Mitigation des attaques de déni de service dans les réseaux définis par logiciel

par

Lobna DRIDI

MÉMOIRE PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
COMME EXIGENCE PARTIELLE À L'OBTENTION DE LA MAÎTRISE
AVEC MÉMOIRE EN GÉNIE CONCENTRATION
RÉSEAUX DE TÉLÉCOMMUNICATIONS
M.Sc.A.

MONTRÉAL, LE 11 SEPTEMBRE 2017

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

PRÉSENTATION DU JURY

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE:

M. Mohamed Faten Zhani, Directeur de Mémoire
Département de génie logiciel et des TI à l'École de technologie supérieure

Mme. Latifa Guerrouj, Présidente du Jury
Département de génie logiciel et des TI à l'École de technologie supérieure

M. Jean-Marc Robert, membre du jury
Département de génie logiciel et des TI à l'École de technologie supérieure

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 30 AOÛT 2017

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

Au terme de ce projet, je tiens à exprimer ma profonde gratitude et mon immense respect à Monsieur Mohamed Faten Zhani, Professeur à l'École de technologie supérieure, pour m'avoir encadrée et guidée tout au long de la réalisation de ce travail.

Avec beaucoup d'égard, je ne manquerai pas d'exprimer ma grande reconnaissance à tous les membres du jury pour avoir accepté d'évaluer ce modeste travail.

De même, je souhaite transmettre l'expression de ma reconnaissance et ma plus profonde gratitude à mes collègues pour leur aide aussi bien technique que morale.

Mes vifs remerciements à mes parents, mon frère et ma sœur pour leur soutien moral dans les périodes difficiles. Je remercie aussi mon mari qui m'a soutenue et encouragée pour achever ce travail.

Mes remerciements vont enfin à toute personne qui a contribué de près ou de loin à l'élaboration de ce travail.

MITIGATION DES ATTAQUES DE DÉNI DE SERVICE DANS LES RÉSEAUX DÉFINIS PAR LOGICIEL

Lobna DRIDI

RÉSUMÉ

La réseautique définie par logiciel (SDN) est une nouvelle technologie réseau offrant la programmation nécessaire pour permettre aux opérateurs réseau de gérer leurs infrastructures d'une façon simple et dynamique. Cependant, malgré ces avantages, ces réseaux sont très vulnérables aux attaques de déni de service (DdS) qui peuvent facilement surcharger le contrôleur SDN et les tables de commutation des commutateurs, ce qui entraîne une dégradation critique des performances du réseau.

Pour résoudre ces problèmes, nous proposons SDN-Guard, une nouvelle approche pour protéger les réseaux SDN contre les attaques de déni de service et atténuer leurs répercussions sur la performance du réseau. SDN-Guard exploite un système de détection d'intrusion (IDS) pour détecter les éventuelles attaques DdS et atténuer efficacement leur impact par le réacheminement du trafic malveillant, l'ajustement des *timeout* des règles de commutation et l'agrégation de ces règles dans les tables TCAM des commutateurs. De plus, nous cherchons des solutions pour minimiser le trafic entre les commutateurs et l'IDS sans affecter la précision de détection de l'IDS. Nous proposons donc d'utiliser les techniques d'échantillonnage de trafic et nous établissons un Programme Linéaire en Nombres Entiers (PLNE) pour trouver l'emplacement optimal de l'IDS qui permet de déterminer les commutateurs qui devraient dupliquer leurs flux vers l'IDS afin de minimiser la consommation de bande passante du réseau.

Les expériences que nous effectuons montrent que SDN-Guard maintient la performance du réseau pendant les attaques DdS et réussit à réduire leur impact sur la performance du contrôleur, l'utilisation des tables TCAM du commutateur et de la bande passante du plan de contrôle. De plus, SDN-Guard réduit, d'une manière significative, la perte des paquets ainsi que leur temps moyen de transmission dans le réseau durant les attaques DdS. Les résultats prouvent aussi que le fait de placer soigneusement l'IDS et de bien sélectionner les commutateurs, qui vont dupliquer le trafic pour l'analyse, réduit de 90% le trafic entre les commutateurs et l'IDS. Ils montrent également que la précision de l'IDS reste à 100% en analysant seulement 11% du trafic réseau.

Mots clés: Réseautique définie par logiciel (SDN), sécurité des réseaux, attaque de Déni de Service (DdS), précision des IDSs, emplacement de l'IDS, échantillonnage de trafic

MITIGATION OF DENIAL OF SERVICE ATTACKS IN SOFTWARE DEFINED NETWORK

Lobna DRIDI

ABSTRACT

Software Defined Networking (SDN) has recently emerged as a new networking technology offering an unprecedented programmability that allows network operators to dynamically manage their infrastructure. However, despite these benefits, Deny-of-Service (DoS) attacks are considered a major threat to such networks as they can easily overload the SDN controller and flood switch CAM tables, resulting in a critical degradation of the network performance.

To address this issue, we propose SDN-Guard, a novel holistic approach to protect SDN networks against DoS attacks. SDN-Guard leverages an Intrusion Detection System (IDS) to detect potential DoS attacks and then efficiently mitigate their impact by dynamically rerouting malicious traffic, adjusting flow timeouts and aggregating flow rules. Furthermore, we investigate for solutions to minimize the switch-to-IDS traffic without impacting the IDS accuracy. We hence propose to use sampling techniques and devise an Integer Linear Program to find the optimal placement for the IDS and to determine the switches that should mirror the flows towards it so as to minimize network bandwidth consumption.

Extensive experiments show that SDN-Guard maintains network performance during DoS attacks and succeeds in reducing their impact on the controller performance, switch TCAM usage and control plane bandwidth. Moreover, SDN-Guard can significantly reduce packet loss and average packet round trip time in the network during DoS attacks. Furthermore, our results show that carefully placing the IDS and selecting the switches mirroring the traffic can reduce by up to 90% the switch-to-IDS traffic. They also show that the IDS accuracy remains at 100% by analyzing only 11% of the network traffic.

