MASTER OF SCIENCE IN ENGINEERING

DDoS Attack Detection & Mitigation in SDN

AOUIDENE Imed

21 – May – 2018

INTRODUCTION

ARCHITECTURE

DEVELOPPMENT DE L'APPLICATION

ENVIRONNEMENT DE SIMULATION

TEST & EVALUATION (DEMO)

INTRODUCTION

Introduction

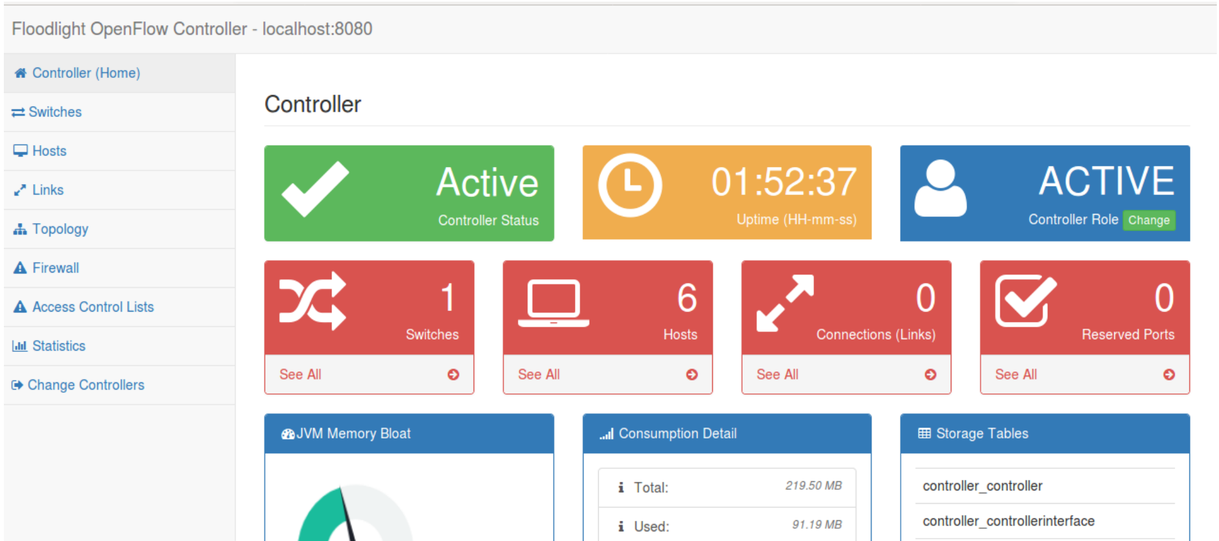
- Attaques de déni de service distribué (DDoS) sont devenus une arme de choix pour les pirates informatiques, et cyber terroristes.
- MALGRÉ le grand nombre de solutions d'atténuation traditionnelles aujourd'hui, DDoS continue de croître en fréquence, en volume, et la gravité.
- Comment détecter et atténuer en temps réel ce genre d’attaques.

ARCHITECTURE

Controller

Controller : FLOODLIGHT

- Un contrôleur OpenFlow basé sur Java et ouvert sous licence Apache
- Open source, actuellement hébergé sur github
- Licence Apache → Utilisation pour tout usage
- Supporte actuellement OpenFlow v1.0 et v1.3.
- Admet un interface web.
- Interaction et contrôle faciles grâce à une API REST



