Security Challenges in Software Defined Networks (SDN)

Lecture 18
Outline

- Market and SDN
- Conventional Networks v.s SDN
- OpenFlow-enabled SDN devices
- SDN Security Applications
- SDN Security Challenges
- Community Debate regarding Security in SDN
Market and SDN

- In 2016, the market research firm IDC predicted that the market for SDN network applications would reach **US$3.5 billion** by **2020**.

- Leading IT companies such as Nokia, Cisco, Dell, HP, Juniper, IBM, and VMware have developed their own SDN strategies.

  Marc C. Dacier, Hartmut Cwalinski, Frank Kargl, Sven Dietrich, Security Challenges and Opportunities of Software-Defined Networking, Apr 3, 2017

- In 2015, AT&T reduced provisioning cycle by 95% with SDN.

  “We have taken a process from **low automation and weeks** to complete to **high automation and minutes** to complete. We’re turning the industry on its head in an unprecedented way.” John Donovan

  AT&T’s analyst conference in August 2015, John Donovan
Conventional Networks vs. SDN

**Control Plane**
- Smart

**Data Plane**
- Dumb, fast

Decoupling

**Network Applications**
- Traffic mngmnt, QoS
- Policy Imp.
- Security services

**OpenFlow**
- South-bound API

**Decentralized Control**
- Limited visibility
- Vendor-specific
- Missconfiguration
- Poor responses
- Policy conflicts
- Security breaches
- Decentralized
- Complex
- Static architecture
- Innovation is difficult
- Costly
- Yes costly

**Software Defined Networks**

## OpenFlow-enabled SDN devices

**OpenFlow is: Enabler of SDN**
- Protocol between the control plan and data plane
- Describes how controller and a network forwarding device should communicate

### Match Fields

<table>
<thead>
<tr>
<th>Switch port</th>
<th>MAC src</th>
<th>MAC dst</th>
<th>Eth type</th>
<th>VLAN ID</th>
<th>IP src</th>
<th>IP dst</th>
<th>TCP psrc</th>
<th>TCP pdst</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switching</td>
<td>*</td>
<td>00:2E</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>port3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>300</td>
</tr>
<tr>
<td>Routing</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>4.5.6.7</td>
<td>*</td>
<td>*</td>
<td>port5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>250</td>
</tr>
<tr>
<td>Firewall</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>drop</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>500</td>
</tr>
</tbody>
</table>
SDN security applications

examples

• **Load Balancer**: send each HTTP request over lightly loaded path to lightly loaded server.

• **Firewall**: inform Central Controller about malware’s packets, controller pushes new rules to drop packets.

Routing, Load Balancer, Access Control, monitoring, firewall, DDoS Mitigation, IDS/IPS

Application plane

Abstract Network View

Network Virtualization

Up-to-date Global Network View

Control Plane

Incoming packets -> Server

A → B drop
The big Picture
SDN Architecture

SDN Security Challenges
Application Plane
Security Challenges

- Lack of Authentication and Authorization
- Lack of Access Control and Accountability
- Fraudulent flow rule insertion

SDN aware & SDN unaware apps
Nested applications

App classes:
- Service apps
- Sensitive apps
  - Path characteristics
  - Access ports
  - Monitor traffic
  - Reject/Accept flows

Lack of Authentication and Authorization
- Fraudulent flow rule insertion

Path characteristics
- Access ports
- Monitor traffic
- Reject/Accept flows
Application Plane
Targeted Threat/Proposed Solution

Threats within/from apps

Framework for security apps development
(FRESCO Scripting language)

Security policy violation

Security policy verification framework
- **Flover**: on controller
  new/old rules conflict
- **ndb**: root cause
- **OFRewind**: trace anomalies

flow rules contradiction

**Assertion**-based language
- catch bugs before deployed
- forwarding loops
- black holes

Access control breach

Permission system (PermOF):
least privilege on apps
The design is based on a Set of permissions & Isolation mechanisms

- Ensures controller superiority over applications
- Isolates control flow and data flow
- Controller should be able to mediate all the apps’ activity

