Projet : SDN et openflow pour le monitoring et la détection d'attaques VoIP SIP (SIPDoS et DDos)

Problématique :

Dans un premier temps, le projet vise à comprendre et assimiler par la pratique, les principes et concepts des réseaux programmables (Software Defined Networks ou SDN) et du protocole « openflow » dans le contexte de la sécurité des réseaux. En effet, les principes du SDN et du protocole Openflow associé, permettent dans certaines conditions, d'avoir une visibilité accrue et globale (end to end) du trafic réseau.
L'objectif du projet sera de tirer parti et d'exploiter cette caractéristique importante pour le monitoring et la détection « rapide » d'un trafic « anormal » (ex attaque de type Deni de Service DoS ou DdoS) Ce projet vise donc à illustrer par la pratique et à travers un cas d'usage spécifique (Attaque DoS, SIP INVITE …) l'intérêt du SDN et openflow pour la sécurité réseau.

Etapes du projet :

1-Définition et déploiement de l'architecture cible sur le simulateur mininet

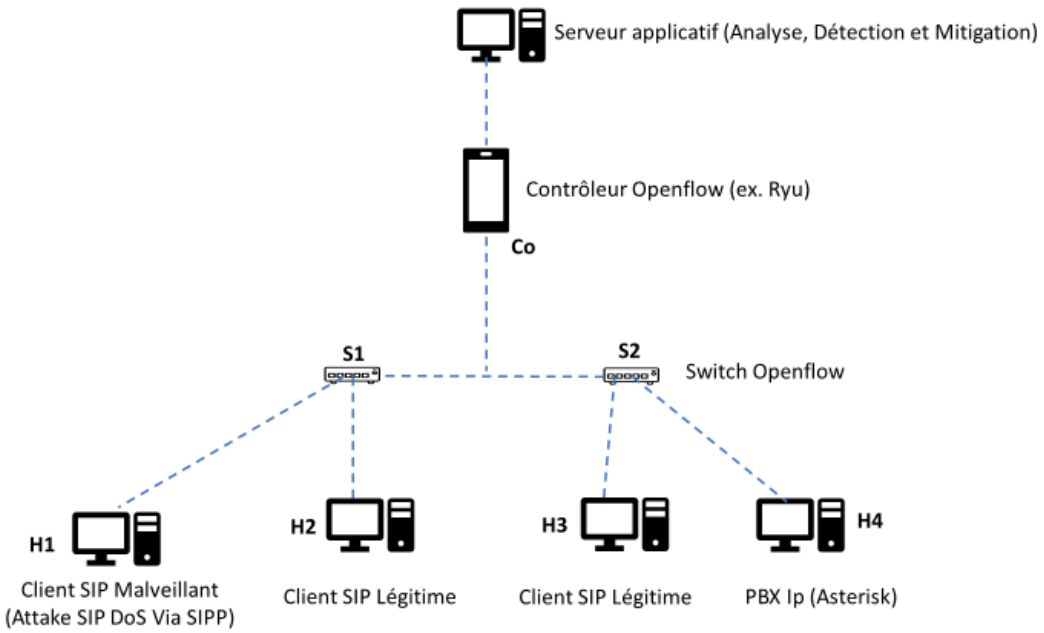
2- Configuration de Minintet, Sflow, (Ryu ou **Floodlight** ?), règles openflow

3-Etude et analyse des règles openflow (flow pipe, OF table…)

