A Software-Defined Firewall Bypass for Congestion Offloading

F. Heimgärtner, M. Schmidt, D. Morgenstern, M. Menth

http://kn.inf.uni-tuebingen.de
Motivation

- Increasing network bandwidth
  - 1 Gb/s access ports, ≥ 10 Gb/s backbones

- Packet filtering at network perimeter
  - Stateful firewall
    - Advanced filtering capabilities
    - Can handle large numbers of flows
    - Limited throughput
  - SDN switch
    - Limited filtering capabilities
    - Limited number of flows
    - High throughput

- Idea
  - Selective bypassing of stateful firewall using SDN switch
Firewall Bypass

- Firewall bypass
  - Relieve congestion pressure
  - Divert trusted flows around firewall using switch

- Static Bypass
  - Whitelist or blacklisting using switch ACLs

- Dynamic Bypass
  - Detect accepted flows by sampling outgoing packets at firewall
  - Use SDN to install rules at switch
Dealing with Limited Flow Tables

► Problem with dynamic bypass approach
  ▪ Per-flow rules required for bypassing
    – E.g. 2 rules per TCP connection
  ▪ Limited size of hardware flow tables in commodity SDN switches
    – Number of usable rules: $2000 < n_r < 20000$
  ▪ Detecting connection teardown not possible in OpenFlow $< v1.5$
    – Rules cannot be reused immediately after connection is terminated

► Strategy
  ▪ Adapt bypass usage to firewall load
  ▪ Preferentially bypass large flows

► Challenges
  ▪ Measure traffic rate and detect overload
  ▪ Determine offloading rate and select appropriate flows
Detecting Congestion

Packet flow sampling using sFlow
- Random sampling of every $n^{th}$ outgoing packet

Trade-off for $n$
- Needs to be small enough to allow for exact measurement
- Must be large enough so that agent can comply with rate

Export raw packet headers to sFlow collector

Estimate packet rate and byte rate

High-load threshold: 80% of bottleneck capacity
Offloading Rate and Flow Selection

► Determine offloading rate
   - Objective: prevent rule exhaustion
   - Do not install rules faster than they can be reclaimed
   - Make sure that remaining rules suffice for a minimum time at current rate

► Flow selection
   - Objective: prefer large flows
   - Random offloading (ROff)
     – Select flows for offloading randomly based on sampled packets
     – Larger flows $\rightarrow$ larger sampling probability
   - Intelligent offloading (IOff)
     – Stronger preference of large flows compared to ROff
     – Count packets received for a flow
     – Increase offloading probability for flows with higher packet count
Proof-of-Concept Implementation

- Proof-of-Concept Implementation based on Ryu
  - Python SDN controller framework
  - OpenFlow 1.3, sFlow v5

- Hardware
  - OpenFlow-capable switch: HP ProCurve 5412zl
  - Stateful firewall: Cisco ASA 5550 (transparent mode)

- Virtual Machines (VMs)
  - Inside (HTTP Client)
  - Outside (nginx web server)
  - Router
  - Controller

- Linux/KVM virtualization with dedicated physical NICs for VMs
Experiment Setup

► No 10 Gb/s modules
  ▪ 100 Mb/s bottleneck

► sFlow only samples outgoing packets
  ▪ Extra sampling ports

► Traffic
  ▪ Parallel HTTP file downloads
  ▪ Poisson process
  ▪ Mean size: 1 MB
  ▪ $c_{var} = 3$

► Experiment parameters
  ▪ 2000 flow rules (= 1000 bypasses for TCP connections)
  ▪ 300 s timeout
Measured Rate on Firewall
Average Download Time

The graph shows the average download time (s) versus the load (Mb/s). The graph is labeled with different markers for different conditions: w/o, ROff, and IOff. The x-axis represents the load in Mb/s, ranging from 70 to 200. The y-axis represents the average download time in seconds, ranging from 0 to 25. The graph indicates how the average download time changes with increasing load for each condition.
Aborted Downloads

![Graph showing aborted downloads (%) vs load (Mb/s)]
Rule Usage

![Graph showing the relationship between load (Mb/s) and free bypasses with two lines labeled ROff and IOff. The graph indicates a decrease in free bypasses as load increases.](image)

A Software-Defined Firewall Bypass for Congestion Offloading, KuVS FG Network Softwarization, 2017-10-12
Analytic Model & Results

- Bypass density: ratio of no. bypasses to firewall capacity: $\frac{n_{by}}{C}$
- $t_{reuse}$: time after which flow rule can be reused (dominated by timeout)

Analysis: 42%
Experiment: 40%
Summary

► Dynamic bypass: offloads flows accepted by firewall
  ▪ Controller samples traffic from firewall using sFlow
  ▪ Detects congestion and possibly offloads accepted flows using OF
  ▪ Proof-of-concept implementation

► Performance evaluation of dynamic firewall bypass
  ▪ Problem: only few flow rules supported on switches
  ▪ Effective offloading in downscaled experiment

► Theoretical model to predict offloadable traffic
  ▪ Validated by experimental results
  ▪ Switches w/ many flow rules needed for effective offloading

► Accepted at CNSM’17, Tokyo, Japan, Nov. 2017