Keywords: Software Defined Networking (SDN), network security, DoS attacks, IDS accuracy, IDS placement, traffic sampling

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

AIDS	Application based Intrusion Detection System
API	Application Programming Interface
ARP	Address Resolution Protocol
BGP	Border Gateway Protocol
CAM	Content Addressable Memory
CPU	Central Processing Unit
DdS	Déni de Service
DDdS	Déni de Service Distribué
FAI	Fournisseur d'Accès à Internet
FIB	Forwarding Information Base
FML	Forge Mod Loader
ForCES	Forwarding and Control Element Separation
HIDS	Host based Intrusion Detection System
ICMP	Internet Control Message Protocol
IDPS	Intrusion Detection and Prevention System
IDS	Intrusion Detection System
IJNM	International Journal on Networking Technologies
ILP	Integer Linear Program
IP	Internet Protocol
IRS	Interface to the Routing System
MAC	Media Access Control
NetConf	Network Configuration Protocol
NIDS	Network based Intrusion Detection System

NIPS	Network based Intrusion Prevention System
OF	OpenFlow
OISF	Open Information Security Foundation
ONF	Open Networking Foundation
OSPF	Open Shortest Path First
OVS	Open VSwitch
PLNE	Programme Linéaire en Nombres Entiers
QoS	Quality of Service
REST	Representational State Transfer
RR	Round Robin
RTT	Round Trip Time
SDN	Software Defined Networking
sFlow	sampled Flow
SSH	Secure SHell
STP	Spanning Tree Protocol
TCAM	Ternary Content Addressable Memory
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VLAN	Virtual Local Area Network
VNF	Virtual Network Function
WRR	Weighted Round Robin

LISTE DES SYMBOLES ET UNITÉS DE MESURE

GB	Giga Byte
GHz	Giga Hertz
min	minutes
msec	millisecond
k-pkt	kilo packet
sec	second
μsec	microsecond

INTRODUCTION

Contexte

Avec la croissance de l'utilisation de la technologie d'information, il y'a une énorme augmentation du trafic qui circule dans les réseaux à cause du grand nombre des périphériques connectés et des applications internet modernes, telles que les réseaux sociaux et le partage de documents. Les administrateurs de réseau doivent gérer une large gamme de formats de données, de types de services et de périphériques, ce qui est difficile avec les outils de gestion des réseaux traditionnels qui n'ont pas été conçus pour faire face à des topologies évolutives à très grande échelle.

Le concept de réseau défini par logiciels (Software Defined Networking - SDN) est la solution pour répondre aux besoins des utilisateurs de ces services et ces applications réseau. L'architecture du SDN permet aux opérateurs du réseau de réagir et gérer dynamiquement leurs infrastructures face aux changements de comportement du trafic dans le réseau. SDN permet la séparation entre le plan de contrôle, qui est responsable à la prise de décision du routage de flux dans le réseau et de la gestion des services et des applications offerts par ce dernier, et le plan de données (ou de transmission), qui est en charge de transmettre les flux de données suivant les directives fournies par le plan de contrôle.

Cette approche centralise et simplifie la gestion du réseau, ce qui permet aux administrateurs de l'orchestrer et l'automatiser à travers une interface de contrôle logicielle et centrale sans accéder physiquement aux composants matériels (routeurs, commutateurs, etc.). Par conséquent, les administrateurs du réseau peuvent facilement gérer le trafic en modifiant les règles de commutation d'un commutateur directement depuis le contrôleur SDN afin de changer les priorités et les directions des flux. Grâce à cette simplicité, SDN devient le meilleur choix pour les fournisseurs d'internet, vu le grand nombre des équipements utilisés et la quantité énorme des données à gérer.

Comme la technologie SDN gagne du terrain et que plusieurs fournisseurs d'internet et d'administrateurs de centres de données l'adoptent progressivement, on s'intéresse de plus en plus aux problèmes de sécurité qui pourraient se poser en ce qui concerne son déploiement en production. Une enquête récente sur les menaces de la sécurité dans les réseaux SDN classifie les types des attaques selon des catégories comme l'accès non autorisé, la modification des données, les applications malicieuses, le déni de service (DdS) et les problèmes de configuration (Scott-Hayward, S., O'Callaghan, G. et Sezer, S., 2013). Les résultats de cette étude montrent aussi que le plan de contrôle et le plan de données sont toujours infectés simultanément dans la plupart des attaques, ce qui explique la relation complémentaire entre ces deux plans dans le concept du réseau SDN.

L'architecture centralisée du SDN s'avère comme une arme à double tranchant. D'une part, elle donne une vue globale pour l'administrateur du réseau à travers le contrôleur afin de faciliter la gestion du réseau. D'autre part, cette centralisation est une vulnérabilité pour les réseaux SDN.

Dans ce travail, nous nous intéressons aux attaques de déni de service (DdS). Ce type d'attaque a pour but de surcharger un réseau ou un serveur ciblé en l'inondant d'une manière agressive avec une grande quantité de trafic jusqu'à ce qu'il devienne indispensable de servir ces usagers légitimes. Plusieurs travaux de recherches tels que (Kandoi, Rajat et Antikainen, Markku, 2015) et (Shin, Seungwon et Gu, Guofei, 2013) ont prouvé que ces attaques DdS peuvent avoir de graves effets sur la performance du réseau SDN, conduisant à des cas où tout le plan de contrôle devient totalement paralysé.

Problématique

Les attaques de DdS sont les menaces les plus populaires et inévitables pour les réseaux SDN comme ils l'ont toujours été pour les réseaux traditionnels. Dans un réseau SDN, le DdS peut entraîner les problèmes suivants :

1) la surcharge du contrôleur SDN : selon les spécifications de SDN (Open Networking Fundation, 2013), le contrôleur est la seule entité responsable de la décision de routage des flux dans le réseau. Lorsque le commutateur reçoit un paquet appartenant à un nouveau flux, il demande au contrôleur de lui attribuer une règle de flux pour l'utiliser afin de transmettre le paquet au prochain nœud. Lors d'une attaque de DdS, à cause de l'énorme nombre des paquets envoyés par l'attaquant et qui nécessitent des règles de flux pour joindre la destination demandée, les commutateurs envoient un grand nombre de demandes des règles de flux au contrôleur. Par conséquent, le contrôleur sera surchargé, les messages de demande des règles de flux vont saturer dans la file d'attente du contrôleur et par la suite, aucune décision de routage ne peut être prise pour les nouveaux flux. Dans ce cas, les flux sans règle seront soit bloqués dans les files d'attente des commutateurs, soit dirigés vers une route par défaut ;
2) l'épuisement de la bande passante du plan de contrôle : ce problème est étroitement lié au précédent. Due à la communication excessive entre le contrôleur et les commutateurs demandant des nouvelles règles de flux pour envoyer les paquets à la destination, la liaison d'un commutateur vers le contrôleur peut être congestionnée et certaines demandes de règles de flux peuvent être perdues. Cela engendre un retard dans la décision de routage pour les flux en attente ;
3) le dépassement de la mémoire TCAM du commutateur : chaque commutateur possède une mémoire de stockage bien déterminée pour stocker les règles des flux dans leurs tables de commutation. En cas d'une attaque de DdS, l'attaquant envoie un grand nombre de flux ; cela engendre la saturation des tables de commutation du commutateur et donc de sa mémoire TCAM (Margaret, Rouse, 2013) à cause de sa capacité de stockage limitée. Lorsque cela se produit, les commutateurs sont forcés d'ajouter et de supprimer continuellement des entrées de flux dans leurs tables et d'envoyer plus de paquets vers le contrôleur.