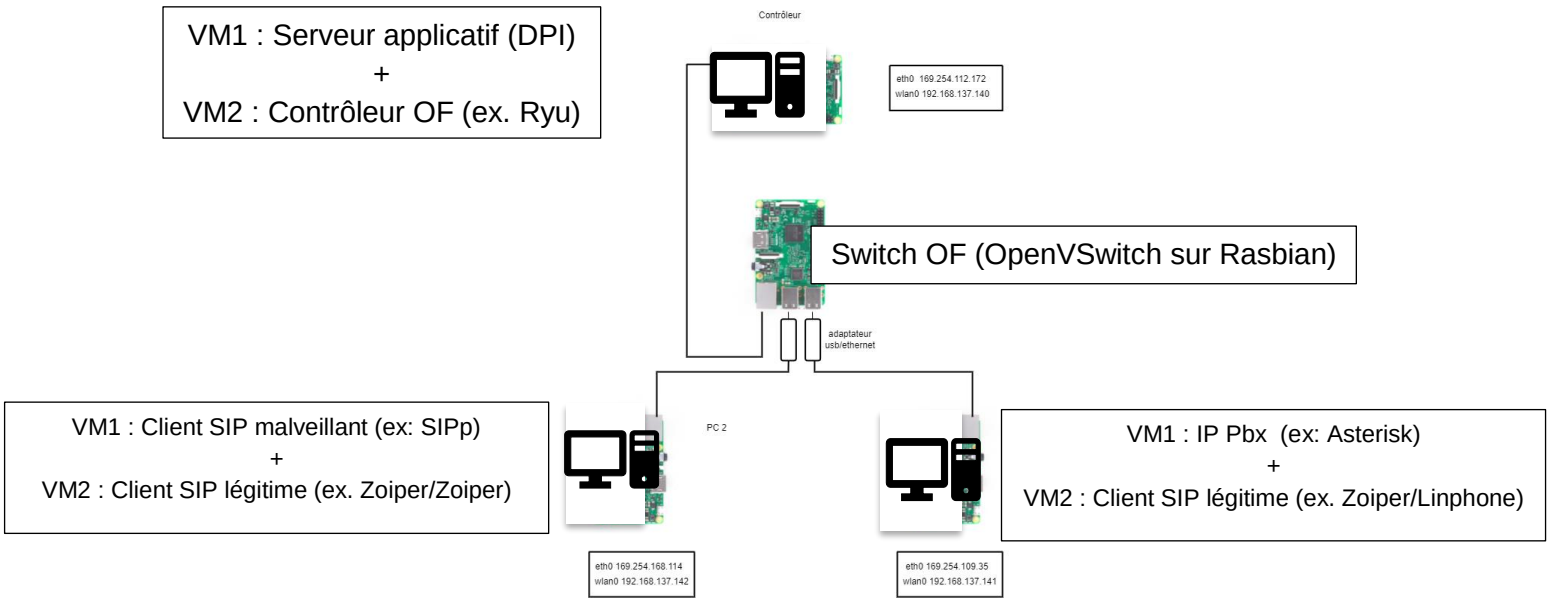
4- Autoriser et vérifier le ping entre 2 machines quelconques du réseau cible.

5- Règle pour filtrer le ping entre deux machines quelconques.

6- Modifier le programme de l'application server (application plane) pour détecter et filtrer une attaque de type DDoS à partir de la machine H1 (seuil de trafic à définir pour le filtrage)



7- (éventuellement en fonction de l'avancement) Déploiement et comparaison sur une plateforme "réelle" à base de cartes Raspberry Pi (si reste du temps)



Ressources :

- ✓ Simulateur Mininet (http://mininet.org/)
- ✓ Wireshark/tcpdump (https://www.wireshark.org/)
- ✓ Contrôleurs SDN au choix (**ryu**, floodlight, POX, NOX)
- ✓ sflowRT(https://sflow-rt.com/)
- ✓ Brique de base d'un Programme Python ("Application Server") pour l'extraction et l'exploitation de données "SDN"

Etude bibliographique préalable (en complément des cours) :

1-Software-Defined Networking: A Comprehensive Survey lien : https://www.researchgate.net/publication/262805723_Software-Defined_Networking_A_Comprehensive_Survey

2-Getting Started With SDN - Abhishek Agarwal. lien: https://medium.com/@click4abhishekagarwal/getting-started-with-sdn-597663e5caef

3-Floodlight SDN Controller Installation on Ubuntu 20.04 / Also run as docker container. lien: https://www.youtube.com/watch?v=UgF8zSHXnGI

4-Quickstart Python — Requests 0.13.9 documentation. (s. d.). python-request lien: https://fr.python-requests.org/en/latest/user/quickstart.html

….