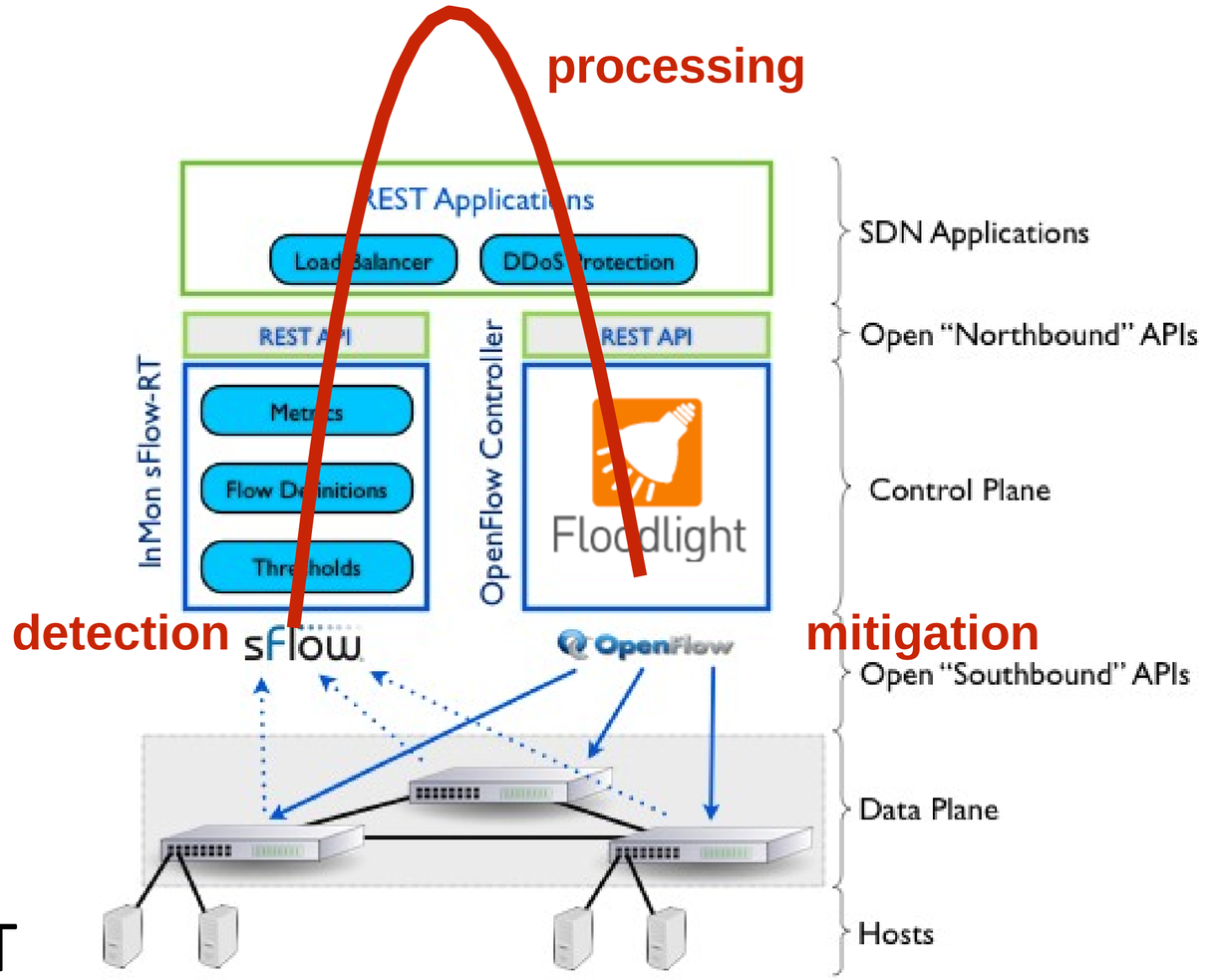
sFlow

- sFlow = sampled Flow
- sFlow-RT reçoit un flux de télémétrie continu de la part des agents sFlow intégrés dans les périphériques réseau et les convertit en métriques exploitables.
- Accessible via une API REST
- Il offre :
  - ✓ configuration des mesures personnalisées
  - ✓ La récupération des métriques
  - ✓ La définition de seuils
  - ✓ La réception de notifications.

