SDN Switch](image-url)
Background

• Software Defined Network: often used **programmable software switches**
  • Increase probability of compromise than traditional hardware switches
  • CVE-2016-2074: buffer overflow vulnerability caused by compromised MPLS packet in Open vSwitch
Background

- Attacks possible from **compromised switches**
  - Packet misrouting
  - Packet dropping (Switch blackhole)
  - Packet injecting

✓ These attacks affect the entire network!

Protection of SDN data plane is more important than traditional networks! [1],[2]

Related Work

**Byte consistency check** *(SPHINX [M.Dhawan et al. / NDSS’15]*)

- Compare the entire view of network *(Flow graph)* with the amount of packet transfer
- Construct flow graph with collected route information *(FLOW_MOD messages)* and calculate expected path

**Issue**

Verification accuracy depends on the controller performance

Statistics data*(STATS_REPLY)*: *asynchronous* message from switches respond to STATS_REQUEST

 ✓ *Depends on the timing of STATS_REQUEST message received*
Background Technology

Time4 [T.Mizrahi et al./INFOCOM’16]

- Schedule network updating at the same time with Scheduled Bundle
- Utilizes IEEE1588 Precision Time Protocol (PTP)
  - Enables nanosecond order time synchronization using hardware-timestamping enabled NIC module.
- **Without depend on the controller performance**
  - It is scalable!

Our Challenge

Solve the issue of Byte consistency check with **statistics gathering at the same time**
Our method
Forwarding State Verification with Timed Statistics Gathering

- **Schedule the timing of statistics gathering at the same time to each switch**
  - Synchronize clocks between Verifier and OpenFlow switches
  - Enables to verify without depending on the performance of the controller

**Performing verification without depend on the performance of the controller**

**Statistics (STATS_REPLY)**
Get actual transfer statistics of all switches at the same time with **Scheduled Bundle**.
Assumption

- The controller applications are trusted
- Majority of switches are legitimate
  - These assumption similar to SPHINX
- Focus on the data plane security
  - Knows reliable physical topology
  - Control-plane security is well studied by other work [3]

Sequence of Verification

1. **SPHINX (Existing solution)**
   - Calculate current path

2. **Proposal Method**
   - Gather statistics at the same time

3. Verify transfer state consistency
   - Verify with the moving average of the statistics (Similarity Index)
   - Verify with the statistics gathered at the same time
1. Calculate Current Path

- Uses a flow graph (network assumed by a trusted controller) and physical topology information
  - Uses match fields and instruction: src/dst MAC address, src/dst IP address, and in/out port
2. Statistics Gathering with Scheduled Bundle

• Scheduled Bundle
  [T. Mizrahi et al. / INFOCOM’16][OpenFlow 1.5]
  • A generic method to schedule any OpenFlow message
  • Included in OpenFlow 1.5

✓ Schedules STATS_REQUEST message with Scheduled Bundle to each switch
  • Enables gathering statistics at the same time
  • Mainly uses byte_cnt and match field
  • Uses the moving average of last four statistics in verification
    • Eliminates the effects of switch performance

At $T_s$, processes STATS_REQUEST message

Send Bundle Commit contains schedule execution time $T_s$
3. Verification (1)

• Verify each traffic flow with two points of view
  • Whether the flow statistics of switches included in the current path reports similar value between each switches
    • The margin calculate by threshold (input value)
    • Detects malicious packet delaying and packet dropping
  • Whether the flow statistics of switches not included in the current path reports zero
    • Detects malicious packet injecting and misrouting
3. Verification (2)

• Difference of SPHINX: Reference value of verification

• **SPHINX**: The average of Similarity Index each switches
  • (Similarity Index: Moving average of byte transfer statistics)

• **Our method**: The statistics of the verification already passed switches
  • Considers the difference of the statistics values which occurs by propagation delay
Example of our method’s algorithm behavior

- Case of all switches are legitimate:

  ![Diagram showing legitimate switched behavior]

  ByteDiff = 100  Compare  ByteDiff = 98  Compare  ByteDiff = 96

  - Case of switch s2 is compromised switch (perform blackhole):

    ![Diagram showing compromised switch behavior]

    ByteDiff = 100  Compare  ByteDiff = 98  Compare  ByteDiff = 96

    Detect attacks by honest downstream switch

    Raise an alarm
Experiment Environment

- Emulated Mininet network
  - Use minimal topology (linear topology, 3 switches and 3 hosts)
  - Separated the controller application host and the Mininet host
    - To mitigate the impact of resource conflicting

Same host to emulate time synchronization with accurate time

Controller

Controller Application

Verifier

Floodlight v1.1

OpenFlow Proxy (implemented our method)

Time synchronized OpenFlow switches

h1

h2

h3

ofsoftswitch13_EXT-340
Experiments

• Scheduling Accuracy
  • Measured the accuracy of scheduling emulated testbed (Mininet)

• False-Positive Rate
  • Measured the probability of detecting normal TCP traffic as attacks
  • Check the impact to verification when controller performance occurs degradation.

• False-Negative Rate
  • Measured the probability of the lack of genuine alarm
  • Check the impact to verification when controller performance occurs degradation.

• Simulated the variation of the controller performance by delay (d)
  • Insert before sending STATS_REQUEST from the verifier.
Scheduling Accuracy of our testbed

- Measured the accuracy of scheduling emulated testbed (Mininet)
  - Measured the difference between actual execution time and scheduled execution time

This environment is sufficient to evaluate our method

NTP synchronization accuracy: millisecond order
Scheduling error in NTP synchronization environment: max millisecond order

Less than 0.3 millisecond in the 90 percentiles at all switches
False-Positive Rate: SPHINX vs our method

- Generate TCP flows from h1 to h3 with iperf
- Simulated the variation of the controller performance by delay (d)
- Occurs false-positive when threshold (the margin of statistics value) decrease
False-Positive Rate: SPHINX vs our method

Result

**SPHINX**’s False-Positive Rate

- increases false alarms when controller performance degrade

**Our method**’s False-Positive Rate

- Not increase false alarms!

Our method without depend on controller performance!
False-Negative Rate: SPHINX vs our method

- Generate TCP flows from h1 to h3 with `iperf`
- Emulate malicious behaviors by compromised switches
  - performed packet drop on s2-s3 link with link loss rates (2%, 4%, 6%)
- Occurs false-negative when threshold (the margin of byte count value) increase
False-Negative Rate: SPHINX vs our method

Result

Our method’ false-negative rate is not much different from SPHINX.

(From FP rate experiment)
Our method can set lower threshold than SPHINX

the false-negative rate equivalent to SPHINX by tuning threshold.
Discussion: Future Work

• Evaluates in real-world environment

• Reduce the statistics gathering overhead of Scheduled Bundle
  • Improves Scheduled Bundle to enable periodical execution scheduling

• Distributed Controller environment (such as ONOS)
  • Improves the method to consider a link specific delay
Conclusion

• Background
  • Protection of SDN data plane is more important than traditional networks
  • SPHINX’s verification accuracy depends on the controller performance

• Proposal method:
  Forwarding state verification with scheduled statistics gathering
  • Key idea: Schedule the timing of statistics gathering at the same time to each switches
  • Detects attacks by compromised switches without depending on the controller performance

• Experiment Result
  • Confirm the false positive rate of our method is lower than SPHINX even if the controller performance decrease

Thank you for listening!