Security as an App and Security as a Service: New Killer Applications for Software Defined Networking?

Guofei Gu
SUCCESS Lab, Texas A&M
Credits

• Seungwon Shin (TAMU)
• Phil Porras, Vinod Yegneswaran... (SRI International)
Roadmap

• Security in the paradigm of SDN/OpenFlow

• Security as an App (SaaA)
  – New app development framework: FRESCO
  – New security enforcement kernel: FortNOX

• Security as a Service (SaaS)
  – New security monitoring service for cloud tenants: CloudWatcher

• Summary
Problems of Legacy Network Devices

• Too complicated
  – Control plane is implemented with complicated S/W and ASIC

• Closed platform
  – Vendor specific

• Hard to modify (nearly impossible)
  – Hard to add new functionalities
Software Defined Networking (SDN)

- Three layer
  - Application layer
    - Application part of control layer
    - Implement logic for flow control
  - Control layer
    - Kernel part of control layer
    - Run applications to control network flows
  - Infrastructure layer
    - Data plane
    - Network switch or router

SDN architecture from ONF
OpenFlow Architecture

A controller application can enforce any flow rules to network switches.
Killer Applications of SDN?

• Reducing Energy in Data Center Networks (load balancing)
• WAN VM Migration
• ...
• How about security?
  – We are going to talk about this, more specifically:
  – Security as an App (SaaA)
  – Security as a Service (SaaS)
Software App Store Today

Available on the iPhone
App Store

Google Play

Windows App Store
Windows 8

[Images of app icons]
Security as an App

• SDN naturally has an application layer
• Security functions can be apps on top of SDN/networking OS
  – Firewall
  – Scan detection
  – DDoS detection
  – Intrusion detection/prevention
  – …
• Why SaaS?
  – Cost efficiency
  – Easy deployment/maintenance
  – Rich, flexible network control
Security as a Service

• Clouds are large, complicated, and dynamic
• How do tenants deploy security devices/functions?
  • Tenant can use some pre-installed fixed-location security devices
    – Not able to keep up with the high dynamisms in network configurations
  • Tenant can Install security devices for themselves
    – Difficult

• Need a new Security Monitoring as a Service mechanism for a cloud network
Challenges and Our Contributions

• *It is not easy to develop security apps*
  – FRESCO: a new app development framework for modular, composable security services

• *It is not secure when running buggy/vulnerable/multiple security apps (e.g., policy conflict/bypass)*
  – FortNOX: a new security enforcement kernel

• *It is not convenient to install/use security devices for cloud tenants*
  – CloudWatcher: a new security monitoring service model based on SDN
FRESCO: Framework for Enabling Security Controls in OpenFlow networks
What is FRESCO?

• A new framework
  – Enables to compose diverse network security functions easily (with combining multiple modules)
  – Enables to create own network security functions easily (without requiring additional H/Ws)
  – Enables to deploy network security functions easily and dynamically (without modifying the underlying network architecture)
  – Enable to add more intelligence to current network security functions
Fresco – Overall Operation

Create Modules → Load Modules → Run Modules → Monitor OpenFlow switches

Notify NOX of loading Fresco modules

Answer from NOX
FRESCO Modular Design

Module

F-DB instance

Event

Parameter (1, ..., n)
Action (1, ..., n)

Input 1
.:.
Input n

Output 1
.:.
Output m

Incoming flow

(parameter1 = 80)
If input1 == parameter1:
Output = 1;
Else:
Output = 0;

Push

Action1 = Drop,
Action2 = Forward
If input1 == 1:
Do Action1
Else:
Do Action 2
FRESCO – Script Language

• Goal
  – Define interfaces, actions, and parameters
  – Connect multiple modules
  – Similar to C/C++ function, start with { and end with }

• Format
  – Instance name (# of input) (# of output)
    • denotes the module name and the number of input and output variables
  – INPUT: a₁,a₂,
    • denotes input items for a module aₙ may be set of flows, packets or integer values
  – OUTPUT: b₁,b₂,
    • denotes output items for a module bₙ may be set of flows, packets or integer values
  – PARAMETER: c₁,c₂,
    • denotes configuration values of a module cₙ may be real numbers or strings
  – EVENT: d₁,d₂,
    • denotes events that will be delivered to a module dₙ may be any predefined string
  – ACTION : condition ; action,
    • denotes actions that will be performed based on condition
Simple Working Example: Reflector Net

```plaintext
find_scan (1) (2) {
  TYPE: ScanDetector
  EVENT: TCP_CONNECTION_FAIL
  INPUT: SRC_IP
  OUTPUT: SRC_IP, scan_result
  PARAMETER: 5
  ACTION: -
  /* no actions are defined */
}

do_redirect (2) (0) {
  TYPE: ActionHandler
  EVENT: PUSH
  INPUT: SRC_IP, scan_result
  OUTPUT: -
  PARAMETER: -
  ACTION: scan_result == 1? REDIRECT: FORWARD
  /* if scan_result equals 1, redirect; otherwise, forward */
}
```

