Enabling Practical SDN Security Applications with OFX (The OpenFlow eXtension Framework)

John Sonchack, Adam J. Aviv, Eric Keller, and Jonathan M. Smith
Outline

Introduction
Overview of OFX
Using OFX
Benchmarks
Basic Networking: Forwarding and Routing

Packet Forwarding

Route Computation
SDNs: Networking in Two Planes

Route computation

Packet forwarding

Control Plane

Data Plane
OpenFlow: A Protocol to Manage Switches

Route computation

Flow rules to implement routes

Packet forwarding

Control Plane

Data Plane
OpenFlow: A Protocol to Manage Switches

Control Plane

Route computation
Flow rules to implement routes

Assumption: Interactions between the control plane and data plane are *infrequent*.

Data Plane

Packet forwarding
SDNs for Network Security

SDNs for **Dynamic** Network Security

- **Traffic Declassification**
- **Access Control**
- **DDoS Defense**
- **Bot Detection**

**OpenFlow**

**Advanced Processing**

**Route for flow**

**Packet from new flow**

**Data Plane**

**Control Plane**
SDNs for **Dynamic** Network Security: Flow Monitoring


Collect flow records without routing through a middlebox.

Install byte counting rule

Packet from new TCP flow
SDNs for **Dynamic** Network Security: Traffic Declassification

Enforce access control on **tagged data leaving the network**.

SDNs for **Dynamic** Network Security

**Control Plane**

- Traffic Declassification
- Access Control
- DDoS Defense
- Bot Detection

**Data Plane**

- Advanced Processing
- Route for flow
- Packet from new flow

---

**OpenFlow**

1. **Traffic Declassification**
2. **Access Control**
3. **DDoS Defense**
4. **Bot Detection**
SDNs for **Dynamic** Network Security

**Control Plane**

- Traffic Declassification
- Access Control
- Bot Detection
- Advanced Processing
- Route for flow

**Data Plane**

- Packet from new flow

Assumption: Interactions between the control plane and data plane are *infrequent*.
Obstacle: Low Throughput Control Path

130 million packets/second!!!!*

*can only forward 500 pps to controller.


Obstacle: Centralized Control Plane
Our question: How Can We Make SDNs More Practical?

- Traffic Declassification
- Access Control
- DDoS Defense
- Bot Detection

Control Plane

Data Plane
The General Approach: Switch Level Security

- Traffic Declassification
- Access Control
- DDoS Defense
- Bot Detection

Control Plane

Data Plane
Previous Work: Security Functionality in the Forwarding Engine

Build new switch chips that support security applications

Our insight: Leverage Switch CPUs

Run security logic on the switch CPUs
OFX: A Framework for Application-Specific Switch Extensions

Each application can load custom functionality into switches. At runtime!
Outline

Introduction

Overview of OFX

Using OFX

Benchmarks
OFX at a High Level
OFX at a High Level
OFX at a High Level

Controller interface

OFX Extension Module

Switch-level logic
OFX at a High Level
**OFX at the Switch Level**

- OFX modules use filters to select packets that they need to process.
- OFX modules process packets with custom handler.
- OFX installs corresponding rules onto OFX tables.

**Diagram:**
- **Ingress Packets** flow into the switch.
- Packets are filtered through **OFX Filtering Tables**.
- Packets are processed by **OFX Module** with a custom **Packet Handler**.
- Packets are directed to **Controller-managed forwarding tables**.
- Egress Packets leave the switch.