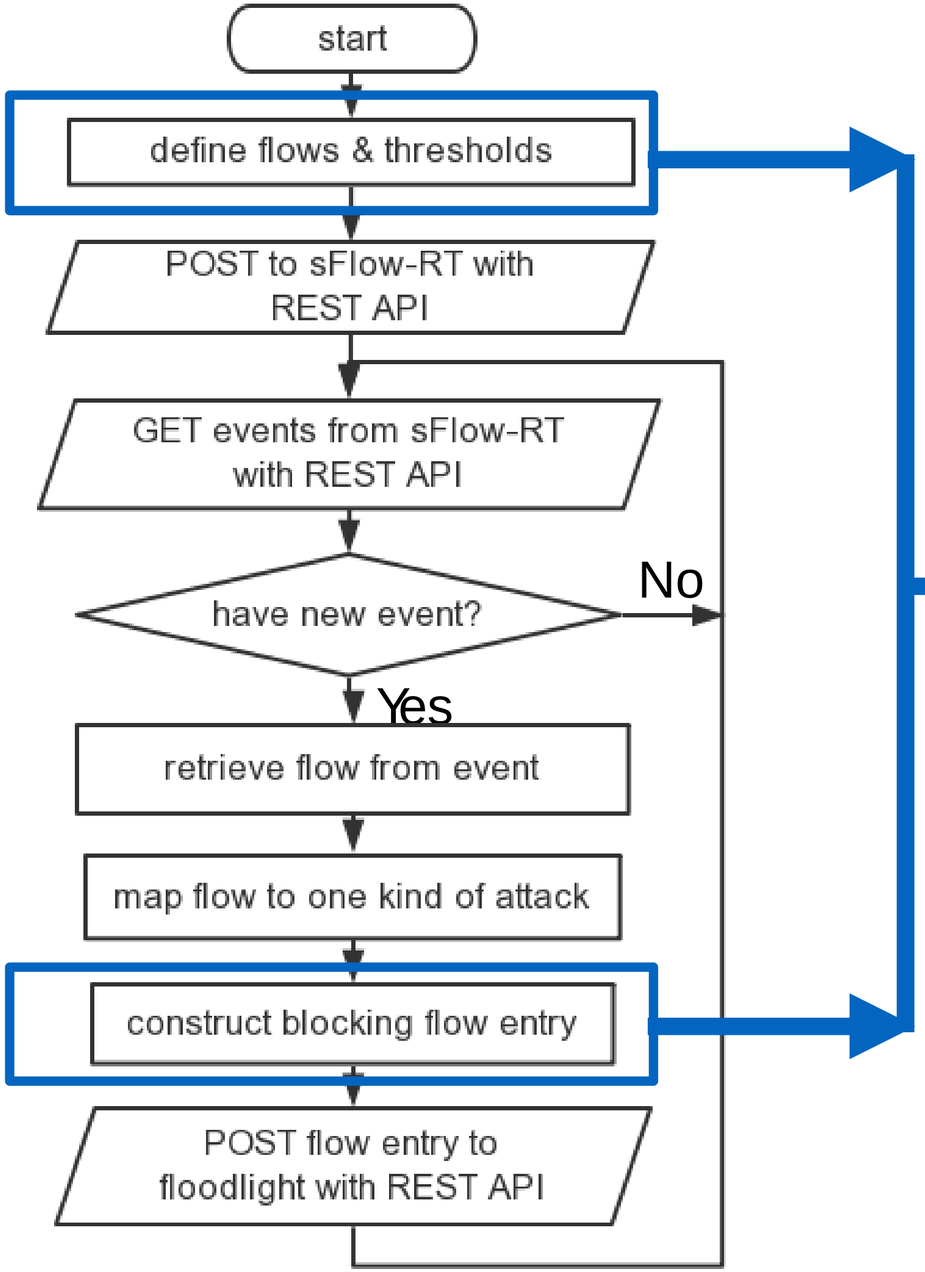
Architecture

sFlow + Openflow

- le commutateur échantillons de paquets
- le commutateur envoie l'en-tête des paquets échantillonnés à sFlow-RT
- En cas de dépassement du seuil, déclencher un événement.
- événements accessibles à partir d'applications externes via l'API REST



Organigramme global de l'application



Doivent être spécifiés pour différents types d'attaques

ICMP Flood Attack

Mechanisme:

Chaque équipement envoie des requêtes ping au serveur à un taux élevé

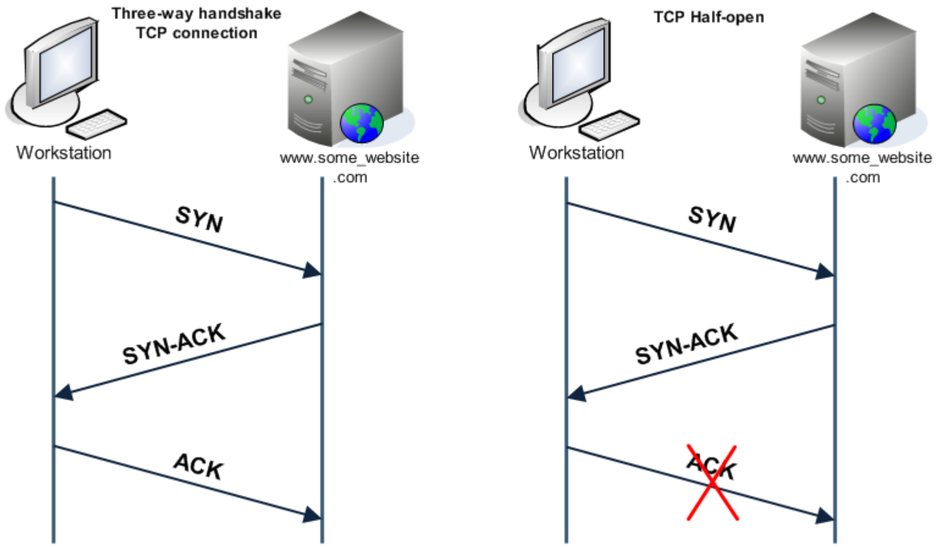
Flow Definition:

```
ipsource=0.0.0.0/0,
ipdestination=10.0.0.2,   #suppose h2 is the server
outputifindex!=discard,   #packet is not discarded
ipprotocol=1  #ICMP
```

SYN Flood Attack

Mechanism:

Chaque équipement envoie des Paquets TCP SYN sur le serveur à un débit élevé.



Flow Definition:

```
ipsource=0.0.0.0/0,
ipdestination=10.0.0.2,   #suppose h2 is the server
outputifindex!=discard,   #packet is not discarded
tcpflags~…….1.=1   #TCP SYN packet
```

APPLICATION DEVELOPMENT

Application Development

python

Importer des « Requests »& « Json » pour exécuter GET / PUT / POST via l'API REST

L'attaque ICMP Flood comme exemple.

