HEX Switch:
Hardware-assisted security extensions of OpenFlow

Taejune Park / KAIST / taejune.park@kaist.ac.kr
Zhaoyan Xu / StackRox Inc. / z@stackrox.com
Seungwon Shin / KAIST / claude@kaist.ac.kr
Software-Defined Networking

- Centralized management
- Dynamic traffic engineering
- Programmable network operation
- High-compatibility with virtualized environments
Software-Defined Networking

- Centralized management
- Dynamic traffic engineering
- Programmable network operation
- High-compatibility with virtualized environments

Security is still required
Security in Software-Defined Networking

Control-Plane Layer

Network Control Apps.
- Network Application
- Security App.

Security Apps.
- Security Application

Standard Protocol (e.g., OpenFlow)

Data-Plane Layer

Middle-box
Security in Software-Defined Networking

- Security applications on a control plane
- Applying security features network-wide
- Cheap price
- Easy to manage
Security in Software-Defined Networking

- **Security applications on a control plane**
  - Applying security features network-wide
  - Cheap price
  - Easy to manage

- **Limitation**
  - Simple security only available
  - Slow-path for inspection
  - Controller overhead
Security in Software-Defined Networking

- Middle-boxes on a data plane
  - Better performance
  - Rich features such as payload inspection
  - No controller overhead

Control-Plane Layer
- Network Control Apps.
  - Network Application
- Security Apps.
  - Security Application

Data-Plane Layer
- Standard Protocol (e.g., OpenFlow)

Middle-box

Network Application

Security Application
Security in Software-Defined Networking

- **Limitation**
  - Network overhead by traffic detouring (Taking extra hops)
  - Require flow steering for NFs
  - Additional control channels for NFs

- Middle-boxes on a data plane
  - Better performance
  - Rich features such as payload inspection
  - No controller overhead
## Summary

<table>
<thead>
<tr>
<th>Category</th>
<th>SDN Applications</th>
<th>Middle-boxes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flexibility</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Management</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Deployability</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Performance</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Functionality</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>
Related works:
Extending SDN architecture to support security


Related works:
Extending SDN architecture to support security

- Their security functions are not fully consolidated into a data plane
- Application module, Tap-based interface...


Related works:

Extending SDN architecture to support security

- In essence, they are NOT different from the middle-box structure!
- It's just a scale down!

Related works:

**UNISAFE: A union of security actions for software switches**


- Fully integrated security functions into a data plane, *not modular one*
- Security functions as a set of OpenFlow actions

**UNISAFE (based on Open vSwitch)**

- Lookup Flow table
- Execute actions
- Security actions

**Flow table**

<table>
<thead>
<tr>
<th>MATCH</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow_A</td>
<td>sec_dos(mbps=100), output:2</td>
</tr>
<tr>
<td>Flow_B</td>
<td>sec_dos(mbps=500), sec_scan(...), output:3</td>
</tr>
</tbody>
</table>
Security actions of UNISAFE

- High-compatibility with common OpenFlow actions
  - \( \text{actions} = \text{sec\_dos}(\text{mbps}=1000), \text{set\_nw\_src}(\ldots), \text{output}: 2 \)

- Fine-grained security enforcement per a flow
  - \( \text{in\_port}=1, \text{nw\_src}=10.0.0.1, \text{tp\_dst}=80, \text{actions} = \text{sec\_dos}(\ldots), \ldots \)
  - \( \text{in\_port}=2, \text{nw\_dst}=10.0.1.2, \text{actions} = \text{sec\_dpi}(\ldots), \ldots \)

- Easy configuration for a security service chaining
  - \( \text{actions} = \text{sec\_dos}(\ldots), \text{sec\_scan}(\ldots), \text{sec\_dpi}(\ldots), \ldots \)
Performance in UNISAFE

- Achieve line-rate latency for all security

- But, lack of throughput in some actions
  - Payload Inspection (DPI) throughput
    - Throughput less than 100Mbps on 1Gbps
Performance in UNISAFE

- Achieve line-rate latency for all security
- But, lack of throughput in some actions
  - Payload Inspection (DPI) throughput:
    - Throughput less than 100Mbps on 1Gbps

**Challenge 1:** Performance limitation
Security operation in UNISAFE

- Manual operation for security violations by an administrator
Security operation in UNISAFE