*Module 1*  
*Module 2*
Reflector Net

Scanner 10.0.0.2

(1) Scan Trial

Target 10.0.0.4
Port 445 is open

(2) Redirect to Honeynet

HoneyNet 10.0.0.3
Port 444 is open

(3) Return to Scanner

(4) Scanner thinks Port 444 is open
Cooperating with Legacy Security Applications

![Diagram showing cooperation between security applications and host systems to detect and quarantine infected hosts.]

- **FRESCO Application**
- **FRESCO DE**
- **NOX**
- **FRESCO SEK**
- **BotHunter**

**Steps:**
1. **Attacker** connects to **Host A**
2. **BotHunter** alerts on **Host C**
3. **Alert level and detected host** is communicated
4. **Quarantine Host C**
5. **Connection** to **Host A** is blocked
6. **Host C** is infected!
BotMiner - Overview

• How to detect botnet C&C channels
  – Find C-plane
    • Who is talking to whom?
      – Flow: SRC IP, DST IP, DST Port, Protocol
      – Features
        » BPS (bytes per second), FPH (flows per hour)
        » BPP (bytes per packet), PPF (packets per flow)
      – Clustering based on features
  – Find A-plane
    • Who is doing what?
      – Clients perform malicious activities
        » E.g., scanning, spam activity and etc
      – Clustering based on malicious actions
        » E.g., scan cluster
  – Co-Clustering
    • Combine results of two clusters to find botnet C&C channels
    • Channels showing similar C-plane patterns and performing malicious actions
BotMiner in FRESCO (Diagram)

TCP connection fail or TCP connection success

- Lookup table(input1)
  - If table contains input1:
    - output1 = 1
    - output2 = input1
  - Else:
    - output1 = 0
    - output2 = input1

TCP connection fail or TCP connection success

- Lookup flow table(SRC IP, DST IP)
- Gather flow information(SRC IP, DST IP)
- Find BPS and PPS(SRC IP, DST IP)
- Cluster(SRC IP, DST IP)
- Output = [cluster results]

A-Plane Clustering

- If input1 == 1:
  - output1 = input2
- Else:
  - Do scan_detection()
  - If scan_detection() detects scan:
    - output1 = input2

C-Plane Clustering

Co-Clustering

- Co-Cluster(input1, input2)
- If Co-Cluster() finds common IPs:
  - output1 = 1
  - output2 = [common IPs]

Action

- Action1 = Drop
- If input1 == 1:
  - Do Action1 on output2
BotMiner in FRESCO (Script)

**A-Plane Clustering**

BM1 (1) (2) {
  **EVENT**: TCP_CONNECTION_FAIL, TCP_CONNECTION_SUCCESS
  **INPUT**: Source IP
  **OUTPUT**: Result, Input1
  **PARAMETER**: -
  **ACTION**: -
}

BM2 (2) (1) {
  **EVENT**: PUSH
  **INPUT**: BM1-0, BM1-1
  **OUTPUT**: Result
  **PARAMETER**: 10
  **ACTION**: -
}

**B-Plane Clustering**

BM4 (2) (2) {
  **EVENT**: PUSH
  **INPUT**: BM2-0, BM3-0
  **OUTPUT**: Result1, Result2
  **PARAMETER**: -
  **ACTION**: -
}

BM5 (2) (0) {
  **EVENT**: PUSH
  **INPUT**: BM4-0, BM4-1
  **OUTPUT**: -
  **PARAMETER**: -
  **ACTION**: BM4-0 == 1 ?Drop
}

**C-Plane Clustering**

BM3 (0) (1) {
  **EVENT**: TCP_CONNECTION_FAIL, TCP_CONNECTION_SUCCESS
  **INPUT**: -
  **OUTPUT**: Result
  **PARAMETER**: -
  **ACTION**: -
}
More ...