Definition of flows, thresholds,…:

```
# ICMP flood attack attributes #
icmp_flood_keys = 'inputifindex,ethernetprotocol,macsource,macdestination,ipprotocol,ipsource,ipdestination'
icmp_flood_metric_name = 'icmp_flood'
icmp_flood_threshold_value = 200
icmp_flood_filter = 'group:ipsource:lf=external&group:ipdestination:lf=internal&outputifindex!=discard&ipprotocol=1'
icmp_flood_flows = {'keys': icmp_flood_keys, 'value': value, 'filter': icmp_flood_filter}
icmp_flood_threshold = {'metric': icmp_flood_metric_name, 'value': icmp_flood_threshold_value}
```

POST the definition to sFlow-RT:

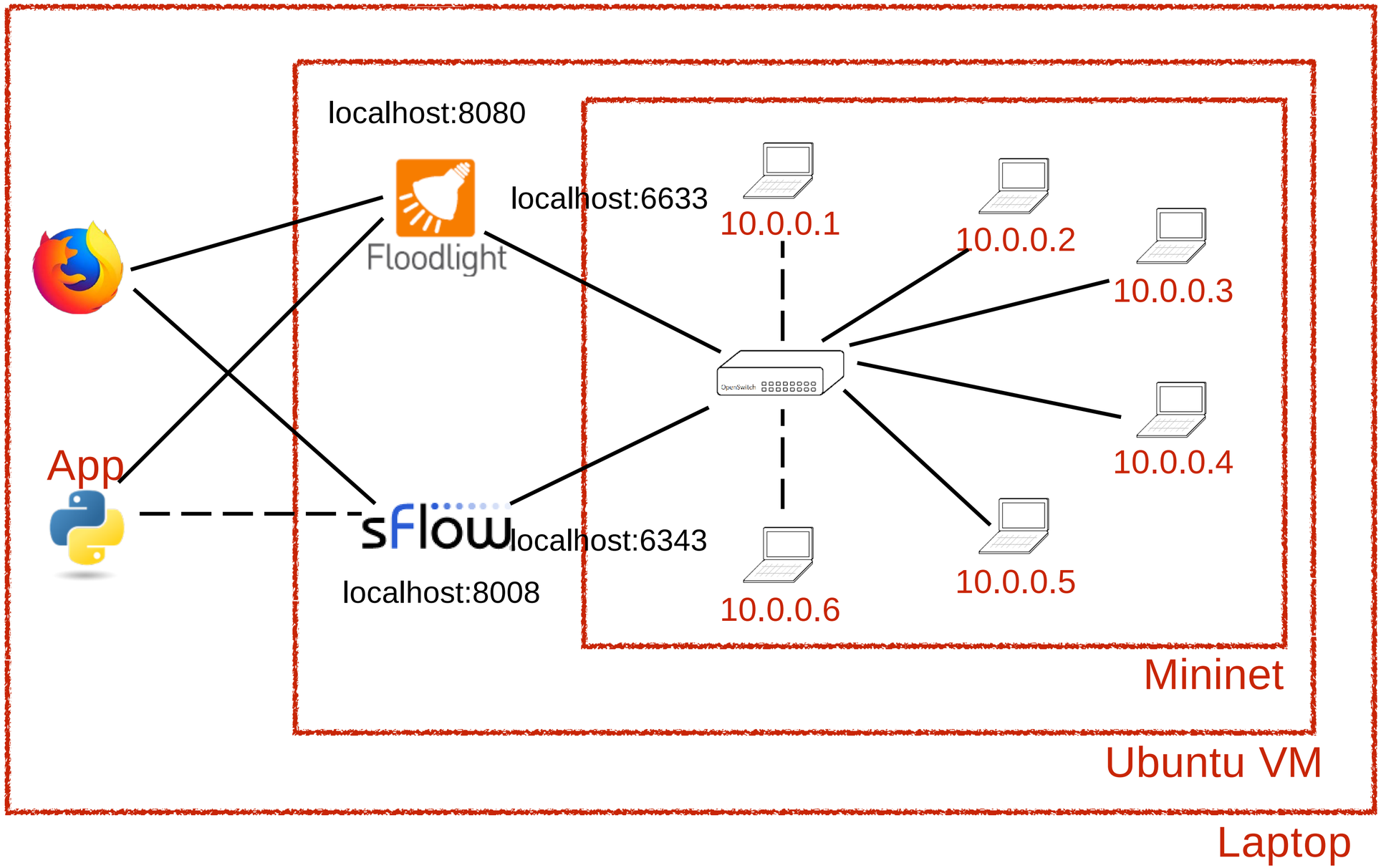
```
# define flows and threshold of ICMP flood
r = requests.put(sFlow_RT + '/flow/' + icmp_flood_metric_name + '/json', data=json.dumps(icmp_flood_flows))
r = requests.put(sFlow_RT + '/threshold/' + icmp_flood_metric_name + '/json', data=json.dumps(icmp_flood_threshold))
```

