SDN AND SECURITY: A MATCH MADE IN HEAVEN - OR NOT?

Chuck Black
Senior Software Developer
Tallac Networks
SDN TODAY

"Who Stole My SDN"?
The Cisco Effect

APIC-EM/CLI  NETCONF/YANG  BGP-LS/PCEP  APIC-DC/OpFlex

Proactive OpenFlow  Reactive OpenFlow

Legacy (no disruption)
Evolutionary (minimal changes)
Network Management ++

Disruptive
Revolutionary
Exciting
The Overlay Effect (Nicira)

Intractable Data Center Networking Problems

- MAC table overflow
- @$%! Spanning Tree
- VLAN exhaustion
- Traffic Engineering

- Flood to all ports
- Convergence times
- Wasted bandwidth
- Multiple tenants
- QoS & Routing
SDN Landscape Today

**Overlay-based**
- VMware NSX
- Juniper Contrail
- Cisco ACI
- Turnkey

**API-based**
- OpenDaylight
- APIC-EM
- Tail-f (NSO)
- Application development

**OpenFlow-based**
- ONOS
- BigSwitch
- NEC VTN
- Turnkey or App development
SDN Devices Today

OpenFlow

Device API

- REST
- CLI
- NETCONF

- RIB
- MPLS LSPs
- BGP-LS
- QoS, ACL, PBR

TCAM
SDN Device Control Points

- **Config**: Configuration parameters for security and policy
- **RIB**: Routing Information Base entries, MPLS Label Switched Paths
- **FIB**: Forwarding Information Base entries such as OpenFlow and BGP-FS
SDN Applications Today: OpenFlow

OpenFlow

Applications

Events

Packets In

Packets Out

Flows

Topo

OpenFlow Controller

‘Reactive’ OpenFlow

“Network Revolution”

Actions
Flow Modifications
Packets Out

Device

OpenFlow

TCAM
OpenFlow-based SDN Interfaces

Device-level OpenFlow
- OVS-OFCTL

Controller-level OpenFlow
- Matches/Actions
- Set flows per-device

Policy-level OpenFlow
- Intents
- Declarative
SDN Applications Today: APIs

Events → Applications → Controller → Devices

NETCONF, BGP, BGP-LS, PCEP, RIB, Control Machine, TCAM

"Network Evolution"
SDN APIs Today

**Device-level APIs**
- NETCONF, REST, SNMP, CLI

**Controller-level APIs**
- Abstraction
- Multiple-device operations

**Policy-level APIs**
- Intents
- Declarative
SDN Applications Today: Overlays

"Network Circumvention"
MAC-in-IP ‘Overlay’ Tunnels: VXLAN, NVGRE, STT
SDN – Automation?

- 2003: ForCES
- 2005: Clean Slate
- 2008: OpenFlow
- 2010: Production OpenFlow Switches
- 2012: SDN Explosion

OpenFlow-based Nirvana

APIs

Overlays

Automation
Is Automation a Type of SDN?

- **What is Automation?** Using Ansible, Python, StackStorm, SaltStack, on-device containers, scripts, etc. to automate tedious manual processes, and to dynamically respond to network and policy changes.

- **Compare Automation to traditional SDN characteristics:**
  - **Plane Separation:** Of forwarding and control planes?
  - **Programmability:** Automation of tasks?
  - **Centralized Control:** Network-wide views and policies?
  - **Simplified Devices:** Reduced device complexity?
  - **Openness for Innovation:** Ability to create new networking solutions to old or persistent problems?
  - **Virtualization:** Of network functions and resources?
Summary: SDN Today

OpenFlow-based SDN
- SDN Application
  - Controller
  - OpenFlow
- Forwarding

API-based SDN
- SDN Application
  - Controller
  - NETCONF, CLI, etc
- Control

Overlay-based SDN
- SDN Application
  - Controller
  - Virtual Networks
  - Physical Device
- Forwarding

Automation-based SDN
- SDN Application
  - Agent, Python, etc.
  - Control
  - Forwarding

More risk: Less Risk
SECURITY ISSUES AND SDN
Vulnerability and SDN Type

Risk depends on SDN application type

- **OpenFlow-based**
  - **Reactive**: susceptible to denial of service attacks, overloaded links, overloaded CPU/disk, as well as centralized attacks on controllers
  - **Proactive**: less risk, only centralized attacks on controllers

- **API-based**: less risk, really just network management ++

- **Overlay-based**: less risk, contained/secure datacenter environment
Vulnerability and SDN Application Type

Risk depends on SDN application type:

- **Internal** (runs inside JVM container, uses Java APIs): Running inside controller mean greater performance and capabilities, but failures more likely to jeopardize operation of entire controller and other applications.

