FRESCO: Modular Composable Security Services for Software-Defined Networks

Seungwon Shin, Phil Porras, Vinod Yegneswaran, Martin Fong, Guofei Gu, and Mabry Tyson
SUCCESS LAB, Texas A&M and SRI International
Contents

• Background
  – SDN and OpenFlow

• FRESCO
  – Design
  – Use cases
  – Evaluation

• Summary
Problem of Legacy Network Devices

- Complicated and Closed platform
  - Complicated S/W and ASIC
  - Vendor specific
- Inflexible: Hard to modify (nearly impossible)
- Non-extensible: Difficult to support emerging technologies
  - E.g., VM mobility

- New proposal: Software Defined Networking
  - Separate the control plane from the data plane
Software Defined Networking (SDN)

• Three layers
  — Application layer
    • Application part
    • Implements logic
  — Control layer
    • Kernel part
    • Runs applications
  — Infrastructure layer
    • Data plane
    • Network switch or router
SDN Operation

1. L2 Forwarding application
2. Controller (e.g., NOX)
3. SDN Controller
4. A → B: Forward
5. Flow Table in SDN Switch

Host A

SDN Switch

Host B
Killer Application of SDN?

- Reducing energy in data center networks
- Dynamic virtual machine migration in cloud networks
- .... diverse network applications

- What about security?
  - Can SDN enable new capabilities to improve network security?
Exemplar SDN Security Apps

- Security functions can be applications of SDN
  - Firewall
  - DDoS detection
  - Scan detection
  - Reflector net
  - Tarpit
  - Dynamic quarantine
  - and more...

```python
import logging
from nox.lib.core import *
import nox.lib.openflow as openflow
from nox.lib.packet.ethernet import ethernet
from nox.lib.packet.packet_objects import *

log = logging.getLogger('nox.coreapps.examples.demo')

class demo(Component):
    def __init__(self, ctxt):
        pass

    def create_and_enforce_policy(self, dpid, policy_type, outport_find, inport, outport, bufid, buf, packet):
        if outport_find == 0:
            print('DBG: No Specific Out Port: Flooding')
        elif policy_type == 'ARP':
            print('DBG: ARP packet')

        self.send_openflow(dpid, bufid, buf, openflow.OFPP_FLOOD, inport)

        if policy_type == 'REQ':
            flows = extract_flow(packet)
            flows = {
                core.IN_PORT: inport,
                core.DL_TYPE: ethernet.IP_TYPE,
                core.NW_PROTO: ipv4.ipv4.TCP_PROTOCOL,
                core.NW_SRC: '10.0.0.2'}
```

(1) Host A sends packet to Host B
(2) Switch asks a controller form a flow rule
(3) F/W application decides to block the packet
(4) Switch drops this packet
SDN Security App Development Challenges

• However, it is not easy to create security apps in SDN
  – Security service creation and composition challenge
    • How do we simplify development of security applications?
  – Information deficiency challenge
    • E.g., TCP session, network status
  – Threat response translation challenge
    • How do we enforce security policies to the network devices?
FRESCO

• FRESCO is a new frame work that
  – Provides a new development environment for security applications
  – Effectively manages shared resources among security applications
  – Simplifies deployment of security policies
    • provides a set of 7 new intelligent security action primitives
      – E.g., block, deny, allow, redirect, and quarantine
Architecture

• Component
  – Application layer
    • Development env. (DE)
    • Resource controller (RC)
  – Kernel layer
    • Security enforcement kernel
    • FortNOX
      – paper in HotSDN 2012
Development Environment

- **FRESCO Module**
  - Basic operation unit

- **FRESCO DB**
  - Simple database
    - (key,value) pairs

- **FRESCO script**
  - Define interfaces
  - Connect multiple modules
Development Environment