Attack classification & Static Flow Entry

```
elif e['metric'] == icmp_flood_metric_name:
    r = requests.get(sFlow_RT + '/metric/' + e['agent'] + '/' + e['dataSource'] + '.' + e['metric'] + '/json')
    metrics = r.json()
    if metrics and metrics.__len__() > 0:
        metric = metrics[0]
        if metric.__contains__("metricValue") \
                and metric['metricValue'] > icmp_flood_threshold_value\
                and metric['topKeys']\
                and metric['topKeys'].__len__() > 0:

            for topKey in metric['topKeys']:
                if topKey['value'] > icmp_flood_threshold_value:
                    key = topKey['key']
                    print key,
                    parts = key.split(',')

                    msg = {'name': 'ICMP_block_'+parts[5],
                           'src-ip': parts[5]+'/32',
                           'dst-ip': parts[6]+'/32',
                           'action': 'deny',
                           'nw-proto':'icmp'}

                    new_data = json.dumps(msg)
                    blockrule = {'ruleid':rule_id}
                    rule_id = rule_id+1
                    block_rule = json.dumps(blockrule)
                    #print push_data
                    #r = requests.post(floodlight + '/wm/staticflowentrypusher/json', data=push_data)
                    r = requests.post(floodlight + '/wm/acl/rules/json',data=new_data)
                    black_list.append([time.time()+block_time, block_rule])
                    result = r.json()
```

ENVIRONNEMENT DE TRAVAIL

ENVIRONNEMENT DE TRAVAIL



TEST & EVALUATION

Test & Evaluation

Launch floodlight: ./floodlight.sh

```
mininet@mininet-VirtualBox:~/floodlight$ java -jar target/floodlight.jar
2018-05-21 16:50:48.327 INFO  [n.f.c.m.FloodlightModuleLoader] Loading modules from src/main/resources/floodlightd
efault.properties
2018-05-21 16:50:49.444 WARN  [n.f.r.RestApiServer] HTTPS disabled; HTTPS will not be used to connect to the REST
API.
```

Launch InMon sFlow-RT: ./start.sh

```
mininet@mininet-VirtualBox:~/sflow-rt$ ./start.sh
2018-05-21T16:52:22+0200 INFO: Listening, sFlow port 6343
2018-05-21T16:52:23+0200 INFO: Listening, HTTP port 8008
```

Launch InMon sFlow-RT: sudo python ddos_topo.py

```
mininet@mininet-VirtualBox:~$ sudo python ddos_topo.py
*** Creating nodes
Connecting to remote controller at 127.0.0.1:6653
*** Creating links
*** Configuring hosts
h1 h2 h3 h4 h5 h6
*** Starting network
*** Running CLI
*** Starting CLI:
mininet>
```

set s1 is a sFlow agent, and set up bridge between s1 and sFlow-RT

```
mininet@mininet-VirtualBox:~$ sudo ovs-vsctl -- --id=@sflow create sflow agent=eth0
 target=\"127.0.0.1:6343\" sampling=10 polling=20 -- -- set bridge s1 sflow=@sflow0
b2c0b52a-6be0-4304-995a-d4c3d1671d18
```

