Badr Boucherine

Implémentation d’un learning switch avec OpenvSwitch et Ryu

Hepia 2024

Vue d'ensemble	**2**
Sans Shutdown	**2**
Tests	**6**
Conclusion	**9**

Vue d'ensemble

Dans ce projet, nous allons utiliser Ryu pour renforcer la sécurité du réseau en limitant l'accès sur la base du constructeur supposé de la carte Ethernet. En vérifiant que toutes les adresses MAC apprises par le switch ont un Organizational Unique Identifier (OUI) spécifié dans la configuration de l'application. De plus, nous allons implémenter une politique “sans shutdown” sur nos switch afin de sécuriser les ports de ce dernier.

Sans Shutdown

Pour réaliser une politique sans shutdown, c’est à dire qui autorise uniquement une seul MAC adresse par port, j’ai tout d’abord initialiser les variable suivantes :

```
self.swToSecurise = {13402403897345,13615925886977}
self.swMacAuthorized = { 13402403897345 : ["",2], 13615925886977 : ["",2]}
self.portViolationLogCsv = "/root/.ssh/p Notes de publication csv"
self.MAX_VIOLATION = 10
self.violationCountPerPort = { 13402403897345 : [0,2], 13615925886977 : [0,2]}
self.ALLOWED_OUI = ["0c:97:23", "0c:89:fe"]
```

- ***“swToSecurise”*** est une liste qui stocke les différents datapath id des switch auxquels je souhaite appliquer cette politique.
- ***“swMacAuthorized”*** est un dictionnaire qui avec pour clef le datapath du switch, et pour valeur la MAC adresse autorisé pour un port spécifié.
- “***portViolationLogCsv*** est une variable qui stocke le path du fichier des logs dans le contrôleur.
- “***MAX_VILATION***” est une constante qui indique le nombre à atteindre avant de spécifier une violation dans le fichier des logs.

- "***violationCountPerPort***" est un dictionnaire qui à comme clé un switch spécifique est en valeur un compteur qui une fois arriver à la valeur de MAX_VIOLATION ecrit la violation dans le fichier des logs. Il y a un compteur par port.
- ***"ALLOWED_OUi"*** est un tableau qui contient les différents OUI des constructeurs autorisés dans notre infrastructure.

Pour réaliser cette politique de sécurité, il faut modifier la méthode _packet_in_handler qui est appelée à chaque fois que le contrôleur reçoit un paquet qui ne match aucune règle dans la flow table. J'ai commencer par vérifier si le OUI de la mac adresse était paris ceux autorisé:

```
src_oui = src.split(":")[:3]
src_oui_str = ":".join(src_oui)

if  src_oui_str in self.ALLOWED_OUI:
```

Ensuite j'ai ajouté une condition qui va vérifier si le datapath du switch qui transmet la requête est parmi les switch qui implémentent la politique "sans shutdown".

```
if dpid in self.swToSecurise:
```

Si ce derniers est à sécuriser, je vais vérifier si le port source correspond au port qui est à sécuriser :

```
if self.swMacAuthorized[dpid][1] == in_port:
```

Ensuite, si c'est le cas, je vais vérifier si une MAC adresse à déjà été apprise dans le passé avec la condition suivante :

```
if self.swMacAuthorized[dpid][0] == "":
```

Si la valeur est vide, cela signifie que la mac adresse source est la première à utiliser ce port. Je peux donc l'enregistrer comme détenteur de ce port dans le dictionnaire ***swMacAuthorized*** avec les lignes de code suivante:

```
self.swMacAuthorized[dpid][0] = src #Mac Affectation
self.swMacAuthorized[dpid][1] = in_port #Port Affectation
```

Par contre, si une adresse MAC est déjà détentrice de ce port, (c'est à dire que la valeur ***swMacAuthorized[dpid][0]*** n'est pas vie), je dois tout d'abord vérifier que la MAC adresse source du message ne correspond pas à celle de la MAC qui détient ce port avec la condition suivante:

```
if self.swMacAuthorized[dpid][0] != src:
```

Si c'est le cas, le message est transféré normalement, autrement cela signifie qu'une violation de port à été détectée et qu'une autre MAC adresse souhaite utiliser un port qui est déjà réservé.