Dans ce travail, nous proposons une solution pour protéger les réseaux définis par logiciel contre les attaques DdS et atténuer leur impact sur les performances du contrôleur SDN, l'utilisation de la bande passante du plan de contrôle et l'utilisation de la mémoire du commutateur. Contrairement aux travaux existants, notre solution est conçue pour atténuer simultanément ces problèmes en gérant dynamiquement le routage des flux dans le plan de données, les règles de commutation dans les commutateurs, en se basant sur des alertes générées par un système de détection d'intrusion (IDS).

Objectif et contributions

L'objectif principal de ce travail est de concevoir et de développer une nouvelle solution qui permettrait de protéger le réseau SDN contre les attaques de DdS et mitiger leur impact sur la performance du réseau. Notre solution devrait être capable de réduire simultanément la charge de contrôleur, la congestion de la liaison entre le commutateur et le contrôleur et également éviter la surcharge des tables de commutations des commutateurs OpenFlow. Ainsi, nous proposons une solution, nommée SDN-Guard, qui est conçue pour atténuer simultanément les problèmes déjà motionnés en gérant dynamiquement les chemins des flux, le temps d'utilisation des règles de commutation dans les commutateurs et l'agrégation des règles de commutation. Les décisions de SDN-Guard sont prises en se basant sur la probabilité de menace de flux fournie par un système de détection d'intrusion (IDS) qui analyse les flux et détecte les activités malicieuses en temps réel.

De plus, nous investiguons sur l'impact de l'emplacement du l'IDS pour réduire davantage l'utilisation de la bande passante dans le réseau. Ainsi, nous visons à minimiser les trafics (les flux à analyser) entre les commutateurs SDN et l'IDS tout en conservant sa précision de détection des attaques. Pour cela, nous proposons d'utiliser la technique d'échantillonnage de trafic et nous établissons un programme linéaire à nombres entiers (PLNE) qui permet de trou-

ver l'emplacement optimal pour l'IDS et de déterminer les commutateurs SDN qui devraient dupliquer leurs flux vers celui-ci.

Méthodologie du travail

La méthodologie que nous proposons pour répondre aux questions de la problématique et atteindre notre objectif est subdivisée en trois parties. D'abord, nous commençons notre projet avec l'exploration du domaine de recherche en nous basant sur la revue de littérature. Dans cette partie, nous analysons les solutions antérieures qui abordent la même problématique. Ensuite, nous passons à la conception et le développement de la nouvelle technique pour la protection du réseau SDN contre les attaques de DdS en concevant des modules pour la gestion des flux et des règles de commutation et en exploitant un système de détection des intrusions (IDS) pour analyser le trafic. Enfin, nous procédons à la dernière phase de test et de validation du système proposé. Nous étudions d'abord l'impact de l'emplacement de l'IDS sur sa performance ainsi que celui du réseau. Par la suite, nous étudions l'efficacité de la solution proposée pour mitiger l'impact des attaques de DdS sur les réseaux SDN.

Publications

Durant la période du projet, nous avons réalisé deux publications à propos notre travail. Nous avons publié un article nommé «*SDN-Guard : DoS Attacks Mitigation in SDN Networks*» (Dridi, Lobna et Zhani, Mohamed Faten, 2016) qui a été présenté à la conférence «*CloudNet2016*» à PISE (Italie) en Octobre 2016.

Par la suite, nous avons realisé un deuxième article «*A Holistic Approach to Mitigating DoS Attacks in SDN Networks*» qui a été accepté pour publication au journal «International Journal on Networking Technologies - IJNM» en Août 2017, où nous abordons certaines limites de la solution proposée dans le premier article. Ainsi, nous ajoutons une étude sur l'évaluation de la

performance de l'IDS et une investigation sur son emplacement pour minimiser davantage la consommation de la bande passante dans le réseau.

Organisation du rapport

Comme illustré dans le diagramme présenté dans la figure 0.1, le reste de ce mémoire est organisé en quatre chapitres. Le deuxième chapitre présente l'état de l'art qui détaille les notions de base sur la technologie SDN, les attaques de déni de service ainsi que les systèmes de détection d'intrusion (IDS). Ce chapitre donne ensuite un aperçu des travaux antérieurs sur les attaques DdS dans les réseaux SDN ainsi que les solutions proposées afin de mitiger leurs effets sur la performance du réseau. Une dernière section dans ce chapitre présente une étude comparative entre notre solution et les travaux existants.

Le troisième chapitre présente les détails de la solution baptisée, SDN-Guard. Nous décrivons les motivations et les considérations prises pour la conception et le développement d'une nouvelle technique pour la protection du réseau SDN contre les attaques de DdS. Nous y présentons aussi l'architecture et le principe de fonctionnement de SDN-Guard.

Les résultats expérimentaux sont ensuite décrits dans le chapitre quatre. La première partie présente l'implémentation de l'environnement d'expérimentation, une investigation sur la précision et la performance de l'IDS utilisé et son emplacement optimal. La deuxième partie de ce chapitre se concentre sur l'étude de l'efficacité du SDN-Guard. Ce rapport se termine par une conclusion générale sur notre travail de recherche et quelques perspectives pour les futurs travaux.

