Workshop OpenFlow

3. Packet filtering (Firewall + Web Interface)

Akbari Indra Basuki
Pusat Penelitian Informatika, LIPI

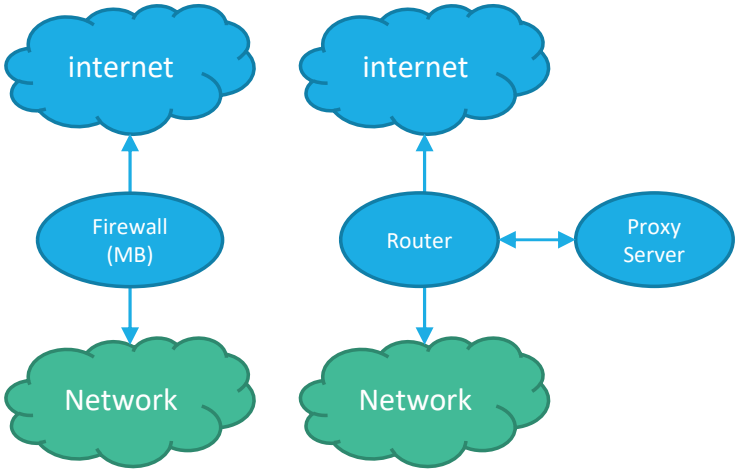
Daftar Materi

Topik	Keterangan
Basic forwarding	Dasar-dasar OpenFlow
Routing & Monitoring	Program Controller: Shortest-path routing Monitor node and link status NetworkX integration
Packet Filtering (Firewall + Web Interface)	Program Controller: Bloom Filter, Flask integration
Load balancing	Group bucket and group tables Round robin load balancing Main-backup path protection
Rate limiting	Meter tables
Stateless vs Stateful data plane Stateful data plane	Jenis data plane dalam memproses paket OpenState SDN Arp handling Port Knocking

Packet filtering (Firewall)

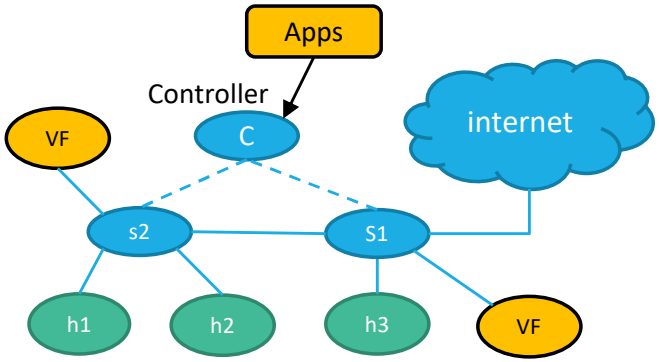
Legacy network:

- Middlebox (MB)
  - Perangkat keras tambahan disisipkan ke dalam jaringan
  - Tidak fleksibel: merubah lokasi middlebox ➔ memutus jaringan atau perlu menyiapkan backup path
- Proxy server (PS)
  - Konfigurasi di end-user
  - Kompatibilitas aplikasi
  - Peletakan server statis



SDN:

- Berbasis Aplikasi Controller
  - Pengecekan dilakukan oleh aplikasi di controller
  - Dapat diupdate dengan algorithma yang lebih baik kedepanya
  - Tidak perlu merubah topologi jaringan
- Berbasis virtual firewall (VF)
  - Pengecekan dilakukan oleh virtual firewall (server)
  - Paket di redirect ke virtual firewall sebelum dikirim ke tujuan asal (chain routing)
  - Tidak perlu merubah topologi jaringan,
  - Cukup merubah aturan routing paket (flow rule)



Studi kasus 1 – Firewall dengan bloom filter

Aplikasi controller bertugas untuk memfilter alamat IP tujuan.