Lorsqu'une violation est détectée, je commence par incrémenter le compteur qui correspond au port en question (ce dernier permet de compter chaque violation par un port spécifique).

```
self.violationCountPerPort[dpid][0] += 1
```

Ensuite, je vais vérifier si le compteur est égale au nombre de violation maximum avant l'ajout d'une règle dans la flow table du switch:

```
if self.violationCountPerPort[dpid][0] >= self.MAX_VIOLATION:
```

Si c'est le cas, je vais ajouter une ligne dans mon fichier de log ainsi que réinitialiser le compteur avec lignes suivante:

```
current_time = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

with open(self.portViolationLogCsv, mode='a', newline='') as file:
    writer = csv.writer(file)
    if file.tell() == 0:
        writer.writerow(['Date/Heure', 'Switch-ID', 'Port-ID', 'Adresse MAC'])
    writer.writerow([current_time, dpid, in_port, src])

self.violationCountPerPort[dpid][0] = 0
```

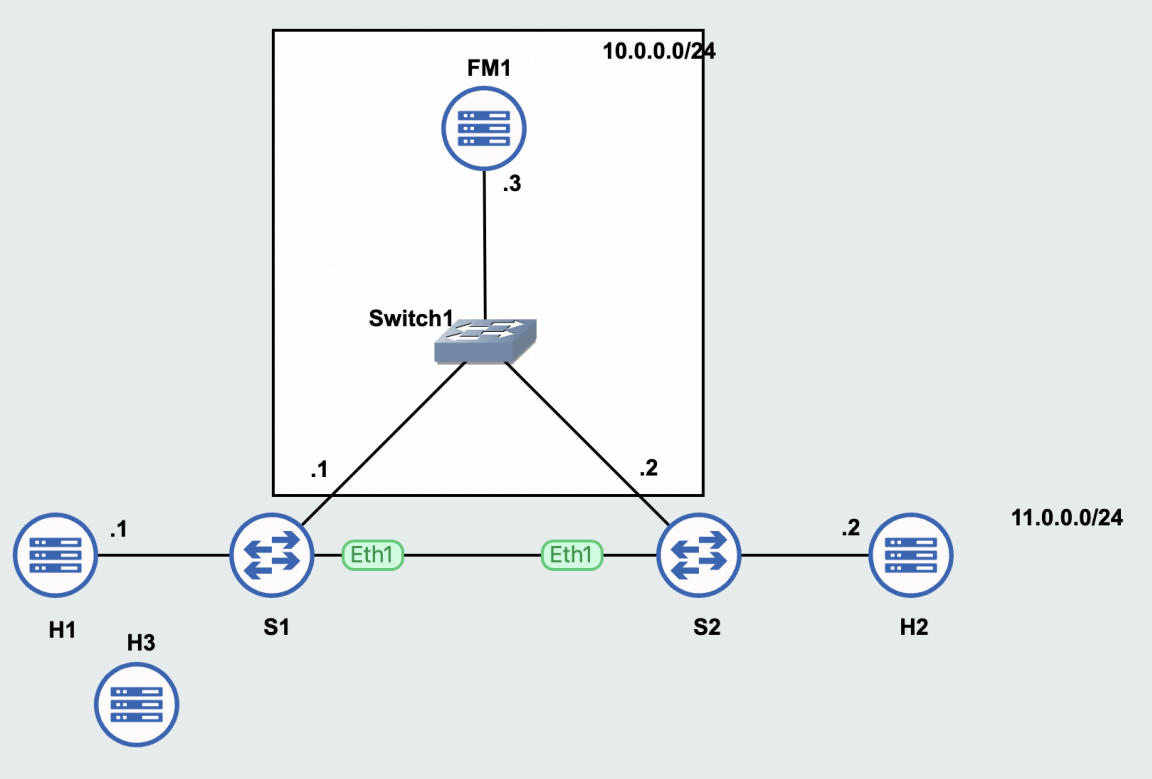
Notes de publication

Une règle DROP est également ajoutée dans la flow table du switch afin de bloquer la MAC adresse qui a commis la violation ce qui permet d'éviter toutes tentatives de flood de ce dernier.

```
if self.violationCountPerPort[dpid][0] >= self.MAX_VIOLATION:
    actions = []
    priority = 65535
    match = datapath.ofproto_parser.OFPMatch(eth_src=src)
    self.add_flow(datapath, priority, match, actions)
```

Tests

La topologie est la suivante:



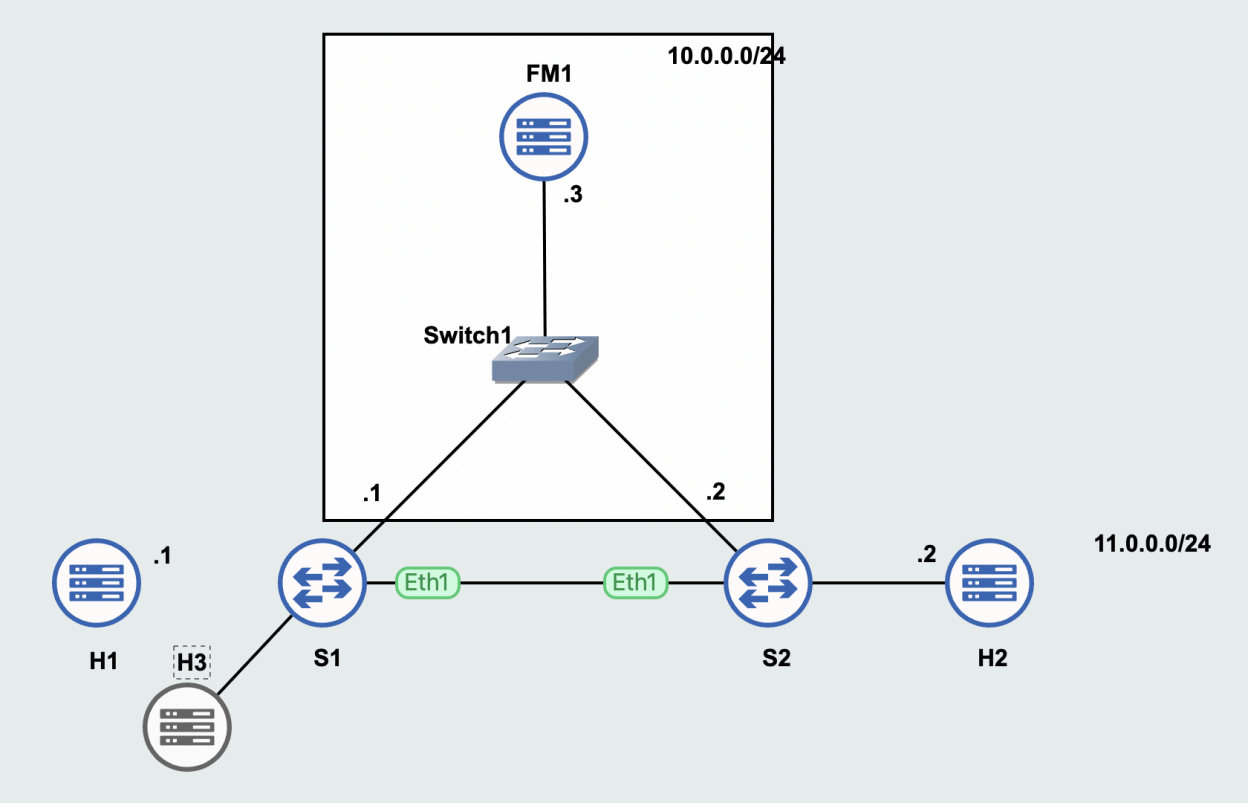
Pour tester mon implémentation, j'ai commencé par s'assurer que lorsque qu'aucun port n'était réservé, la personne pouvait en réserver un et que cela était ajouté dans mon dictionnaire.

J'ai réalisé un ping de l'hôte H1 vers l'hôte H2 et mon tableau à bien été modifié.

```
Nouvelle adresse mac enregistrer , Port: 2 | MAC: 0c:97:23:4e:00:00
{13402403897345: ['', 2], 13615925886977: ['0c:97:23:4e:00:00', 2]}
packet in 13615925886977 0c:89:fe:c9:00:00 0c:97:23:4e:00:00 1
packet in 13402403897345 0c:89:fe:c9:00:00 0c:97:23:4e:00:00 2
Nouvelle adresse mac enregistrer , Port: 2 | MAC: 0c:89:fe:c9:00:00
{13402403897345: ['0c:89:fe:c9:00:00', 2], 13615925886977: ['0c:97:23:4e:00:00', 2]}
```

A noter que le port 2 de S1 ainsi que S2 ont été réservés avec cette unique commande. Cela est dû à la réponse de l'hôte H2 au ping de H1.

En deuxième temps, j'ai remplacé H1 par H3 sur le port 2 pour vérifier que ce dernier était bien sécurisé.