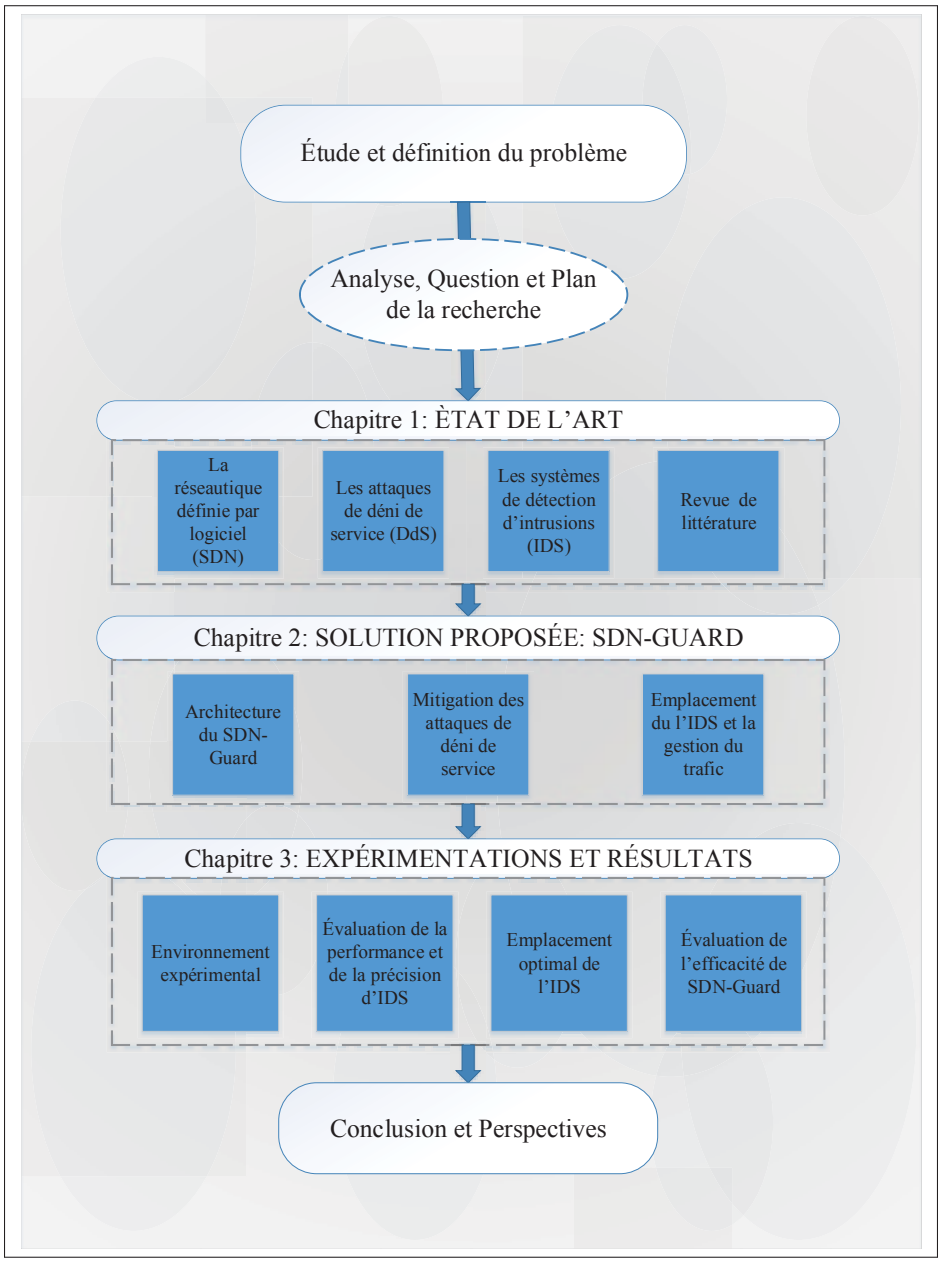

Figure 0.1 Diagramme des chapitres

CHAPITRE 1

ÉTAT DE L'ART

1.1 Introduction

Dans la première partie de ce chapitre, nous allons exposer les notions de base sur la technologie SDN, les attaques de DdS, leur impact sur les réseaux SDN et les systèmes de détection d'intrusions (IDS). La deuxième partie présente une revue de la littérature qui résume les travaux déjà réalisés ainsi que les solutions existantes pour mitiger l'impact des attaques de DdS sur la performance du réseau SDN. Cette partie met aussi la lumière sur les recherches qui visent à garantir la précision à la détection des attaques par l'IDS face à ses ressources physiques limitées (par exemple, la mémoire, la capacité de traitement, etc.). Ces travaux proposent des solutions pour bien gérer le trafic à analyser par l'IDS, durant les attaques de DdS, où la quantité des flux reçus est beaucoup plus grande que sa capacité de traitement.

1.2 La réseautique définie par logiciel – SDN

1.2.1 Définition

L'Open Network Foundation (ONF) définit le réseau SDN comme suit : *«Software-Defined Networking (SDN) is an emerging architecture that is dynamic, manageable, cost-effective, and adaptable, making it ideal for the high-bandwidth, dynamic nature of today's applications. This architecture decouples the network control and forwarding functions enabling the network control to become directly programmable and the underlying infrastructure to be abstracted for applications and network services»* (Open Networking Foundation, 2017a).

La réseautique définie par logiciel (SDN) est un nouveau paradigme qui transforme la façon dont les infrastructures des réseaux informatiques sont gérées, contrôlées et configurées. Le concept du SDN repose sur la séparation du plan de contrôle, qui représente l'intelligence du réseau, et du plan de données qui est en charge du transfert des paquets. Le plan de contrôle est

sous la responsabilité d'un contrôleur centralisé qui possède une vue globale sur l'ensemble du réseau et qui prend en charge toutes les décisions de routage des flux ainsi que toutes les fonctions de gestion et de contrôle au sein du réseau. Cette vue globale du réseau, offerte par SDN, permet de simplifier la gestion pour ses administrateurs (par exemple, le changement de la topologie, le choix des routes pour les flux, la surveillance, la sécurité).

Le plan de données est constitué de commutateurs qui sont responsables de transmettre le trafic dans le réseau SDN, de la source vers la destination, en se basant sur les règles des flux attribuées par le contrôleur. La communication entre ces commutateurs et le contrôleur se fait par un protocole de communication standard comme OpenFlow (OF) (Open Networking Foundation, 2017b), ForCES (Halpern et Hadi, Salim, 2010), NetConf (Enns, Bjorklund, Schoenwaelder et Bierman, 2011), IRS (Clarke, Salgueiro et Pignataro, 2016). Ces protocoles permettent au contrôleur SDN de modifier les tables de commutation des commutateurs avec les règles de commutation nécessaires pour assurer la transmission des flux dans le réseau. En outre, la technologie SDN offre une programmabilité permettant aux administrateurs de configurer, de gérer et de contrôler automatiquement l'ensemble du réseau par des interfaces de programmation applicatives (Application Programming Interface - API) (TechTerms, 2017).