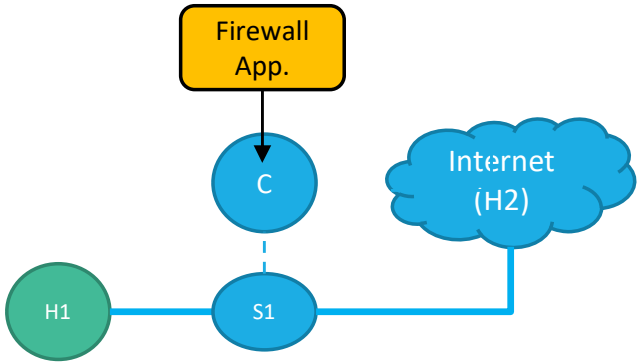
- Apabila alamat IP tujuan termasuk dalam alamat yang diblokir, maka paket akan di drop.
- Apabila alamat IP tidak termasuk dalam daftar blokir, akan di install flow rule untuk memforward paket.

Kemungkinan IP tujuan yang difilter = 2^32 = 4 Milyar alamat IP

- Menggunakan bloom filter → cepat, tersedia banyak library di Python
- Install : sudo pip install bloomfilter

Implementasi (simplefilter.py):

- Aturan *default* IP *forwarding*: Paket dari H1 kirim ke *controller*
- *Controller* menginstall *flow rule* sebagai berikut ke *switch* S1:
  - ***Drop***, jika alamat IP tujuan terdapat di dalam bloom filter
  - ***Output*** ke port 2, jika alamat IP tujuan tidak termasuk di bloom filter



Source code

Tentukan IP destination yang akan di blokir secara acak

```
from bloom_filter import BloomFilter


class MyFilterSwitch(app_manager.RyuApp):
        OFP_VERSIONS = [ofproto_v1_3.OFP_VERSION]
        ### OFP_VERSIONS: menentukan versi OpenFlow yang dipakai, disini memakai versi OF 1.3

        def randomFilter(self, seed, maxcount):
                random.seed(seed)
                def randomIP(upperbound):
                        return str(random.randint(1, upperbound))
                allIP = []
                for i in range(maxcount):
                        newIP =  "10." + "0" + "." + randomIP(10) + "." + randomIP(250)
                        print newIP
                        allIP.append(newIP)
                        self.bloom.add(newIP)
                print "IP address blocked: "+str(len(allIP))

        def __init__(self, *args, **kwargs):
                super(MyFilterSwitch, self).__init__(*args, **kwargs)
                self.hostDB = {}        #berisi pairing antara host.ID - port number
                self.bloom = BloomFilter(max_elements=10000, error_rate=0.1)
                self.randomFilter(12345678, 1000)
```

Install flow rule sesuai status IP destination menurut bloom filter

```
@set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
def _packet_in_handler(self, ev):
        msg = ev.msg
        in_port = msg.match['in_port']
        dp = msg.datapath
        ofproto = dp.ofproto
        dpid = dp.id
        pkt = packet.Packet(msg.data)


        pkt_ipv4 = pkt.get_protocols(ipv4.ipv4)[0]

        if pkt_ipv4:
                dst_ip =  pkt_ipv4.dst
                print "new request: ",dst_ip
                if dst_ip in self.bloom:
                        #pasang flow rule untuk mendrop paket
                        match = dp.ofproto_parser.OFPMatch(in_port=1,eth_type=0x800,ip_proto=0x11,ipv4_dst=dst_ip)
                        actions = []
                        self.add_flow(dp, match, actions,2)
                        print ("install flow rule, match: IP to "+dst_ip+" output:drop")
                else:
                        #pasang flow rule untuk memforward paket ke host 2
                        actions = [dp.ofproto_parser.OFPActionOutput(2 , 0)]
                        data = msg.data
                        out = dp.ofproto_parser.OFPPacketOut(datapath=dp, buffer_id=msg.buffer_id,
                          in_port=in_port, actions=actions, data=data)
                        dp.send_msg(out)

                        #install flowrule untuk memforward paket ke host 2 tanpa menghubungi controller
                        match = dp.ofproto_parser.OFPMatch(in_port=1,eth_type=0x800,ip_proto=0x11,ipv4_dst=dst_ip)
                        actions = [dp.ofproto_parser.OFPActionOutput(2 , 0)]
                        self.add_flow(dp, match, actions,2)
                        print ("install flow rule, match: IP to "+dst_ip+" output:2")
```