- Manual operation for security violations by an administrator

Challenge 2: Security operation
HEX Switch: Hardware-assisted security extensions of OpenFlow

- Hardware-based approach for UNISAFE
- Using NetFPGA
- Providing line-rate performance with configurability

Security Actions
Security Policy
Controller communication
Design

• **Security Processor** between the packet processing sequence.
• Six-stages pipeline: Mainly consist of *data storage* and *inspection logic*
• Flow table controller forwards *flow keys, stats and action key* after matching
Design

- **Security Processor** between the packet processing sequence.
- Six-stages pipeline: Mainly consist of data storage and inspection logic
- Flow table controller forwards *flow keys, stats and action key* after matching

![Diagram](image-url)
Design

- **Security Processor** between the packet processing sequence.
- Six-stages pipeline: Mainly consist of *data storage* and *inspection logic*
- Flow table controller forwards *flow keys, stats and action key* after matching
Design

- **Security Processor** between the packet processing sequence.
- Six-stages pipeline: Mainly consist of data storage and inspection logic
- Flow table controller forwards *flow keys, stats and action key* after matching
Security Action Processing

- All security actions are performed in parallel
- Forward the data storage data to inspection logic through the wide data bus.

**Challenge**

- Pattern list for payload inspection requires width bandwidth

=> Transfer the address first and read directly memory
After Processing: Applying Policy

- Actions can handle violated packets according to a policy
  - e.g., actions=sec_dos(mbps=1000,policy=redirect:2)
    => If the current bps exceeds 1000 Mbps, redirect the flow to port 2.

- Four polices
  - **Neglect**: Ignores the violation
  - **Alert**: Send an alert msg to a controller
  - **Discard**: Terminates the packet processing and drop the packet
  - **Redirect**: Forward packets to an alternative port
Communication with a controller

- By the host device with its software
- The host device and the HEX switch are bound by the device driver
Communication with a controller: Transferring an alert message

- The device driver reads the registers and the HEX handler transfers it to a controller through a OpenFlow channel.
- A controller provides a handler API to process the alert message.
Communication with a controller:

**Deploying security actions:**

- The security actions are deployed by `flow_mod` messages
- Security actions are compatible with common OpenFlow actions
Challenge in flow-level security deployment

- The flow-level security cannot represent a security policy across multiple flows

- Simple example:

  Flow A
  \[\text{800Mbps}\]

  Flow B
  \[\text{700Mbps}\]

  \begin{tabular}{|c|c|}
  \hline
  Match & Actions \\
  \hline
  Flow A & \text{sec\_dos(mbps=1000),output:1} \\
  Flow B & \text{sec\_dos(mbps=1000),output:2} \\
  \hline
  \end{tabular}

  The total incoming bandwidth from Flow A/B evidently exceeds 1000 Mbps, but the DoS detectors never trigger an alert!
Action Clustering

- Security actions have a cluster ID in their parameter
- The actions that use the same cluster ID are considered to belong to the same cluster
- The clustered action works as the integrated single action across different flow rules
- Implementing by sharing the data storage by the cluster map

<table>
<thead>
<tr>
<th>Match</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>sec_xyz (id = 10 ...)</td>
</tr>
<tr>
<td>B</td>
<td>sec_xyz (id = 10 ...)</td>
</tr>
<tr>
<td>C</td>
<td>sec_xyz (id = 10 ...)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Applying Action Clustering

- Applying the action clustering to the previous example

DoS detector can successfully detect the bandwidth excess and alert this.
Implementation

- NetFPGA-1G-CML

- Based on Reference NIC and OpenFlow switch from the NetFPGA project ([https://github.com/NetFPGA](https://github.com/NetFPGA))

- Support DoS Detector and Deep Packet Inspector (Payload inspector)
Evaluation

- Measure throughput and latency
  
  1) Performance of the HEX switch
  
  2) Performance of simple forwarding by the normal OpenFlow switch
  
  3) Performance of OVS based implementation (i.e., UNISAFE)
Evaluation Result

• Throughput

HEX & Simple Fwd. • Latency

UNISAFE
Conclusion

• The HEX switch that embeds security functions
  • Using NetFPGA
  • As as a set of actions
  • Support security policy and controller APIs
• Achieves line-rate performance without overhead.
Thank you!
Questions?