1.2.2 Architecture et caractéristiques de la technologie SDN

1.2.2.1 Architecture

L'architecture du SDN consiste à découpler les tâches de gestion et de contrôle du réseau (le plan de contrôle) et la tâche de transmission des paquets (le plan de données). Comme représenté dans la figure 1.1, le contrôleur communique avec les périphériques physiques du réseau (les commutateurs) à travers un protocole standard de communication appelée OpenFlow (le plus utilisé avec la technologie SDN), alors qu'il utilise des APIs pour interagir avec la couche application. On peut voir dans la figure 1.1 que la structure de SDN est composée de trois couches suivantes :

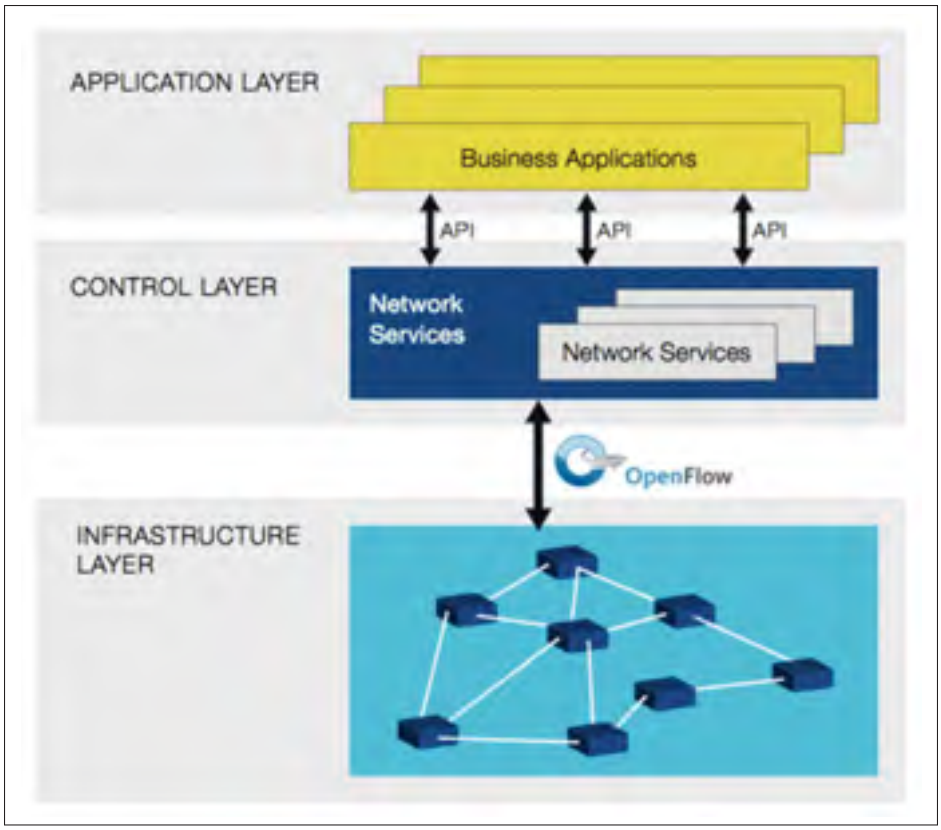

Figure 1.1 Architecture du réseau SDN
Tirée de Open Networking Foundation (2017)

La couche infrastructure

Cette couche désigne l'ensemble des équipements (commutateurs) constituant le réseau. Elle représente le plan de données dans l'architecture SDN. Les commutateurs sont responsables uniquement à la transmission des paquets de la source vers la destination. Ces commutateurs routent les flux en se basant sur des informations contenues dans des tables (par exemple, la FIB ou la CAM/TCAM). Le plan de transmission fournit les fonctions de commutation et de filtrage de paquets.

La couche contrôle

La couche de contrôle est le «cœur» de l'architecture SDN, elle est responsable de la configuration, la gestion et la maintenance du réseau. Comme son nom l'indique, cette couche contrôle

le plan de données en établissant les règles qu'il devra suivre (par exemple, les tables de routage et les tables de commutation). Parmi les protocoles qui participent à ce plan, on peut citer par exemple OSPF, STP, ARP, ou BGP.

Cette couche utilise une entité logique appelée un contrôleur SDN qui est le «cerveau» de réseau. Ce contrôleur est chargé de découvrir et maintenir la topologie, d'informer les périphériques du plan de transmission sur la façon de gérer les paquets dans le réseau, basculer la charge des chemins et communiquer avec les services et les applications offerts par le réseau. Généralement, le contrôleur est centralisé et il vise à assurer que le réseau, dans son ensemble, fonctionne d'une manière optimale en communiquant avec le plan applicatif via des interfaces APIs et le plan de données en utilisant un protocole de communication (OpenFlow, ForCes, NetConf, IRS, etc.).

La couche application

La troisième couche présente le plan applicatif de l'architecture SDN. On trouve dans cette couche des programmes et des services qui réalisent des fonctions spécifiques au contrôleur SDN via les interfaces applicatives API comme REST et FML. Ces applications peuvent construire une vue abstraite du réseau en recueillant des informations à partir du contrôleur à des fins de prise de décision.

Ces applications SDN pourraient inclure la surveillance de réseau, la découverte de la topologie, la réservation des chemins, les statistiques et l'analyse de réseau, la sécurité de réseau, le contrôle d'accès, la virtualisation de fonction de réseau (VNF), etc.

1.2.2.2 Caractéristiques

Le paradigme du SDN s'avère comme une nouvelle ère des réseaux dans la manière de gestion et du contrôle. Selon ONF (Open Networking Foundation, 2017a), l'architecture SDN est :

- **directement programmable :** le contrôle du réseau est directement programmable, car il est dissocié des fonctionnalités de transmission des flux ;

- **agile :** la séparation du contrôle et des fonctionnalités de transmission permet aux administrateurs d'ajuster le trafic de façon dynamique à l'échelle du réseau afin de répondre à l'évolution des besoins ;
- **gérée centralement :** l'intelligence de réseau est (logiquement) centralisée dans des contrôleurs SDN basés sur un logiciel qui maintient une vue globale du réseau, qui apparaît aux applications et aux moteurs de stratégies (policy engines en anglais) comme un seul commutateur logique ;
- **configurée par programmation :** SDN permet aux gestionnaires des réseaux de configurer, de gérer, de sécuriser et d'optimiser très rapidement des ressources réseau par l'intermédiaire des programmes SDN automatisés et dynamiques qu'ils peuvent écrire eux-mêmes, ces programmes ne dépendent pas des logiciels propriétaires ;
- **basée sur des standards ouverts et non liée à un quelconque fournisseur :** lorsqu'il est mis en œuvre grâce à des normes ouvertes, SDN simplifie la conception et l'exploitation du réseau, car les instructions sont fournies par les contrôleurs SDN au lieu de plusieurs périphériques et protocoles spécifiques aux fournisseurs.

