FleXam: Implementing Security Applications in OpenFlow Controller

Sajad Shirali-Shahreza & Yashar Ganjali

Department of Computer Science
University of Toronto

HotI 2013 – San Jose, CA

yganjali@cs.toronto.edu
http://www.cs.toronto.edu/~yganjali
Anti-Outline

• First, what this paper is not about:
  • Unlike what the title might suggest, the focus of this work is not security!

• What is it about then?
  • Simple observation
  • Simple question
  • Simple extension to OpenFlow
  • Simple example as a case study

To Summarize:
The paper is not about security; and, it is very simple! 😊
Observation

• It might be desirable to implement security (or similar) services as SDN controller applications
  • Works for some (e.g. small) networks
  • Can reduce cost and complexity

• Need packet level information
  • Botnet detection: connection patterns, packet contents
  • Port scan detection: packet header, inter-arrival times
  • Worm detection: packet contents
Separation of Controller and Data Path

Data Path Elements

Event-based Messages

Flow Statistics

Packet-In Messages

Controller
Controller Information Channels

**Event-based Messages**
- Generated by switches
- Information about network structure, and topology changes

**Flow Statistics**
- Collected by switches; pulled by the controller
- Flow level information

**Packet-In Messages**
- Generated by switches
- Flow level information; limited packet level
Packet Level Information in OpenFlow

- Lack of packet level information is not intrinsic in SDN
- But a side-effect of how OpenFlow works

- OpenFlow is mainly designed to deal with flows rather than individual packets
- Still, we can access packet level information to some extent.
Packet Level Information in OpenFlow

- **Option 1:** Do not install flow rule
  - Every packet of the flow will be sent to the controller

- **Option 2:** Divert packets to a monitoring device
  - Suggested in original OpenFlow white paper
Limitations …

- **Option 1:** install no rule
  - Controller sits on the packet delivery path
    - Potential bottleneck
    - Increased packet delivery times
  - Switch may buffer the packet
    - Only parts of each packet may reach the controller

- **Option 2:** divert traffic to monitoring device
  - High network overhead
  - Unnecessary overhead
    - E.g. need packet header, receive full packet
  - Complex monitors required
    - Increased network cost and complexity
A Question

- Existing information channels either
  - Provide no packet level information
  - Need to relay all packets

Question:
Is there a simple mechanism to create a tunable channel for packet level information?
FleXam: Flexible Sampling in OpenFlow

- Per-flow sampling as a new information channel
  - Gives the controller access to packet-level information

- The controller defines
  - Which flows need to be sampled
  - Inside a given flow how samples are selected
  - What part of each packet is selected
  - Where the samples are sent to

- Simple, yet flexible
  - ... for different applications
Stochastic Sampling in FleXam

- Select each packet with a fixed probability of $\rho$ and forward to the control plane.
Deterministic Sampling in FleXam

- Select $m$ consecutive packets out of every $k$ consecutive packets, skipping the first $\delta$
Deterministic Sampling – Cont’d

- $m=1$: normal one out of $k$ packets
- $m > 1, k=\infty$: only the first $m$ packets
  - Suitable for applications such as traffic classification
- $\delta > 0$: exclude short flows
  - Suitable for elephant flows
OpenFlow Specification

- New action: OFPAT_SAMPLING
  - Can be easily added to current OpenFlow implementations
  - No overhead for flows with no sampling
- Six parameters
  - scheme: which packet parts should be sent
  - $m$, $k$, and $\delta$: deterministic sampling parameters
  - $\rho$: stochastic sampling parameter
  - destination: where should be sent to
Switch Implementation

- Stochastic sampling:
  - Generate a random number
  - Select the packet if it is less than $\rho$

- Deterministic sampling:
  - ReuseReceived Packets Counter
    - No need to new counter
  - Select packet if:
    $$((\text{Received\_Packet\_Counter} - \delta) \mod k) < m$$

- Both can be executed in the data path at line rate
Applications

• Various potential applications
  • Traffic classification
  • Quality of service
  • Diagnostics and troubleshooting

• In depth look at …
  • Port scan detection; and
  • Elephant flow detection

• Accuracy vs. load trade-off
Case Study: Port Scan Detection

- Example of FleXam used for security applications
- Threshold Random Walk algorithm
  - Assume the probability of a failed connection is
    - Relatively low for a benign host
    - High for attacker hosts
  - Maintaining an attacker likelihood ratio for each host
    - Increase it when a new connection fails
    - Decrease it when a new connection succeeds
  - Mark a connection successful if we observe
    - At least two packets from a UDP connection
    - A non TCP_SYN packet from a TCP connection
Evaluation Setup

