Extensible and Scalable Network Monitoring Using OpenSAFE

Jeffrey R. Ballard    Ian Rae    Aditya Akella
Outline

1. Background
   Network monitoring
   How monitoring is done today

2. OpenSAFE and ALARMS
   OpenSAFE
   ALARMS
   Rule Aggregation
   Distribution

3. Implementation
   Mapping to OpenFlow
   Switch Example

4. Conclusion
Motivation

We want to monitor the network.

Specifically, we want to allow administrators to easily:

- collect network usage statistics
- detect intrusions
- provide forensic evidence
Challenges

Middleboxes are commonly used, however, they present challenges...

1. Speed
2. Cost
3. Flexibility
   1. Setup: rewire
   2. Change: rewire
   3. Add new middlebox: rewire

...making them ill suited for network monitoring.
Example: College of Engineering

- Red links = 10 Gbps
- White links = 1 Gbps

Connections to the College:
- 2 x 10 Gbps links
- 22 x 1 Gbps links
How do people actually do it?

Mirror (or tap) an interesting network interface to another switch port, then listen to that port with something like Snort.

Advantage over a middlebox: monitoring has no impact on the production traffic and routes.

Disadvantages: the traffic can run you over, and it’s still hard to add new detectors.
What it looks like today

Network A

Network B

Firewall

Monitoring Device
What we want to do

Network A

Network B

Firewall

Network B

Programmable Network Layer

Monitoring Device 1

Monitoring Device 2

...
OpenSAFE uses a programmable network fabric to...

- Selectively match network flows
- Arbitrarily direct network flows to other switch ports at line rate
- Direct exceptions to a software component
- Enable the use of commodity network hardware
Why not implement it in software?

We could use something like Click to dynamically manage detectors.

Major problem: software is not fast enough!
Solution: Hardware!

Easiest: Custom ASICs

1. Expensive
2. Non-standard
3. Potentially hard to configure

But we have something that can do this...
While OpenSAFE would be compatible with any programmable network fabric, we implemented OpenSAFE in OpenFlow since it is available today.

The key elements are:

1. speed
2. heterogeneity
3. flexibility
4. cost
Example OpenSAFE Layout

Network A

Network B

Firewall

Network B

OpenFlow

Controller

Snort

dSniff

Decryption
ALARMS: A Language for Arbitrary Route Management for Security

Basic building blocks are paths of:

- **Inputs**: copy of traffic from a mirror switch port
- **Selects**: restricts the set of traffic for this rule
- **Filters**: pass the traffic through an application
- **Sinks**: where to finally direct the traffic

Combining these gives us a rich set of configurations.
Simple Example

We will use the following example over the next few slides:

Mirror
Port: 80

Counter

TCP Dump

Take all TCP port 80 traffic, send it to a counter, and then send it to a machine running tcpdump.
A path is:
A source switch port with selection criteria
... which goes into zero or more filters
... then out to one or more sinks
OpenSAFE Schematic

OpenFlow Switch

Input

Filter₁

Filter₂

Filterₘ

Sink₁

Sinkₙ

OpenFlow Controller
In OpenSAFE all switch ports are named.

Logically, ALARMS articulates paths of named switch ports.
Revisiting our example

Mirror -> Counter -> TCP Dump

... becomes ...

mirror[http] -> counter -> tcpdump;
Let's get some more paths

mirror[http] -> counter -> tcpdump;
Let’s get some more paths

```
mirror[http]  ->  counter  ->  tcpdump;
```

```
mirror[https]  ->  decrypt  ->  counter  ->  tcpdump;
```
Waypoints

As more rules are added, often the rules follow the same paths making rule management difficult.

Solution:

Waypoints are *virtual* destinations for paths.
Waypoint example

mirror[https] -> decrypt -> web;
mirror[http] -> web;
web -> counter -> tcpdump;
In ALARMS, multiple destinations are easy:

```
mirror[http] -> {ALL, tcpdump1, tcpdump2};
```
Distribution rules

When parallel filters or sinks are used, distribution rules describe how traffic flows should be spread.

Rules include:

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Any</td>
<td>Randomly pick a switch port</td>
</tr>
<tr>
<td>All</td>
<td>Replicate packet to all switch ports</td>
</tr>
<tr>
<td>Round Robin</td>
<td>Cycle through the switch ports</td>
</tr>
<tr>
<td>Hash</td>
<td>Apply a hash function</td>
</tr>
</tbody>
</table>
web -> \{ALL, tcpdump1, tcpdump2\};
Mapping the language into OpenFlow

We want to handle lots of traffic, so need high performance.

Hardware is fast. Software is slow.

Install as many precomputed flow entries as possible.

However, when the hardware does not support functions we must go to software. In OpenFlow this includes Any, Round Robin, and Hash distribution rules.
How it works under the hood

Starting with the last path diagram we had before...
How it works under the hood

Mirror → OpenFlow Switch → Decryption → TCP Dump 1 → TCP Dump 2
How it works under the hood

Mirror → OpenFlow Switch → Web

TCP Dump 1 → Decryption
TCP Dump 2 → Decryption
How it works under the hood

- Mirror
- OpenFlow Switch
- TCP Dump 1
- TCP Dump 2
- Decryption
Related Work: Ethane

Ethane (the predecessor to OpenFlow) is an enterprise-wide security solution.

The focus here is to insert a tool just at the border, optimized for the border.
Related Work: Policy-aware switching

*Policy-aware switching*, proposed by Joseph et al. is somewhat similar to Ethane.

It removes the centralized controller, and has each switch determine the next hop.

Also, the policy specification language, like Ethane, is centered around deciding appropriate paths for a flow.
What next?

In the future, we’d like to expand our system by exploring:

• incorporating dynamic feedback from filters and sinks
• precomputing more dynamic distribution rules
Conclusion

OpenSAFE greatly simplifies high-speed network monitoring. It is also:

- Cost effective by using commodity hardware
- Flexible and easy to modify
- Capable of operating at high line rates