En outre, la technologie SDN offre un réseau qui peut dynamiquement s'adapter afin de répondre aux divers besoins des entreprises. Le SDN fournit plusieurs améliorations par rapport au fonctionnement classique. Il offre les avantages suivants :

- **la programmabilité** : la programmabilité permet de répondre aux besoins d'automatisation du réseau. Les administrateurs de réseau peuvent utiliser des APIs afin de programmer les équipements réseau ;
- **la visibilité** : grâce à la vue globale offerte par le réseau, des décisions optimales de routage des flux peuvent être prises facilement à travers le contrôleur SDN plutôt qu'au niveau du l'équipement réseau ;
- **l'orchestration** : la gestion et le contrôle du tout le réseau peut se faire via une seule interface au lieu de configurer chaque périphérique individuellement. La conception de

gestion centralisée simplifie l'administration des infrastructures de grande taille comme le cloud computing par exemple ;

- **la flexibilité** : les administrateurs réseau peuvent facilement changer la taille du réseau en l'augmentant ou en le diminuant par l'ajout ou l'élimination d'un équipement selon sa charge courante. SDN aide aussi les administrateurs à déployer rapidement des nouvelles applications et services pour répondre rapidement à l'évolution des objectifs d'affaires ;
- **la performance** : le réseau SDN optimise l'utilisation des équipements réseau en utilisant l'équilibrage de la charge (load balancing), la gestion de la bande passante, l'optimisation de la capacité, la gestion rapide des pannes (Fast failure handling) ;
- **l'homogénéité** : grâce à la standardisation du protocole de communication OpenFlow, il n'y a plus du problème d'interopérabilité entre les différents matériels réseau utilisés. Par conséquent, les administrateurs du réseau peuvent aisément gérer les flux directement depuis l'interface du contrôleur quelle que soit la marque des composants matériels.

Avec ces avantages, l'approche du SDN optimise l'adéquation entre les ressources réseaux et IT d'une part, et les besoins des entreprises et des affaires de l'autre part.

1.2.3 Protocole de communication OpenFlow

Il existe plusieurs protocoles de communication entre le plan de contrôle et le plan de données tels que ForCES (Halpern et Hadi, Salim, 2010), IRS (Clarke, Salgueiro et Pignataro, 2016) et NetConf (Enns, Bjorklund, Schoenwaelder et Bierman, 2011). Cependant, le protocole de communication OpenFlow reste le plus utilisé sur les équipements du réseau SDN.

OpenFlow est une norme multifournisseur définie par l'ONF qui agit comme un relais entre le contrôleur et les équipements réseau du plan de transmission. Il fournit un protocole ouvert pour programmer les tables de flux sur différents commutateurs et routeurs. C'est un protocole standard utilisé par le contrôleur SDN pour transmettre aux commutateurs des instructions qui permettent de programmer leur plan de données et d'obtenir des informations de ces commutateurs afin que le contrôleur puisse disposer d'une vue globale logique du réseau physique. Cette

vue est utilisée pour toutes les décisions que doit prendre le plan de contrôle (par exemple, routage, filtrage de trafic, partage de la charge, traduction d'adresse) (EFORT, 2016). En effet, le contrôleur peut ajouter, supprimer ou modifier les flux en obtenant des statistiques sur les ports et les flux et d'autres informations à l'aide du protocole OpenFlow.

Le commutateur OpenFlow

Les commutateurs Ethernet et les routeurs les plus modernes contiennent des tables de flux qui sont utilisées pour effectuer des fonctions de transfert selon les couches 2,3 et 4, indiquées dans les entêtes des paquets. Bien que chaque fournisseur ait des tables de flux différentes, il existe un ensemble commun de fonctions pour une large gamme de commutateurs et de routeurs. Cet ensemble commun de fonctions est mis à profit par OpenFlow, protocole entre le contrôleur central OpenFlow et le commutateur OpenFlow et qui, comme indiqué, peut être utilisé pour programmer la logique de transfert ou d'acheminement du commutateur (EFORT, 2016).

On appelle un commutateur OpenFlow (OF) tout équipement réseau prenant en charge le protocole OpenFlow, tel qu'un commutateur, un routeur ou un point d'accès. Chaque périphérique OF maintient une table de flux (autrement dit, une table de commutation ou table d'acheminement), construite à partir de TCAM (Ternary Content Addressable Memory), qui indique le traitement appliqué à tout paquet d'un certain flux (Ali Elamrani Joutei, 2014).

OpenFlow (OF) est censé être un moyen pour tester des nouvelles méthodes de routage ou de transfert pour construire ces tables de flux. La façon dont il l'accomplit est d'établir un tunnel SSH sécurisé entre le commutateur OpenFlow et le contrôleur (figure 1.2). Le contrôleur fonctionnera, quel que soit ce nouveau processus de routage. Lorsqu'un nouveau flux arrive, le commutateur OpenFlow envoie les premiers paquets au contrôleur. Le contrôleur construit alors une entrée dans la table de flux (une règle de flux ou de commutation) pour gérer le reste de cette connexion (Greg Sowell Consulting, 2013).

