Firewall offloading based on SDN and NFV

ITG 5.2.2/5.2.4 05.12.2016

Raphael Durner
r.durner@tum.de
Overview

- Introduction
  - Motivation
  - Main security requirements
- Network Function Virtualization
  - State of the Art
  - SDN & NFV Architecture
- Offloading Approach
  - ByteLim Logic
- Conclusion and open Questions
Motivation

**Software Defined Networking:**
Central control improves programmability and makes innovations easier

**Network Function Virtualization:**
Run Network Functions on Commodity Hardware and in the cloud

- Reduce costs
- Use available resources flexible

→ How can we guarantee a certain security level in these environments?
→ How can security related network functions be virtualized?
Main Security Requirements

- Isolation of services
  - Authentication and Authorization of devices/users/services
  - Isolation of flows
    → Stateless firewalling
- Stateful Firewalling
  - Check states of protocols
    - e.g. TCP, SIP
  - Normalization
    - e.g. filter non-standard DNS replies, filter html

Stateless < Stateful < Application Layer
State of the Art Firewalls

- Firewall Resides on the Networks‘ Edge
- Control Plane (State) and Forwarding Plane not decoupled
SDN and NFV Network Security Architecture

No distinct edge of the network

→ Firewall has to filter „everywhere“
→ Higher Load on Firewalls
→ Potentially higher security
Combine NFV and SDN

- Traffic steering with SDN
- Some parts of the Network function with SDN
- Some parts offloaded to NE
  - More Complex
  + leverages benefits of both approaches

Example TCP:
1. Connection Setup using VNF
2. Established connection using Hardware
Building Blocks

- VNF signals connection state to Logic
- Logic decides for Offloading Flow
- Command to SDN Controller
- SDN Controller installs necessary rules in Hardware
Challenges

- Flow setup in switch
  Can cause duplicates, packet loss and (short time) connection interruption

- Lack of Hardware
  Current OpenFlow switches handle header rewrite in software – very low throughput

- Offloading Decision:
  Flow classification algorithms needed

Which Flows can be offloaded and what are the gains?
Challenges

Which Flows can be offloaded and what are the gains?

→ What limits the usage of Offloading?

- Flow capacity in the hardware
- Flow Setup Rate
- Delay from decision to active Offloading
Constraint: Delay in the complete system

TCP handshake finished

Total Delay

TCP handshake finished

Flow handled in Hardware

Constraint: Delay in the complete system
Building Blocks Realisation

VNF → Logic → SDN Controller → Data Plane

Realisation

IpTables → Logic (Python) → Ryu SDN Controller → OVS/PICA/NEC

Conntrack → REST → OpenFlow
Total Delays

Results:
- ~ 100 ms for OVS & NEC

Implications:
- Not feasible for short lived flows like DNS
- Delay > RTT
  → Effects on TCP algorithm
Effects of Logic on Performance

- Which Flow metrics must be estimated by the Oracle?
- Gain i.e. cost of flow through VNF proportional to packet count
  ➔ Offload Flows with many Packets
Bytelimit Logic

- Logic decides based on Flowsize
  - Flows above a threshold are offloaded
- Oracle predicts Flowsize based on used application

**Mathematical Description:**

\[ f(x): \text{PDF of flow size} \]
\[ x: \text{Flow size} \]
\[ P(x): \text{PDF of Packets} \]

\[
P(x) = f(x) \times x = \int f(x) \cdot x \, dx
\]

\[
\text{discrete} \quad P = F \cdot X
\]
Example: Negative Exponentially distributed Flowsize

Share of Flows $f(x)$

$P(x) = f(x) \cdot x = \int f(x) \cdot x \, dx$

Discrete

$\rightarrow P = F \cdot X$
Bytelimit Logic

Simulation vs Theory

- Trendline similar
- Very big flows are hard to simulate
Bytelimit Logic

Simulation vs Theory

- Trendline similar
- Offset between simulation and theory
- Very big flows are hard to simulate
Conclusion and Open Questions

- Basic building blocks for offloading developed
- Bytlim logic shows promising results
  - Difference between simulation and theory should be evaluated

Open:

- How to predict Bytesize?
  - Simple classification by {source IP, Port}
Questions?