- **External** (runs elsewhere, uses REST API): Running outside controller, potentially on different system or location, protects controller from application failures. Still, invalid or incorrect requests can impact the controller and indirectly, other applications.
Vulnerability and Attack Surface

Attack surface a major criteria for understanding vulnerability

- **Distributed Model**
  - Huge attack surface, must secure entire network of devices because critical policy is spread throughout, one change can effect the entire network

- **Centralized Model**
  - Limited attack surface, just centralized controller(s). Easier to defend, protect, isolate, secure.
Reliability of Centralized Systems

Seriously? How can we expect centralized systems to be reliable?

- Telephony centralized their networks three decades ago
- Google, Amazon, Facebook, Twitter (okay maybe one of those should fail)
- Cloud (centralized servers, cloud management, massive quantities of data)

Centralized intelligence is not a new idea – just new to networking. Some other 'dangerous' new ideas:

- Oog bagoog! Fire! We're all going to burn!
- Egads old chap! Electricity! We're all going to fry!
- OMG! Flying! In the sky? We're all going to die!
Providing redundancy for reliability also provides scale-out

- Shared device load across controller in cluster
- Shared compute load across controllers in cluster
- Shared network load across links connecting to controller
SECURITY ENHANCEMENT THROUGH SDN

SDN can actually make the network *more* secure
Simplified Devices

- Analysis done centrally, device only responsible for forwarding to centralized intelligence
- Frees up device to be less expensive, can concentrate on speeds and feeds, commoditized, not fancy extra proprietary functionality.
- Security application from anybody (no vendor lock-in)
Centralized Analysis and Intelligence

- Gather intelligence and information from all locations and sites
- Processing power to do deep analysis, machine learning, network-wide views, etc.
- Ability to mitigate threats, dynamically closing down malicious users and/or systems, correlation with other data about users, systems, patterns, history, etc.
Detect anomaly, requires further snooping

Dynamically enable port mirroring for highly granular type of traffic, source or destination of traffic, etc.

Analyze traffic at central location benefitting from network-wide views, policies, etc.
SDN Security Example - AMQ

Automatic
Malware
Quarantine

Source: ONF
Dynamic Mitigation

**OpenFlow:**
- Redirect suspicious traffic through IDS/IPS, firewall, etc.
- Place user or traffic into quarantine zone
- Drop packets from malicious user
- Only allow certain traffic types

**NETCONF or BGP-LS/PCEP:**
- Static routes for certain users or traffic types to keep them off main network
- RIB entries to redirect traffic on different path or to IDS/IPS, firewall, etc.
- Deny traffic for malicious users
Cisco – APIC-EM: Using clunky old CLI
Cisco – APIC-EM

APIC Enterprise Module

Security Services
- Quarantine
- Investigation, Mitigation, Remediation

Core Services
- Remediate
- Contain

Other Data
- pxGrid
- ASA
- APIC Enterprise Module
- pxGrid
- Catalyst 3850
- Intranet
- Other Data
- SIEM
- NB-API
- ISE
- pxGrid
- Security Services
Cisco – APIC-EM

APIC Enterprise Module

Core Services

Security Services

Investigation, Mitigation, Remediation

Quarantine

Other Data

Intranet

Catalyst 3850

NB-API

pxGrid

ISE

SIEM

pxGrid

#RSAC
Assuming your organization has chosen to begin its SDN Adventure, the following actions will help you build and maintain a secure SDN network:

- Make sure you have a redundant architecture (controller and links)
- Make sure you have secured communication between network devices and controller
- Make sure you have secured the controller from threats (authentication/authorization of requests, protection from DoS threats, protection from malicious or buggy applications)
Thanks!