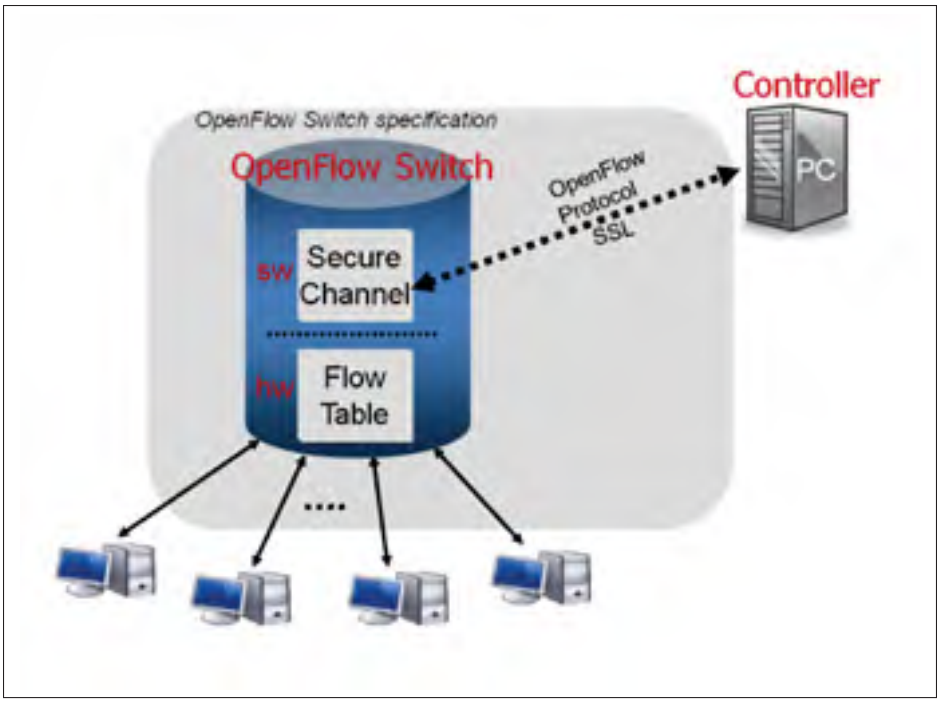

Figure 1.2 Commutateur OpenFlow
Tirée de MediaWiki (2010)

La table de flux

Les tables de flux sont installées sur le TCAM des commutateurs OpenFlow. Ces tables de flux sont mises à jour par le contrôleur en ajoutant et en supprimant les entrées de flux à travers le protocole OpenFlow. Chaque table de flux (ou table de commutation) contient une multitude d'entrées des flux, qui présentent les règles d'acheminement des paquets, associées à des actions pour commander le commutateur, à appliquer certaines actions (transfert, suppression or encapsulation) sur un certain flux (Ali Elamrani Joutei, 2014).

Comme il est illustré dans la figure 1.3, une entrée dans la table de flux est composée de trois champs (MediaWiki, 2010) :

1) le champ «Règle» (ou Rule en anglais) appelé aussi champ d'en-tête : la règle se compose principalement des champs qui se trouvent dans l'en-tête du paquet. Elle constitue les champs de correspondance qui définissent le modèle du flux de paquets à travers l'ins-

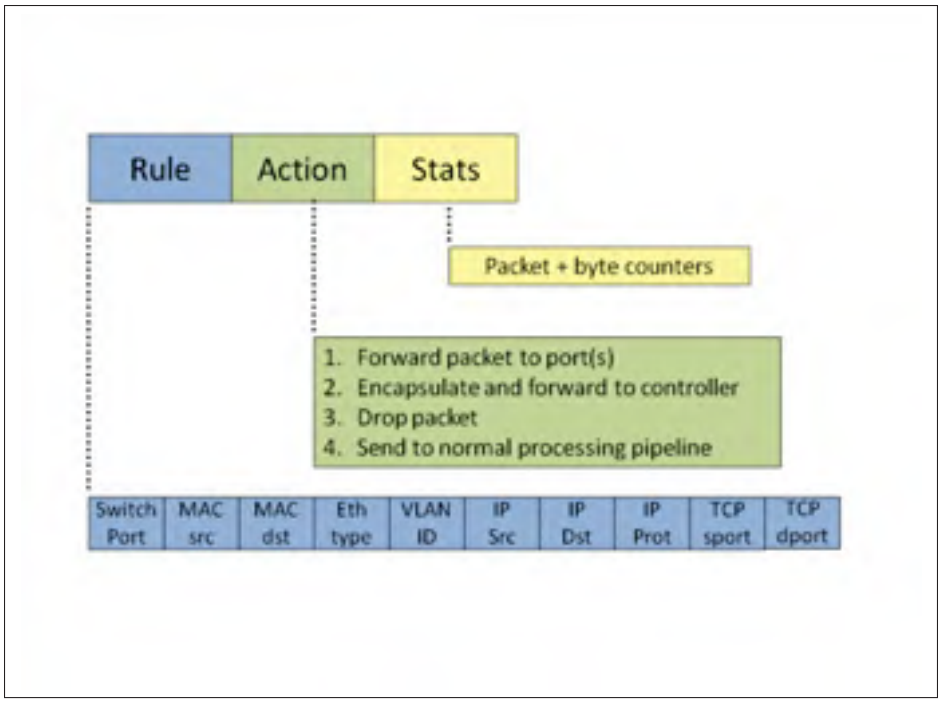

Figure 1.3 Structure d'une entrée du flux
Tirée de MediaWiki (2010)

tanciation des champs d'en-tête allant de la couche Ethernet à la couche Transport. Ce champ est utilisé pour définir la condition de correspondance d'un flux ;

2) le champ «Action» : ce champ détermine l'action à appliquer à un flux spécifique. Il définit comment les paquets doivent être traités. En cas de correspondance, il existe trois actions de base à effectuer : transférer les paquets de ce flux à un port donné, encapsuler et transmettre les paquets de ce flux vers le contrôleur ou supprimer les paquets de ce flux ;

3) les champs «Statistiques» : ils sont utilisés pour compter l'occurrence de la règle à des fins de gestion. Ces statistiques permettent de compter le nombre de paquets et d'octets pour chaque flux, et le temps écoulé depuis le dernier paquet correspondant au flux (pour aider à éliminer les flux inactifs).

Principe de fonctionnement

Comme il est indiqué dans la figure 1.4, le protocole OpenFlow permet au contrôleur d'indiquer au commutateur comment gérer les paquets de données entrants.