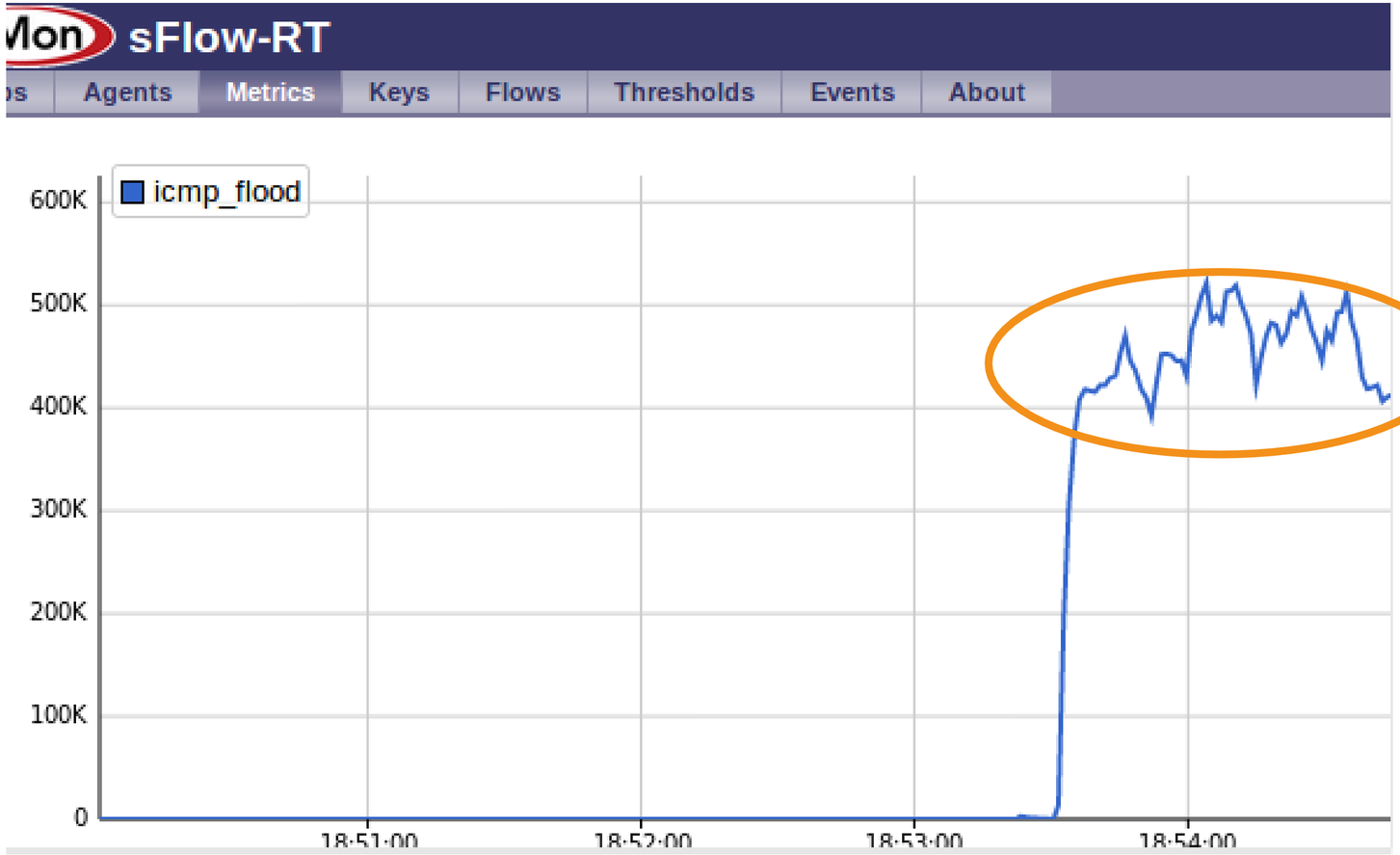
ICMP Flood Attack

Sans mitigation:

h4 ICMP attack on h2 with: *ping -f 10.0.0.2*

```
root@mininet-VirtualBox:~# ping -f 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
```