- Tarpits
- White Holes
- Scan detector
- P2P detector (P2P Plotter)
- Botnet detector (BotMiner)
- ...
- Over 90% reduction in lines of code compared with their standard implementations
- Already include more than 16 commonly reusable modules (expending over time)

“FRESCO: Modular Composable Security Services for Software-Defined Networks.” NDSS’13
FortNOX:
A Security Enforcement Kernel for OpenFlow
New Threat

• SDN apps can compete, contradict, override one another, incorporate vulnerabilities

• Worst case: an adversary can use a vulnerable and deterministic SDN app to control the state of all SDN switches in the network
SDN/OpenFlow Evasion Scenario

Dynamic Flow Tunneling
Prerequisites for a Secure OpenFlow Platform

• Must be resilient to
  – Vulnerabilities in OF applications
  – Malicious code in 3rd party OF apps
  – Complex interaction that arise between OF app interactions
  – State inconsistencies due to switch garbage collection or policy coordination across distributed switches
  – Sophisticated OF applications that employ packet modification actions
  – Adversaries who might directly target our security services to harm the network
Our Contributions

• Development of a security enforcement kernel for the NOX OpenFlow controller
• Role-based authorization
• Rule conflict detection
• Security directive translation
Classic NOX Architecture

- PY OF Apps
- Native C OF Apps
- Python SWIG

Send_OpenFlow_Command()
FortNOX Architecture

- Security Apps
- PY OF Apps
- Native C OF Apps
- OF IPC Proxy
- Actuator
- Directive Translator
- Python SWIG
- IPC Interface
- Aggregate Flow Table
  - Operator Rules
  - SECURITY Rules
  - OF App Rules
- FT_Send_OpenFlow_Command
  - Role-based Source Auth
  - State Table Manager
  - Conflict Analyzer
  - Switch Callback Tracking
- Separate Process
- Switch Callback tracking
Summary of FortNOX

- FortNOX – A new security enforcement kernel for OF networks
  - Role-based Authorization
  - Rule-Authentication
  - Conflict Detection and Resolution
  - Security Directive Translation

- Ongoing Efforts and Future Work
  - Prototype implementations for newer controllers (Floodlight, POX)
  - Security enforcement in multicontroller environments
  - Improving error feedback to OF applications
  - Optimizing rule conflict detection

“A Security Enforcement Kernel for OpenFlow Networks”. HotSDN’12
Some Demonstrations

- [www.openflowsec.org](http://www.openflowsec.org)
- Some technical reports and publications
- DEMO videos
  - Demo 1: **Constraints Enforcement** [high res .mov or Github! ]
  - Demo 2: **Reflector Nets** [high res .mov or Github! ]
  - Demo 3: **Automated Quarantine** [high res .mov or Github! ]
- FRESCO/FortNOX beta to be released soon
CloudWatcher:
Network Security Monitoring Using OpenFlow in Dynamic Cloud Networks

or: How to Provide Security Monitoring as a Service in Clouds?
Goal

• Provide **Security Monitoring as a Service** for a cloud network

• How to Provide
  – Routing algorithms
    • The algorithms guarantee that specified (static) network security devices can monitor (dynamic) specific network flows
  – A script language
    • Register security devices easily
    • Create security policies easily
CloudWatcher

• A new framework
  – Provide security monitoring services for large and dynamic cloud networks
  – Detour network packets to be inspected by pre-installed network security devices automatically
    • OpenFlow
  – Provide a script to operate this framework
Operating Scenario

1. **Register Security Devices**
   - {ID, TYPE, LOCATION, MODE, Func}
   - {1, NIDS, 8, PASSIVE, Detect HTTP}

2. **Create Security Policies**
   - {FLOW CONDITION, DEVICE SET}
   - {10.0.0.* \(\rightarrow\) *:80, {1}}

3. **Parse Security Policies**

4. **Create Routing Rules**

5. **Translate Routing Rules into OpenFlow Rules**

6. **Enforce Flow Rules into Routers**

**Administrator**

**Router (Device ID = 8)**

**NIDS (ID = 1)**
How to Control Flows

• 4 approaches
  – Multipath naïve
  – Shortest through
  – Multipath shortest
  – Shortest inline

- Sample network -
S: start node, E: end node
R: router, C: security device
Selected Controlling Algorithm Example: Shortest Through

- Find the shortest path passing through R4
  - Shortest path between S and R4
  - Shortest path between R4 and E
  - Path: S → R1 → R2 → R4 → R4 → R6 → E

- It considers the security device without producing redundant paths
- However, it may take more time to deliver packets
Summary of CloudWatcher

- CloudWatcher provides a new framework to monitor cloud networks
  - With the help of the SDN technology
- A cloud administrator can select algorithms based on network status
- A cloud administrator can monitor his network by writing simple scripts

- Work in progress; a position paper in NPSec’12
Summary of This Talk

• SDN is a new technology, and security can be a new killer app
  – SDN is impactful to drive a variety of innovations in network security
• We investigate the possibilities of security as an app and security as a service
• We propose key technologies to enable SaaS and SaaS
  – FRESCO
  – FortNOX
  – CloudWatcher
• Let’s contribute together to SDN and Security!
Questions & Answers

Http://faculty.cse.tamu.edu/guofei