<table>
<thead>
<tr>
<th>Category</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read</td>
<td>read_topo, read_all_flow, read_stats, read_pkt_in_payload</td>
</tr>
<tr>
<td>Notification</td>
<td>pkt_in_event, flow_removed_event, error_event, topology_event</td>
</tr>
<tr>
<td>Write</td>
<td>flow_mod_route, flow_mod_drop, flow_mod_modify_hdr, modify_all_flows, set_device_config, set_flow_priority</td>
</tr>
<tr>
<td>System</td>
<td>network_access, file_system_access, process_runtime_access</td>
</tr>
</tbody>
</table>

Figure 1: PermOF Isolation Framework

Control Plane Security Challenges

- Threats due to Scalability
  - Huge # flow rules
  - Saturation

- DoS Attacks
  - SDN response times
  - IP packets with random headers

- Challenges in Distributed Control Plane
Controller scalability

1. Wildcards mechanism
   - Load balancing: direct an aggregate of client requests to replicas
2. Increase the processing power (McNettle controller) parallelism
3. Hybrid reactively/Proactive controller

DDoS Attack

Detection Framework
SDN DDoSDetection

Challenges in distributed control plane
intra-domain & inter-domain (DISO)

Throughput Scaling

- NOX-MT scales to 5m f/s at 10 CPU cores
- Beacon \(\rightarrow\) 13m f/s at 20 CPU cores
- McNettle \(\rightarrow\) 20m f/s at 46 CPU cores

Reactively vs. Proactive Controller

Marcial P. Fernandez, Evaluating OpenFlow Controller Paradigms, 2013
1. **Flow collector module**: gathers flow entries within intervals.
3. **Classifier**: Analyzes → Alarm?

intra-domain & inter-domain (DISO)

- **intra-domain**: manages its own network domain
  - compute the paths of flows
  - dynamically react to network issues (broken line, high latency, bandwidth cap exceeded)
  - redirecting and/or stopping traffic

- **inter-domain**:
  - discovers neighboring controllers and manages communication among controllers
  - exchange aggregated network-wide information with others
Data Plane Security Challenges

Flow rules installation
- Genuine vs. malicious rules
- Limited table entries
- Limited switch buffer

Switch-Controller link
- #switches per controller
- path Length
Data Plane
Targeted Threat/Proposed Solution

flow rule contradiction

man-in-the-middle attacks

Real-time contradiction check
FortNox
High level points
-- Debate
Centralization in SDN

The Good:
- Fast responsiveness
- Easy to removing policy inconsistencies
  - centralized routing algorithms
  - Firewalls
  - network-monitoring

The Bad:
- Single point of failure may be exploited by an internal or external attacker

Regarding DDoS
Bad: centralization added a new type of denial-of-service (DoS) vector.
Good: Effective management of existing DoS attack types
  - Using Global view
  - Traffic analysis

New security challenges but benefits appear to be predominant!!!
Attack Surface vs. Defense Opportunities

Good:

• In SDN defenders can create customized security solutions
• e.g. Anomaly detection systems
  – Global view
  – Open hardware interfaces
  – Centralized control

Bad:

• Benefit the attackers (zero day attacks)
  – The centralized architecture
  – Lack of defender expertise
  – Still immature technology
Centralized vs. Distributed Approach

Good:

• Reduced complexity by splitting into planes.
  – Easier testable
  – E.g, routing algorithms simpler than the distributed approach in conventional networks.

Bad:

• Stressed by two aspects that strongly call for the use of a distributed approach.
  – The need for **scalability**
  – **Operational requirements** (fault tolerance)
Implementing the control plane completely in software

Good:
• Programmability

Bad:
• Opposes simplicity: raises issues about algorithmic complexity.
  – Why: additional requirements that weren’t imposed on classical networks but are now thinkable in SDN.
  – Simplicity is a key design principle in building secure systems.

SDN has the potential to be simple—but making it simple is quite complex.
Open problems & research directions

- How to implement authentication and authorization to certify SDN applications.
- How to implement access control and accountability in SDN.
- How to implement customized security procedures based on the type or categories of applications.
- How can we find automated derivation of Secure SDN Configurations.
- How can we secure the controller-switches communication?
- How can we perform efficient intrusion detection and anomaly detection in SDNs?
- How can we operate SDN in presence of untrusted HW components?
- How can we protect the controller itself?

Without security, SDN will not succeed!
Thank you