network traffic flow



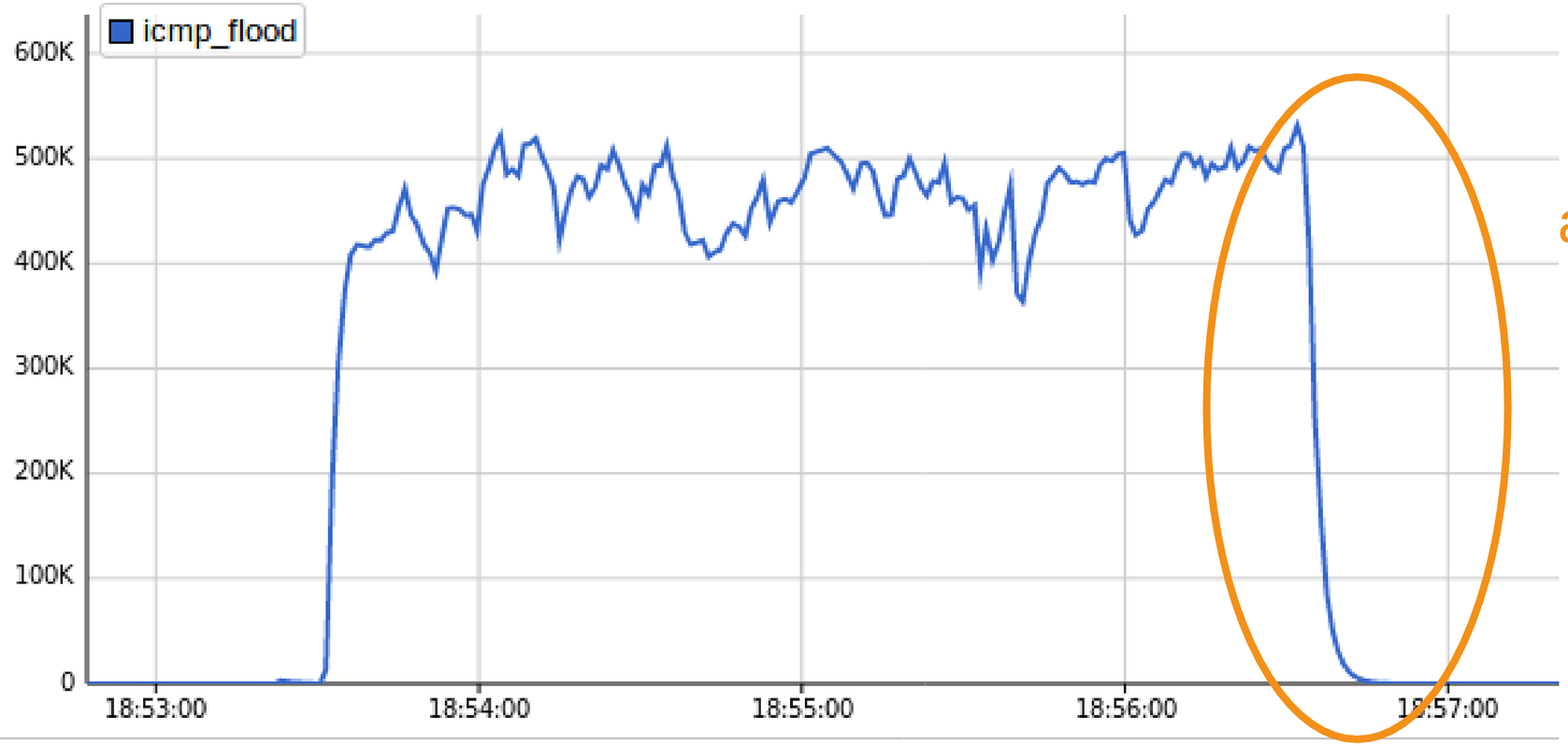
Attack de h4

ICMP Flood Attack

With mitigation:

h4 ICMP attack on h2

network traffic flow



attack from h4 is mitigated

ICMP Flood Attack

Continue: h5 ICMP attack on h2

```
root@mininet-VirtualBox:~# ping -f 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
.....................................................
.....................................................
```