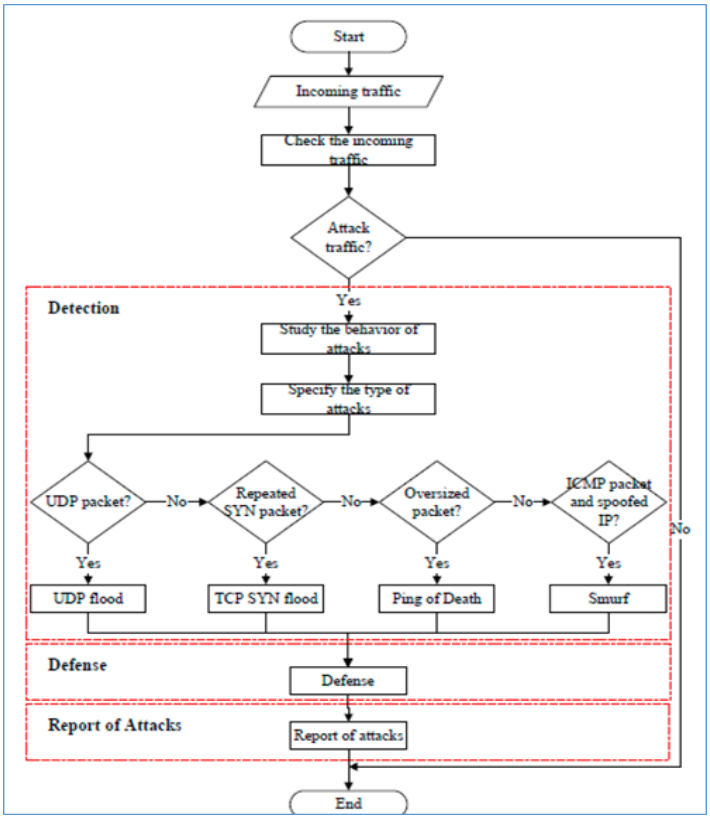
Thèmes abordés :

- Etat de l'art sur les réseaux SDN (usage actuel, perspective M-term L-term)

- Openflow (Southbound API)
- Rest-API (Northbound API)
- Découverte de mininet
- Interaction mininet avec contrôleur
- Création de règles simples et complexes. Discussion/proposition autour d'une politique de règle des contrôleurs SDN.
- Détection d'attaque via sflowRT et script python
- Attaque simple (ping par exemple).
- Attaque multi vecteur (SIP INVITE PACKETS)
- Mitigation des attaques simples et multi-vecteurs.
- Protection du contrôleur contre les attaques DDoS (depuis "l'application plane" par exemple)
- Envoie de msg depuis un équipement OF vers le contrôleur en cas de flux inconnu pour analyse
- En cas de trafic anormal, recopie du trafic pour analyse
- Déploiement d'une "application" firewall/DPI
- …

Perspectives (exemples)

- ✓ Impact de l'activation de TLS sur Asterik et Zoiper
- ✓ explorer la possibilité de détecter/mitiger d'autres types d'attaques ("ping of death", "Smurf", "UDP flood" …) voir diagramme ci-dessous



Source: "Detection and Defense Algorithms of Different Types of DDoS Attacks Using Machine Learning" ; Mohd, A.M.Y.; Fakariah, H.M.A.; Darus, M. In Computational Science and Technology; Springer: Singapore, 2018

Première partie : SIMULATION

Dans cette partie nous allons illustrer les concepts de réseaux SDN et programmables à travers le protocole Openflow.
Les outils utilisés pour réalisés seront (listes non exhaustives) :

- ✓ Mininet
- ✓ Floodlight
- ✓ SflowRT
- ✓ Wireshark

Vous devrez fournir un rapport de TP avec les réponses aux questions posées, les schémas demandés, et tout ce qui vous semblera pertinent.
Il n'y a pas de tutoriel comme vous avez pu en avoir dans vos années d'études antérieures. Vous exploiterez vos connaissances et internet pour progresser étape par étape dans le TP et répondre aux questions.