- Developed an OpenFlow switch simulator
  - Provides an API similar to NOX

- Based on the dataset used by Mehdi et al. [2]
  - Separate attack and benign datasets
  - More details in the paper

- We created 20 different trials
  - Inserting the attack data at different times inside benign data
Effect of Sampling on Port Scan Detection

- **Flow-shortening:** only small fraction of flow packets are observed
  - Small flows: probably see only one packet
  - If only see SYN packet
    - Mark connection as failed
    - Decreases accuracy with false positives

- **Flow-reduction:** only some flows are observed
  - Uniform per-packet sampling
    - Miss many short flows, especially attack flows that are single packets
  - Miss some failed connections
    - Decreases accuracy with false negatives
Resolving Flow-Shortening Problem

- Receive a sampled packet from a new connection
  - TCP non SYN packet $\rightarrow$ mark connection as successful
  - UDP packet or TCP_SYN packet:
    - Ask the switch to send the next packet of this flow: Install an exact match flow rule with deterministic sampling with $m=1$ and $k=\infty$
    - If receive any packet later on, mark connection as successful
    - Otherwise, mark it as failed after a timeout
Resolving Flow-Reduction Problem

- Small number of flows carry most of the traffic

- In our traffic data:
  - Flows with more than 50 packets
    - 6% of total flows
    - Carry 96% of total packets
    - Carry 95% of total bytes

- Identify and exclude large (elephant) flows from sampling
  - Spend sampling budget on small flows
Resolving Flow-Reduction Problem

![Graph showing CDF (Cumulative Distribution Function) of Flow Size (packets) with lines for Total Traffic %, Flow Count %, and Total Packet %.]
Elephant Flow Detection

- Sample all unknown traffic with a given rate of $\rho$

- Identify elephant flows
  - If more than $\varepsilon$ samples received from a given flow

- Exclude elephant flows from sampling
  - Install an exact match flow rule to route them, without any sampling action

- No need to complicated switch modifications
Network Overhead

- Overhead is low for a wide range of values
  - E.g. it is less than 0.4% for $\rho<10$ and $\varepsilon<10$
Accuracy

- Attack rates of 10, 100 and 1000 packets/s
  - All scanners were detected for any pairs of \((\rho, \varepsilon)\)

- For 100% accuracy in other two cases:
  - \((\rho = 50\%, \varepsilon = 3)\) has overhead of 0.7% for an attack rate of 0.1 pps
  - \((\rho = 5\%, \varepsilon = 4)\) has overhead of 0.25% for an attack rate of 1 pps
Uniform Sampling Accuracy and Overhead

- Minimum sampling rate (and overhead) for 100% accuracy:
  - 60% for attack rate of 0.1 pps
  - 7% for an attack rate of 1 pps
  - 1% for higher attack rates (10, 100, and 1000 pps)
Overhead Comparison

- Network overhead for 100% accuracy

<table>
<thead>
<tr>
<th>Method</th>
<th>Attack Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.1</td>
</tr>
<tr>
<td>Our Method</td>
<td>0.7%</td>
</tr>
<tr>
<td>Uniform Sampling</td>
<td>60%</td>
</tr>
<tr>
<td>Mehdi et al. [2]</td>
<td>1.1%</td>
</tr>
<tr>
<td>Reactive OpenFlow Routing (No port scan detection)</td>
<td>0.7%</td>
</tr>
</tbody>
</table>
FleXam and Network Cost/Complexity

- In small networks
  - The controller could also be the monitoring device
  - Eliminating the need for a separate monitoring device
    - Significant reductions in cost and complexity

- For large networks
  - Samples can be directed to different monitoring devices
  - Reduce the load of monitoring devices by only sending the traffic they are interested in, and not all traffic

- Network overhead is tunable by the controller
  - Change parameters in real time, if needed
Conclusion

- Proposed FleXam
  - Flexible sampling extension for OpenFlow
  - Enables the controller to access packet-level information
  - Flexible for different applications
  - Yet simple enough to be implemented entirely in switch data-path and operate at line rate
- Demonstrated how to implement port scan detection with FleXam
  - An example of security applications that need access to packet-level information
- Other applications remain as future work
Thank You!

Questions?