network traffic flow

attack from h5 is mitigated

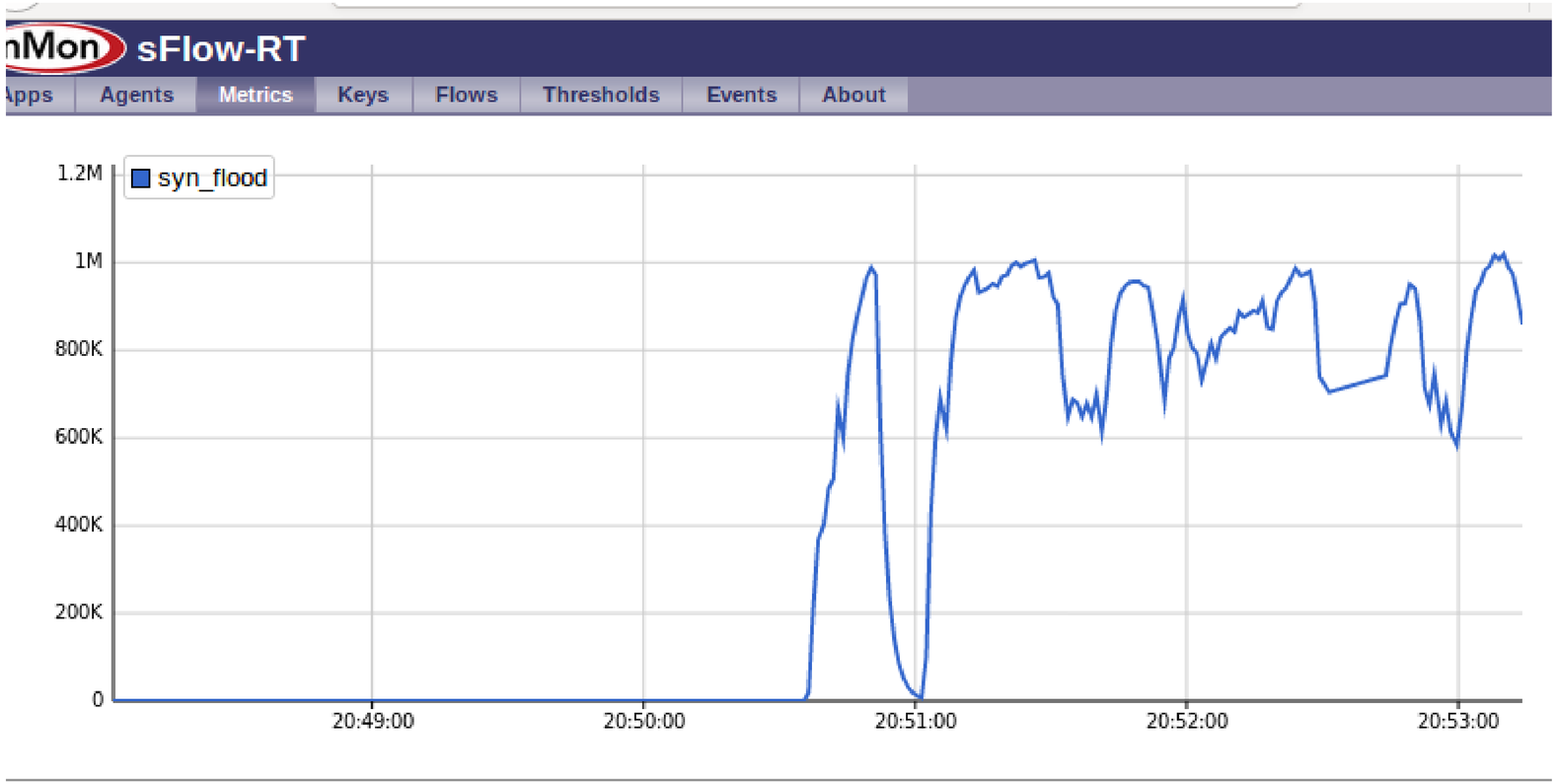
SYN Flood Attack

Without mitigation:

h1 SYN attack on h2 with: *ping —tcp -p 80 —flag syn -rate 2000 —count 20000000 —no-capture —quiet 10.0.0.2*

```
root@mininet-VirtualBox:~# nping --tcp -p 80 --flags syn -rate 2000 --count 200000 --n
o-capture --quiet 10.0.0.2
```

network traffic flow

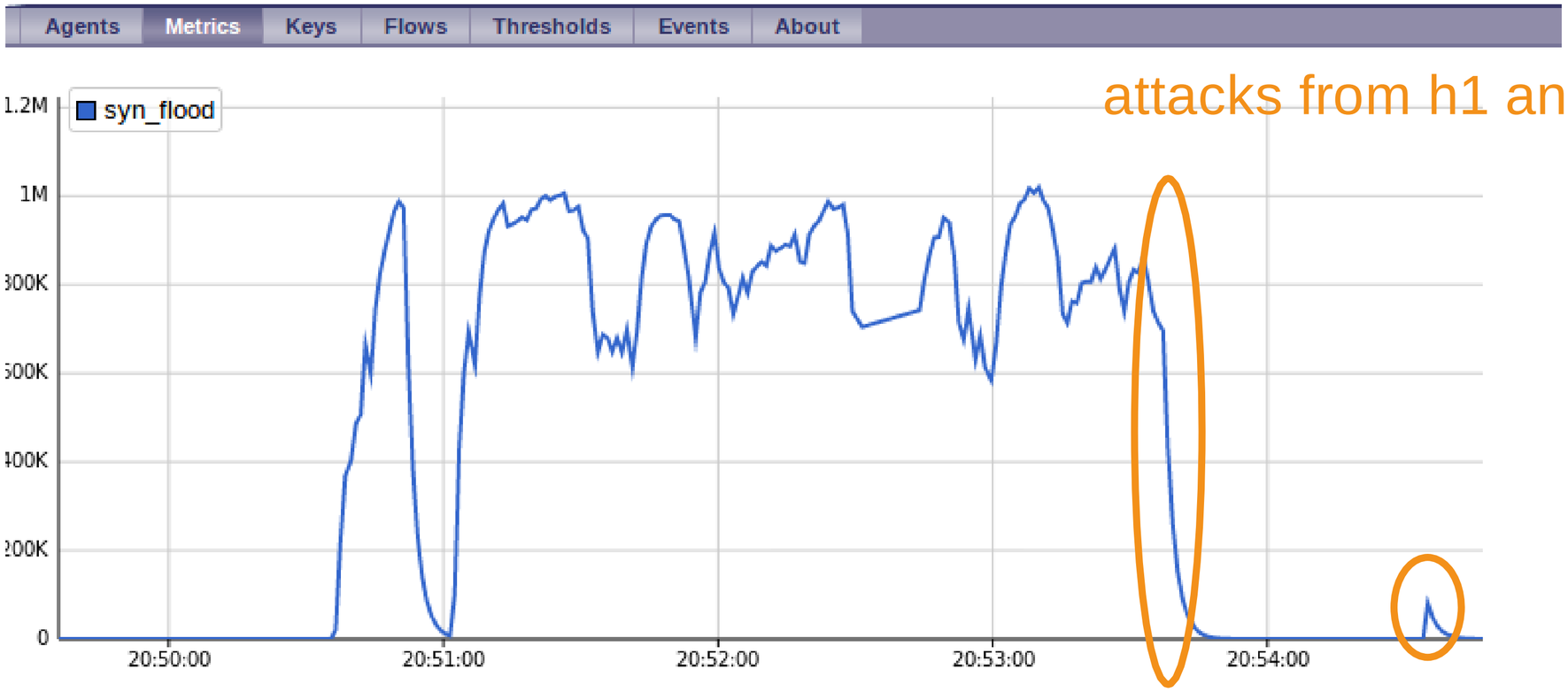


SYN Flood Attack

With mitigation:

h1 and h4 SYN attack on h2

SYN Flood Traffic



attacks from h1 and h4 are mitigated

Merci de votre attention