Mininet

Dans un premier temps, nous allons nous concentrer en particulier sur Mininet. Pour faciliter son installation, nous allons utiliser une machine virtuelle ou mininet est déjà installé.

Vous pouvez télécharger l'image ici : https://github.com/mininet/mininet/releases

Après l'installation (avec VMWare ou VirtualBox), configurez votre réseau (onglet réseau dans VB ou VMWare) de manière à ce que le contrôleur floodlight puisse y accéder par la suite. (NAT ? Bridge ? autres ?)

Quelle configuration réseau avez-vous choisi ? Pourquoi ?

Dans mininet (mn pour les intimes), il existe plusieurs topologies réseaux proposées que nous pouvons configurer via des lignes de commandes.

Est-il possible de personnaliser les topologies et de monter des topologies non disponibles en ligne de commande ? Si oui, comment ?

Quelle est la topologie réseau par défaut ? Comment « prouver » cette topologie via une ligne de commande ?

Décrire la topologie « linear » fournit par mininet. Est-ce une topologie applicable dans le monde réel selon vous (intérêt, cout, résilience, scalabilité, …)

Lancer mininet (sudo mn).
Par défaut, quels éléments ont été créés ? Est-ce cohérent avec votre réponse à la question 3 ?

Une fois le réseau crée, il existe quelques commandes pour décrire le réseau (présence de nœuds, réseau des hosts, nom des hosts, …), donnez quelques commandes utiles (selon vous) et décrivez succinctement leur fonctionnement

Quelles sont les IP de vos « hosts ».

Assurez-vous que vos hosts sont bien interconnectés. Comment vous y prenez-vous ?

Eteignez un des nœuds. Assurez-vous que le (ou les) autres nœuds ne sont plus interconnectés avec le nœud désactivé. Comment procédez-vous ?

Réactivez le nœud (désactivé) et assurez-vous qu'il « ping » à nouveau.

Mininet personnalisé

Avant tout n'oubliez pas de supprimer vos anciennes architectures avant d'en créer une nouvelle. Cela vous évitera de mauvaises surprises, il existe une commande pour « clear » votre architecture avant de relancer « mn ».
Nous allons maintenant tester une topologie un peu plus complexe, la topologie « linear » proposée par mininet. Nous jouerons avec quelques paramètre pour tenter de rendre notre réseau plus en lien avec la réalité (bande passante, latence, …)6

Créez une topologie linear 7, pour rendre cette topologie plus réels, nous allons limiter sa bande passante à 20Mbps et nous ajouterons un « delay » de 3ms.

Donnez le schéma de votre nouvelle topologie

Faites un ping entre vos machines. Le résultat du ping est-il cohérent ? Il existe différente variante du ping dans Mininet (ping full…), n'hésitez pas à les explorer!

Mininet nous permet de faire des tests de performance de notre architecture en mesurant la bande passante par exemple.
Faites un test de performance et interpréter le résultat.

