Towards Secure and Dependable Software-Defined Networks

Diego Kreutz, Fernando Ramos, Paulo Veríssimo
kreutz@lasige.di.fc.ul.pt, fvramos@fc.ul.pt, pjv@di.fc.ul.pt

LASIGE - Large-Scale Informatics Systems Laboratory
FCUL - Faculty of Sciences of University of Lisbon
Outline

On the way to SDN
Threat Vectors
Security & Dependability
  Architecture
  Mechanisms
  by Design
Final Remarks
On the way to SDN
(currently: tied data/control planes)

Complicated and hard to integrated management solutions.

Integrated (hardware & software) and complex control planes. Proprietary, expensive and hard to evolve solutions. Innovation is very slow and costly.
On the way to SDN
(SDN: decoupled data and control planes)

Simple and easy to develop apps. A load balancing mechanism in only 500 lines of code?

Integrated (hardware & software) and complex control planes. Proprietary, expensive and hard to evolve solutions. Innovation is very slow and costly.
Software-Defined Networks

- Decoupled control and data planes
- Logically centralized controllers
- Programmability
- Openness
- Interoperability
SDN: Logically centralized controllers (a.k.a. NOS)

Management apps (**net intelligence**): routing protocols, energy-aware routing, access control, security policy control, etc.

NOS (e.g. NOX, POX, Onix, Floodlight, Trema, Ryu NOS, Beacon, Maestro)

Flow tables updated by management apps, through NOS.

Network OS (programming abstractions, data distribution, low level controls, etc.)

Applications (control logic)
Outline

On the way to SDN

Threat Vectors

Security & Dependability

Architecture

Mechanisms

by Design

Final Remarks
Threat vectors

Management connection
(e.g., SSH)

SDN control protocol
(e.g., OpenFlow)

Data plane physical/logical connections

SDN Controller

Admin Station

Control & Management

Data Plane
Threat vectors

1. Threat vector 1: forged or faked traffic flows

Management connection (e.g., SSH)
SDN control protocol (e.g., OpenFlow)

Data plane physical / logical connections

SDN device

Data Plane

SDN controller

SDN control protocol (e.g., OpenFlow)
Threat vectors

1. Data plane physical / logical connections
2. SDN device
3. SDN device
4. SDN device
5. SDN Controller
6. Admin Station

Management connection (e.g., SSH)
SDN control protocol (e.g., OpenFlow)

Threat vector 2: attacks on vulnerabilities in switches
Threat vectors

Threat vector 3: attacks on control plane communications

1. SDN device
2. SDN device
3. SDN device
4. SDN device
5. SDN device
6. SDN device
7. SDN device

Data plane physical/logical connections

Management connection (e.g., SSH)
SDN control protocol (e.g., OpenFlow)

SDN Controller

Admin Station

Control & Management

Data Plane
Threat vectors

Threat vector 4: attacks on and vulnerabilities in controllers
Threat vectors

Threat vector 5: lack of mechanisms to ensure trust between the controller and management applications.
Threat vectors

Threat vector 6: attacks on and vulnerabilities in administrative stations

Management connection (e.g., SSH)
SDN control protocol (e.g., OpenFlow)
Threat vector 7: lack of trusted resources for forensics and remediation
### Threat vectors

<table>
<thead>
<tr>
<th>Threat</th>
<th>Specific to SDN?</th>
<th>Consequences in SDN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector 1</td>
<td>no</td>
<td>can be a door for DoS attacks</td>
</tr>
<tr>
<td>Vector 2</td>
<td>no</td>
<td>but now the impact is potentially augmented</td>
</tr>
<tr>
<td>Vector 3</td>
<td>yes</td>
<td>communication with logically centralized controllers can be explored</td>
</tr>
<tr>
<td>Vector 4</td>
<td>yes</td>
<td>controlling the controller may compromise the entire network</td>
</tr>
<tr>
<td>Vector 5</td>
<td>yes</td>
<td>malicious applications can now be easily developed and deployed on controllers</td>
</tr>
<tr>
<td>Vector 6</td>
<td>no</td>
<td>but now the impact is potentially augmented</td>
</tr>
<tr>
<td>Vector 7</td>
<td>no</td>
<td>it is still critical to assure fast recovery and diagnosis when faults happen</td>
</tr>
</tbody>
</table>
Outline

On the way to SDN

Threat Vectors

Security & Dependability

Architecture

Mechanisms

by Design

Final Remarks
Security & Dependability

System/Application Distribution/Replication Services

Common Northbound API
Security & Dependability

- Replication
- Diversity
- Self-healing mechanisms
- Dynamic device association
- Trust between controllers and devices
Security & Dependability

- Trust between applications and controller software
- Security domains
- Secure components
- Fast and reliable software update and patching
Security & Dependability

App A will fail as well

App B will keep working since it is replicates

Switches should be able to dynamically re-associate with new or working controllers
Outline

On the way to SDN
Threat Vectors
Security & Dependability
Architecture
Mechanisms by Design
Final Remarks
Security & Dependability

A simple message: Security & Dependability by Design! And not bolted on!
Towards Secure and Dependable Software-Defined Networks

Diego Kreutz, Fernando Ramos, Paulo Veríssimo
kreutz@lasige.di.fc.ul.pt, fvramos@fc.ul.pt, pjv@di.fc.ul.pt

LASIGE - Large-Scale Informatics Systems Laboratory
FCUL - Faculty of Sciences of University of Lisbon
SDNs: improvements
(scalability: from 30K to 1M flows/s)

**Challenge: 100 edge switches = 10M flows/s**

- Original NOX: 30K flows/s
- Maestro: 300K & 600K flows/s
  - Scales linearly with the number of machines
    - \(10M / 300K = 34\) Maestro instances
- NOX-MT: 1M flows/s
  - 1.8M flows/s for 32/64 edge switches (10^5 MACs/switch)
  - 1.0M flows/s for 256 edge switches
    - \(10M / 1M = 10\) NOX-MT instances
- Kandoo: ? flows/s
  - Two-layers of controllers (local and root)
  - Can easily scale to N (?) on-demand edge switches

Still not convinced? Give a look at (suggested reading) “The Scaling Implications of SDN.”
SDNs: improvements
(scalability: fighting with the counters)

- DevoFlow
  - Reduce control traffic (updates of counters)

- Software-Defined Counters (SDCs)
  - Solve the counters’ ASIC design problem

- DIFANE
  - Reduce control overhead (counters)

- Software-Defined Cell Networks (SDCNs?)
  - Reduce control overhead (counters)
  - Use specific TCAMs/tables for DPI

- Kandoo
  - Scalable controllers (reduce the monitoring overhead)