Ma condition de violation à bien été déclenché:

```
Violation !!!, Port: 2 | MAC: 0c:9a:06:e8:00:00
```

```
Violation !!!, Port: 2 | MAC: 0c:9a:06:e8:00:00
packet in 13402403897345 0c:9a:06:e8:00:00 ff:ff:ff:ff:ff:ff 2
Violation !!!, Port: 2 | MAC: 0c:9a:06:e8:00:00
packet in 13402403897345 0c:9a:06:e8:00:00 33:33:00:00:00:16 2
Violation !!!, Port: 2 | MAC: 0c:9a:06:e8:00:00
packet in 13402403897345 0c:9a:06:e8:00:00 ff:ff:ff:ff:ff:ff 2
Violation !!!, Port: 2 | MAC: 0c:9a:06:e8:00:00
packet in 13402403897345 0c:9a:06:e8:00:00 ff:ff:ff:ff:ff:ff 2
Violation !!!, Port: 2 | MAC: 0c:9a:06:e8:00:00
packet in 13402403897345 0c:9a:06:e8:00:00 ff:ff:ff:ff:ff:ff 2
Violation !!!, Port: 2 | MAC: 0c:9a:06:e8:00:00
packet in 13402403897345 0c:9a:06:e8:00:00 ff:ff:ff:ff:ff:ff 2
Violation !!!, Port: 2 | MAC: 0c:9a:06:e8:00:00
packet in 13402403897345 0c:9a:06:e8:00:00 ff:ff:ff:ff:ff:ff 2
Violation !!!, Port: 2 | MAC: 0c:9a:06:e8:00:00
packet in 13402403897345 0c:9a:06:e8:00:00 ff:ff:ff:ff:ff:ff 2
Violation !!!, Port: 2 | MAC: 0c:9a:06:e8:00:00
packet in 13402403897345 0c:9a:06:e8:00:00 ff:ff:ff:ff:ff:ff 2
Violation !!!, Port: 2 | MAC: 0c:9a:06:e8:00:00
Violation Logged !!!, Port: 2 | MAC: 0c:9a:06:e8:00:00
```

Fichier ***portViolationLog.csv*** :

```
2024-04-10 08:19:12,13402403897345,2,0c:9a:06:e8:00:00
2024-04-10 08:19:13,13402403897345,2,0c:9a:06:e8:00:00
2024-04-10 08:19:18,13402403897345,2,0c:9a:06:e8:00:00
2024-04-10 08:19:19,13402403897345,2,0c:9a:06:e8:00:00
2024-04-10 08:19:20,13402403897345,2,0c:9a:06:e8:00:00
2024-04-10 08:19:21,13402403897345,2,0c:9a:06:e8:00:00
2024-04-10 08:19:22,13402403897345,2,0c:9a:06:e8:00:00
2024-04-10 08:19:23,13402403897345,2,0c:9a:06:e8:00:00
2024-04-10 08:19:24,13402403897345,2,0c:9a:06:e8:00:00
2024-04-10 08:19:25,13402403897345,2,0c:9a:06:e8:00:00
2024-04-10 08:19:25,13402403897345,2,0c:9a:06:e8:00:00,Max violation reached
2024-04-10 08:21:57,13402403897345,3,0c:9a:06:e8:00:00,OUI Violation
```

Lors d’une violation spécifique (OUI violation ou lorsque le maximum de violation est atteint pour un port donné), cette dernière est spécifiée dans un champ nommé ‘Alert’.

En ce qui concerne les “organizationally unique identifier” des MAC adresse, j’ai ajouté uniquement les OUI des MAC adresse des hôtes H1 ainsi que H2 comme autorisé pour vérifier le bon fonctionnement de mon code. Lorsque j’essaie de pinger H2 avec H3 cela déclenche une violation de OUI:

```
packet in 13402403897345 0c:9a:06:e8:00:00 0c:97:23:4e:00:00 2
Violation de sécurité: OUI non autorisé pour MAC 0c:9a:06:e8:00:00
```

```
root@debian:~# ping 11.0.0.2
PING 11.0.0.2 (11.0.0.2) 56(84) bytes of data.
^C
--- 11.0.0.2 ping statistics ---
9 packets transmitted, 0 received, 100% packet loss, time 8187ms
```

En plus d’être loggé, une règle DROP est ajoutée dans la flow table du switch (identique à celle des violation de port) afin de bloquer l'accès à cette MAC adresse à l’infrastructure.

```
# If OUI is not authorized
self.logger.info("Violation de sécurité: OUI non autorisé pour MAC %s" %src)
actions = []
priority = 65535
match = datapath.ofproto_parser.OFPMatch(eth_src=src)
self.add_flow(datapath, priority, match, actions)
self.logger.info("Added flow to ignore the following MAC adress:  %s" %src)
```

Conclusion

Pour conclure, ce travail m'a permis de comprendre le fonctionnement des switchs OpenFlow ainsi que la librairie Ryu qui permet de gérer les différents appareils réseaux. J’ai trouvé ce travail pratique très intéressant car il propose une vision totalement différente en ce qui concerne la sécurité ainsi que la gestion des différents appareils en comparaison aux visions plus traditionnelles connues par tous. Cette méthode de travail horizon énorme à l’aide de la programmation logiciel. Il facilite également grandement l'innovation dans le domaine.