**Key Concepts:**
- **OpenFlow Switch Agent**
- **OFX Module**
- **Packet Handler**
- **OFX Filtering Tables**
- **Controller-managed forwarding tables**
Outline

Introduction
Overview of OFX
Using OFX
Benchmarks
Refactoring OpenFlow Applications to use OFX

class DeclassifierApp(app_manager.RyuApp):

    def __init__(self, *args, **kwargs):
        super(SimpleSwitch13, self).__init__(*args, **kwargs)
        self.permissionsDb = dbServer.connect()
        self.monitoredServers = []
        self.switchIds = []

    def switch_up_handler(self, switch):
        self.switchIds.append(switch.id)
        ...

    def packet_handler(self, switch, pkt):
        action = self.compute_next_hop(pkt, switch)

        if pkt.src in self.monitoredServers:
            permission = check_permission(pkt)
            if permission:
                switch.send_packet(pkt, action)
                switch.add_flow(pkt.src, pkt.dst, action)
            else:
                resetPkt = build_reset(pkt)
                switch.send(resetPkt)
                switch.add_flow(pkt.src, pkt.dst, DROP)
            else:
                switch.send_packet(pkt, action)
        ...

OFX Declassifier Module
Refactoring OpenFlow Applications to use OFX

```python
import OFXLib

class DeclassifierApp(app_manager.RyuApp):
    def __init__(self, *args, **kwargs):
        super(SimpleSwitch13, self).__init__(*args, **kwargs)
        self.permissionsDb = dbServer.connect()
        self.monitoredServers = []
        self.switchIds = []
        self.declassifierModule = OFXLib.load_module("dec_module")
        self.declassifierModule.permissions = self.permissionsDb

    def switch_up_handler(self, switch):
        self.switchIds.append(switch.id)
        OFXLib.install(switch, self.declassifierModule)

    def packet_handler(self, switch, pkt):
        action = self.compute_next_hop(pkt, switch)
        switch.send_packet(pkt, action)
        ...
```
Outline

Introduction
Overview of OFX
Using OFX
Benchmarks
Benchmarking OFX

How much raw overhead is there for processing packets with OFX?

How do OFX based security applications perform, compared with Middlebox and OpenFlow implementations?
OFX Benchmark: Packets Per Second

Packet handler in controller
Packet handler in OFX module

Log^{10} Scale

Packets per Second

Packet Size

10,000
1,000
100
10
1

64 128 256 512 1024 1500

100 PPS @ MTU
45,000 PPS @ MTU
Benchmarking OFX

How much raw overhead is there for processing packets with OFX?

How do OFX based security applications perform, compared with Middlebox and OpenFlow implementations?
### Benchmark: Declassifier

**Packet Drop Rate**

<table>
<thead>
<tr>
<th>Implementation</th>
<th>Frequent arriving flows</th>
<th>Median</th>
<th>High bandwidth flows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Middlebox Proxy</td>
<td>0.1%</td>
<td>0.1%</td>
<td>20.4%</td>
</tr>
<tr>
<td>OpenFlow</td>
<td>97.5%</td>
<td>88.2%</td>
<td>0.1%</td>
</tr>
<tr>
<td>OFX</td>
<td>5.1%</td>
<td>3.2%</td>
<td>0.1%</td>
</tr>
</tbody>
</table>

- **OpenFlow implementation limited by flow arrival rate**
- **Proxy implementation limited by bit rate**
- **OFX implementation performed well in all workloads**

<table>
<thead>
<tr>
<th>Workload Name</th>
<th>Frequently arriving flows</th>
<th>Median flows</th>
<th>High bandwidth flows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow Inter-arrival Period</td>
<td>0.0015 Seconds</td>
<td>0.015 Seconds</td>
<td>0.15 Seconds</td>
</tr>
<tr>
<td>Average Transmission Bandwidth</td>
<td>19.75 Mbps</td>
<td>43.57 Mbps</td>
<td>970.99 Mbps</td>
</tr>
</tbody>
</table>

In the Paper

OFX API and Implementation Details

Application Specific Modules

Enhanced Switch API Modules

More benchmarks

Running on unmodified OpenFlow hardware!

DDoS Defense

Bot Detection

TCP Handshake Validation

Push Based Alerts

New TCP Flow

Condition Reached

More benchmarks

Running on unmodified OpenFlow hardware!
Thank You

**OFX: The OpenFlow Extension Framework**

OFX lets OpenFlow security applications push parts of their control plane logic down to switch CPUs, which can greatly improve performance and scalability on existing hardware and software.