- FRESCO script
  - Format
    - Instance name (# of input) (# of output)
    - type: class of this module
    - input: input for this module
    - output: output of this module
    - parameter: define some variables
    - event: trigger a module
    - action: conduct this action

```
port_comparator (1)(1) {
  type: Comparator
  event: PUSH
  input: destination_port
  output: comparison_result
  parameter: 80
  action: -
}
```

Inspired by Click Modular Router
Operational Scenario

Administrator

Fresco Security Kernel Enforcement

Controller

Fresco Script

OpenFlow switch

OpenFlow switch

OpenFlow switch

OpenFlow switch

OpenFlow switch

Event

Monitoring

DE

RC

Table

Module 1

Module 2

Module 3

Module 4

Instance 1

Instance 2

Fresco DB

Execution
Implementation

• NOX (open source OpenFlow controller) based
  – Development environment
    • NOX based Python application
  – Resource controller
    • NOX based Python application
  – Security enforcement kernel
    • Modify NOX (C++)
Example: Scan Detection

- **Steps**
  - Check blacklist → Threshold based scan detection → Drop or Forward
Example: Reflector Net

- Confuse network scan attackers
- Steps
  - Threshold based scan detection → Reflect or Forward

```
scan_detect (1)(1) {
  type: ScanDetect
  event: TCP_CONNECTION_FAIL,
       TCP_CONNECTION_SUCCESS
  input: blacklist_out
  output: detect_result
  parameter: NONE
  action: NONE
}
```

```
final_action (1)(0) {
  type: ActionHandler
  event: PUSH
  input: detect_result
  output: NONE
  parameter: NONE
  action: detect_result == REDIRECT(10.0.0.3):FORWARD
}
```
Example: Reflector Net

- Test result

Scanner 10.0.0.2

1) Scan Target
2) Redirect to HoneyNet

Target 10.0.0.4
Port 445 is open

HoneyNet 10.0.0.3
Port 444 is open

(4) Scanner thinks Port 444 is open

(3) Return to Scanner
Examples

More Examples in the paper
- BotMiner
- P2P Plotter
And etc..
Evaluation

**Source code length comparison**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Implementation</th>
<th>Standard</th>
<th>OpenFlow</th>
<th>FRESCO</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRW-CB</td>
<td></td>
<td>1,060</td>
<td>741</td>
<td>66 (58 + 8)</td>
</tr>
<tr>
<td>Rate Limit</td>
<td></td>
<td>991</td>
<td>814</td>
<td>69 (61 + 8)</td>
</tr>
</tbody>
</table>


**Flow rule setup time**

<table>
<thead>
<tr>
<th></th>
<th>NOX</th>
<th>Simple Flow Tracker</th>
<th>Simple Scan Detector</th>
<th>Threshold Scan Detector</th>
<th>BotMiner</th>
<th>P2P Plotter Detector</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (ms)</td>
<td>0.823</td>
<td>1.374</td>
<td>2.461</td>
<td>7.196</td>
<td>15.461</td>
<td>11.775</td>
</tr>
</tbody>
</table>

Please refer to our paper for the explanation of each test case
Summary and Future Work

• FRESCO
  – Create security applications easily
  – Deploy security applications easily
  – Focus on creating security applications

• Future work
  – Port FRESCO to other controllers for open source release
    • E.g., POX or Floodlight
  – Create more modules (now 16 basic modules)
More Information

www.openflowsec.org

• Demo movies
  – Security Constraints Enforcement
  – Reflector Nets
  – Automated Quarantine
Thank you, Question ?
Optional
OpenFlow Architecture

OpenFlow Switch specification

OpenFlow Switch

Secure Channel

Flow Table

PC

Controller

Application

OpenFlow Protocol

SSL

From OpenFlow tutorial
Resource Controller

• Monitor OpenFlow switches
  – Check current status of security policies
    • How many security policies are active
    • How many packets are matched to policies
    • And etc..
• Remove some old policies
  – Save some space for FRESKO security constraints
  – Garbage collection