Floodlight

Nous allons utiliser le contrôleur Floodlight en remplacement du contrôleur de base. Nous allons ensuite interconnecter notre contrôleur floodlight avec mininet afin d'observer la topologie.66

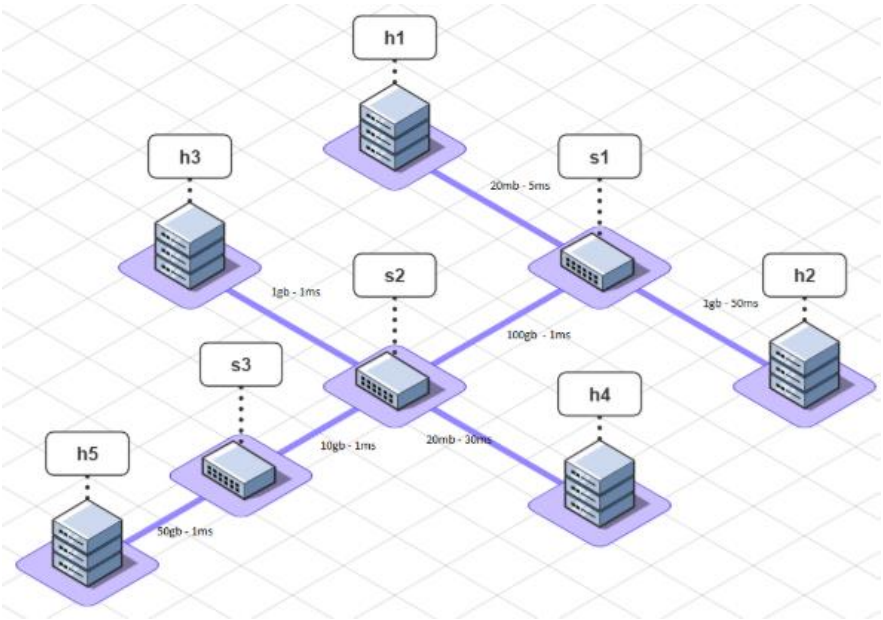
15- Installez floodlight sur votre hyperviseur (donc pas sur la machine mininet).
16- Connectez mininet à floodlight (en précisant à mininet que le contrôleur est « remote » et en précisant l'IP PORT de floodlight)

17- Connectez-vous à l'interface web de floodlight (probablement http://localhost:8080)

Vous pouvez voir votre topologie en cliquant sur l'onglet « topologie » sur la gauche. Explorez l'interface WEB, découvrez les différentes possibilités offertes par l'interface.

Programmation des réseaux.

Dans un second temps, nous allons programmer le déploiement d'un architecture personnalisée.
77A l'aide de python et des bibliothèques mininet, programmez le déploiement de l'architecture suivante. Attention au délai, latence inter liens. Vous trouverez un nombre significatif d'exemples sur internet pour vous aider. A vous de les adapter. Vous fournirez votre code.
N'hésitez pas à vous inspirer des exemples officiels : http://mininet.org/walkthrough/#custom- topologies



19- Vérifiez via votre interface WEB que le réseau a bien été créer (fournir screenshot)
20- Vérifiez que votre réseau fonctionne bien selon les latences attendu. Faites des pings et des tests de perf.

7Ajout d'une première régle

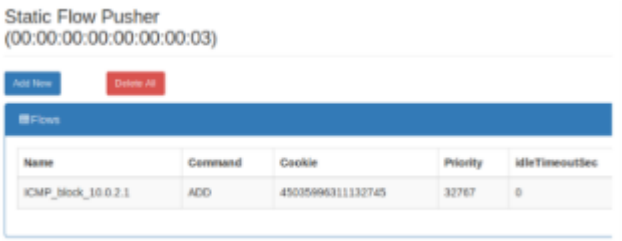
21- En python, écrivez une règle qui interdit l'hôte « h2 d'envoyer des ping vers h1. (indice : ça fonctionne avec du REST et du json)
22- Une fois votre règle chargée, vérifiez à l'aide de l'interface floodlight que la règle est bien appliquée sur les switchs.
23- Faites un test de ping pour vous assurer que les pings vers h2 ne fonctionnent pas (et uniquement vers h2)

Sflow_RT

24- Installez sflow_RT.
25- Maintenant que c'est installé, il est temps que vous vous posiez la question : A quoi sert sflow_RT ?
26- Téléchargez le plugin « flow trend » qui accompagnera sflow_rt pour la suite.
27- Connectez sflow_RT à votre architecture réseau de manière à ce que sflow_RT mesure le flux sur le switch S1.
28- Faites des ings (ou ing ll) t vérifi z d ns l'int rf c WEB d sflow_RT que tout se passe bien.