Drop

Kirim paket pertamakali secara manual

Output ke H2
Paket berikutnya akan otomatis dikirim tanpa perlu bantuan controller

Langkah pengujian

Pengujian:

- Amati perbedaan jumlah paket yang diterima oleh H2:
  1. Tanpa filter (simpleswitch.py)

     Ryu-manager simpleswitch.py
  2. Dengan aplikasi bloom filter (simplefilter.py)

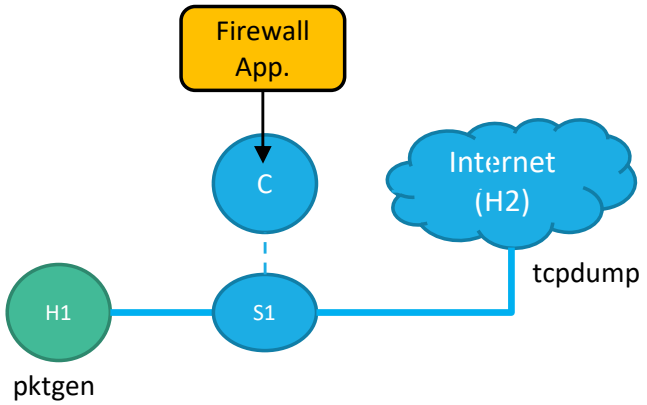
     Ryu-manager simplefilter.py
- Jalankan mininet, topologi: simpleTopo.py

  Sudo python simpleTopo.py
- H2 menjalankan program sniffer (tcpdump)

  Tcpdump –ni h2-eth0

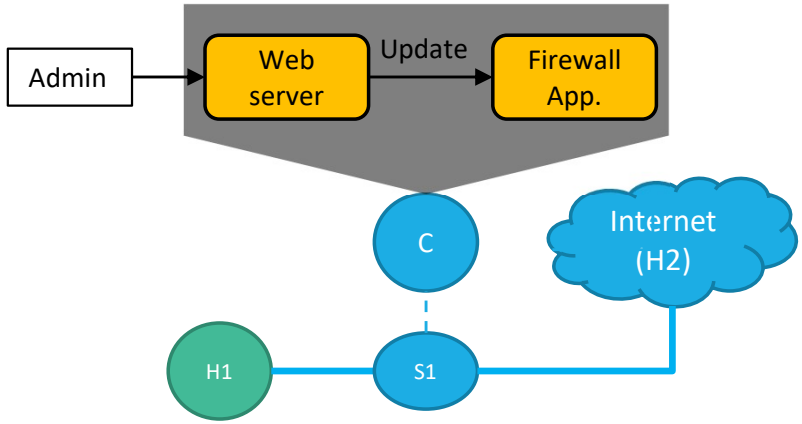
  Ctrl+C untuk mengakhiri program tcpdump
- Kirim paket dari H1 dengan pktgen ke H2

  ./pktgen.sh



Studi kasus 2 – Web interface dengan flask

- Studi kasus 1 belum mendukung input secara manual IP tujuan tertentu ke dalam daftar blokir
- Diperlukan sistem antarmuka antara admin dengan aplikasi controller
- Antarmuka web-server: aksesbilitas via web browser
- Simple server → Flask