DDoS et mitigation

Dans ce dernier point, nous allons mettre en place une simulation d'attaque DDoS ICMP Flood Il s'agit8 simplement d'envoyer un grand nombre de ping vers une destination donnée afin de l'inonder.8
Pour flooder, nous utiliserons simplement la commande « ping » avec le parametre « -f ».
Vous lancerez cette commande depuis l'un des hosts de votre architecture sur Mininet.
Exemple : dans la console mininet, tapez « xterm h1 » pour avoir accès à la console de h1, puis ping IP -f.
Pour « mitiger » cette attaque, vous trouverez en annexe un script Python qui fait « presque » le travail. Vous devrez le lire, le comprendre, l'indenter, le commenter, l'adapter à votre besoin et à python 3. (Bref,
il faut le faire marcher !)
Une fois le script exécuté, relancez un flood entre 2 hosts qui transitent via le switch ciblé par sflow.
Si vous avez bien fait les choses, l'host émetteur des pings devrait être bloqué. Vous verrez alors la règle apparaitre dans les règles du switch sur l'UI floodlight Si c'est pas le cas… while (!success) try()



Annexe :

```
import requests import json import time
targetedSwitch = '00:00:00:00:00:00:00:03'

sFlow_RT = 'http://localhost:8008' floodlight = 'http://localhost:8080' defense = {'icmp': True,
 'syn': False,
 'dns_amplifier': False,
 'udp': False} black_list = []
block_time = 360 fw_priority = '32767'
groups = {'external': ['0.0.0.0/0'], 'internal': ['0.0.0.0/0']} # value = 'bytes' # set to 'bytes' and multiply 8 to get
bits/second # define ICMP flood attack attributes #
icmp_flood_keys =
'inputifindex,ethernetprotocol,macsource,macdestination,ipprotocol,ipsource,ipdestination'
icmp_flood_metric_name = 'icmp_flood'
icmp_flood_threshold_value = 5000
#icmp_flood_filter = 'group:ipsource:lf=external&group:ipdestination:lf=internal&outputif
index!=discard&ipprotocol=1'
icmp_flood_flows = {'keys': icmp_flood_keys, 'value': value} # No  filter, the script will monitor every host
icmp_flood_threshold = {'metric': icmp_flood_metric_name, 'value':  icmp_flood_threshold_value}
while True: r = -1
#r = requests.put(sFlow_RT + '/group/json',  data=json.dumps(groups))
r = requests.put(sFlow_RT + '/group/lf/json',  data=json.dumps(groups))
if defense['icmp']:
 # 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))
event_url = sFlow_RT + '/events/json?maxEvents=10&timeout=60' eventID = -1
if black_list.__len__() > 0 and black_list[0][0] < time.time():
r = requests.delete(floodlight +  '/wm/staticflowentrypusher/json', data=black_list.pop(0)[1])
28
 print r.json()['status']
r = requests.get(event_url + '&eventID=' + str(eventID)) events = r.json()

if events.__len__() > 0:
 eventID = events[0]["eventID"] events.reverse()
for e in events:

 if 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'] parts = key.split(',')
 message = {
 'switch':targetedSwitch,
'name': 'ICMP_block_'+str(parts[5]), "cookie":"0",
"priority": fw_priority,
 'ipv4_src': str(parts[5]),
 'ipv4_dst': str(parts[6]), "active":"true",
"eth_type":"0x0800", }
 print message
 push_data = json.dumps(message) r = requests.post(floodlight +
'/wm/staticflowentrypusher/json', data=push_data)

black_list.append([time.time()+block_time, push_data])
 result = r.json()
 print result['status'] break
 else:
 continue
break
 time.sleep(3)
```