Source code

Jalankan webserver flask di thread terpisah menggunakan ryu.lib.hub

```
from ryu.lib import hub
from bloom_filter import BloomFilter
from flask import Flask, session, redirect, url_for, escape, request, Response

class EndpointAction(object):
    def __init__(self, action):
        self.action = action
        self.response = Response(status=200, headers={})

    def __call__(self, *args):
        self.action()
        return self.response

class FlaskAppWrapper(object):
    def __init__(self, name):
        self.app = Flask(name)

    def run(self):
        self.app.run()

    def add_endpoint(self, endpoint=None, endpoint_name=None, handler=None):
        self.app.add_url_rule(endpoint, endpoint_name, EndpointAction(handler))

class MyFilterSwitch(app_manager.RyuApp):
        OFP_VERSIONS = [ofproto_v1_3.OFP_VERSION]

        def _monitor(self):
                self.web = FlaskAppWrapper('wrap')
                self.web.add_endpoint(endpoint='/add', endpoint_name='add', handler=self.action)
                self.web.run()

        def __init__(self, *args, **kwargs):
                super(MyFilterSwitch, self).__init__(*args, **kwargs)
                self.swList = {}        #daftar switch
                self.hostDB = {}        #berisi pairing antara host.ID - port number
                self.bloom = BloomFilter(max_elements=10000, error_rate=0.1)
                self.randomFilter(12345678, 1000)
                self.monitor_thread = hub.spawn(self._monitor)

        def randomFilter(self, seed, maxcount):
                random.seed(seed)
                def randomIP(upperbound):
                        return str(random.randint(1, upperbound))
                allIP = []
                for i in range(maxcount):
                        newIP =  "10." + "0" + "." + randomIP(10) + "." + randomIP(250)
                        allIP.append(newIP)
                        self.bloom.add(newIP)
                print "IP address blocked: "+str(len(allIP))
```

Random daftar IP tujuan yang akan diblokir

Tambahkan IP yang di input ke bloom filter dan install flow rule untuk mendrop paket

```
def action(self):
        def dropNewIP(dst_ip):
                dp= self.swList[1]
                match = dp.ofproto_parser.OFPMatch(in_port=1,eth_type=0x800,ip_proto=0x11,ipv4_dst=dst_ip)
                actions = []
                self.add_flow(dp, match, actions,2)
                print ("update flow rule, match: IP to "+dst_ip+" output:drop")

        if request.method != 'POST':
                ipadr = str(request.args.get('ip'))
                if ipadr in self.bloom:
                        print "IP address: "+ipadr+" already exist!"
                else:
                        print "Adding IP address: "+ipadr+ " into the black list"
                        self.bloom.add(ipadr)
                        dropNewIP(ipadr)
                return "Hello world"
```

Fungsi handler untuk memproses alamat IP yang dikirim via web server

Alamat Url

Langkah pengujian

Pengujian:

- Jalankan aplikasi controller simpleWebFilter.py

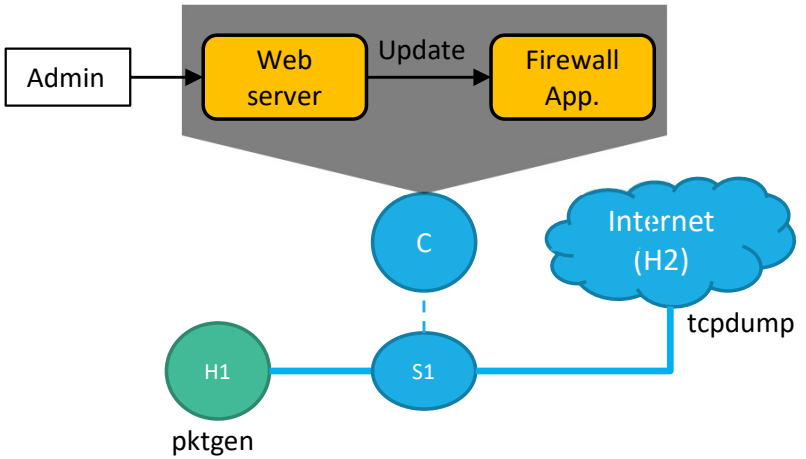
  Ryu-manager simpleWebFilter.py
- Jalankan mininet, topologi: simpleTopo.py

  Sudo python simpleTopo.py
- H2 menjalankan program sniffer (tcpdump)

  Tcpdump –ni h2-eth0
- Kirim paket tunggal dari H1 ke H2 dengan scapy

  Python scapyPktSender.py <ip destination> <mac address destination>
- Tambahkan alamat IP yang barusan dicoba ke daftar terblokir via web browser

  127.0.0.1:5000/add?ip=<ip destination>
- Kirim ulang paket dari H1 dengan alamat IP dan mac address yang sama



Latihan

1. Gunakan algoritma/metode firewall lain untuk menggantikan bloom filter

Daftar pustaka

NoRef