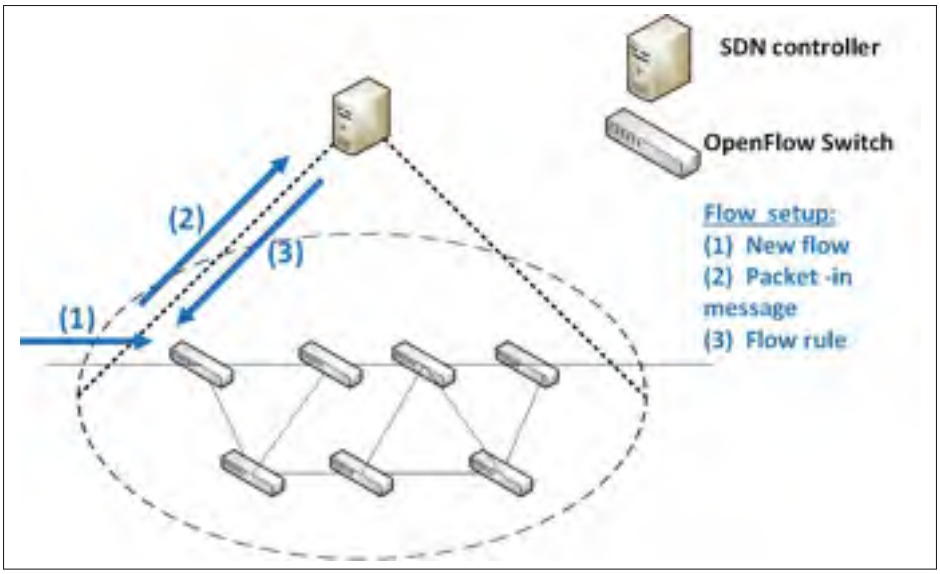

Figure 1.4 Principe de communication du réseau SDN

Dans un réseau SDN basé sur le protocole OpenFlow, lorsqu'un commutateur OpenFlow reçoit un nouveau paquet, il vérifie tout d'abord ses tables de flux pour déterminer le port de sortie du paquet. Si le paquet ne correspond pas à une entrée existante dans la table de flux, le commutateur envoie un message du paquet entrant au contrôleur pour demander une nouvelle règle de commutation afin de l'utiliser pour transférer le paquet vers le prochain nœud. Le contrôleur décide alors le chemin d'acheminement, envoie la règle de flux dans un message au commutateur demandeur et informe tous les commutateurs impliqués des nouvelles règles de commutation à appliquer pour le flux auquel appartient ce paquet.

La figure 1.5 illustre les étapes de l'établissement d'une règle de commutation entre le contrôleur et le commutateur selon les spécifications du protocole OpenFlow.

Lorsque le commutateur reçoit un nouveau paquet «Data packet in» n'ayant pas une règle de flux dans ses tables de commutation, il envoie un message OFPT_PACKET_IN au contrôleur demandant une règle de commutation pour qu'il puisse envoyer le paquet au prochain nœud. Ce message contient l'entête du paquet envoyé qui contient l'adresse IP source, l'adresse IP destination et le numéro de port. Le contrôleur répond alors à la demande du commutateur avec un message OFPT_FLOW_MOD qui contient la règle de flux décidée. Chaque règle de flux

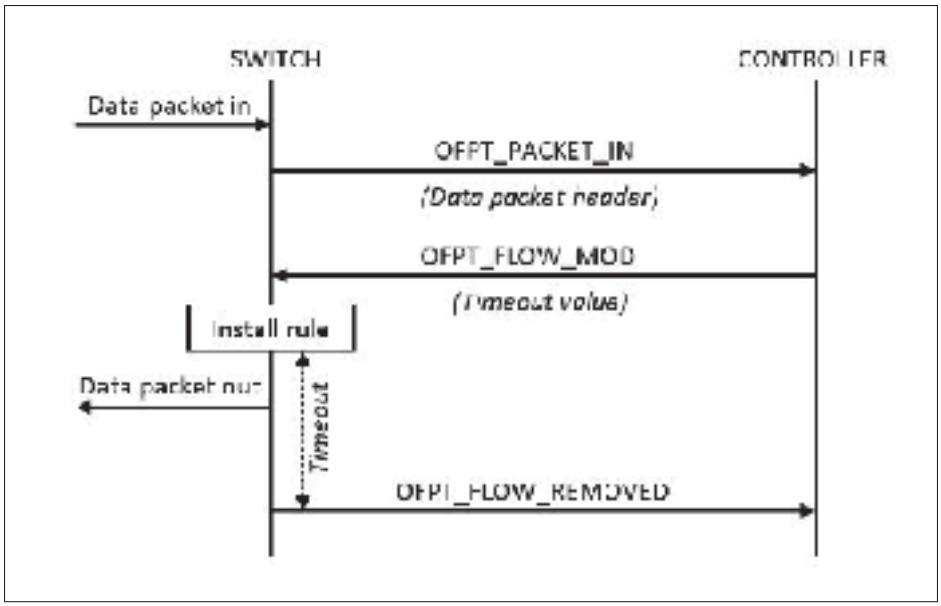

Figure 1.5 Établissement d'une règle de commutation avec OF
Tirée de Kondoi, Rajat et Markku. (2015, p. 1323)

est caractérisée par l'action à effectuer au paquet (transfert, suppression, blocage, etc.), le port de sortie du paquet et la durée pendant laquelle la règle doit être conservée dans sa table de flux appelée «*timeout*».

Chaque règle de flux a deux valeurs de temporisation («*timeout*») associées qui contrôlent la suppression des règles de la table de commutation, le «*idle timeout*» et le «*hard timeout*». Le *idle timeout* est une valeur de temporisation d'inactivité qui indique quand l'entrée doit être supprimée de la table de commutation en raison d'un manque d'activité de cette règle de commutation pendant cette période. Il se déclenche lorsque le flux reste inactif, alors que le *hard timeout* est une valeur qui indique quand l'entrée doit être supprimée, quelle que soit l'activité de la règle de commutation (Open Networking Fundation, 2012). En recevant la nouvelle règle de commutation, le commutateur l'installe dans sa table de commutation et l'utilise jusqu'à l'expiration de l'un de ces temporisateurs. Dans ce cas, le commutateur supprime l'entrée de la règle correspondante de sa table de commutation et envoie un message OFPT_FLOW_REMOVED au contrôleur pour l'informer que la règle a été supprimée.

1.3 Les attaques par déni de service - DdS

1.3.1 Définition

Figure 1.6 L'attaque par déni de service - DdS

L'attaque par déni de service (DdS) est une attaque qui vise à rendre une application informatique, un serveur ou un site web incapable de répondre aux requêtes de ses utilisateurs légitimes (figure 1.6). Dans le cas de DdS, l'attaquant inonde agressivement la cible avec un énorme trafic jusqu'à ce qu'elle devient incapable de servir ses utilisateurs.

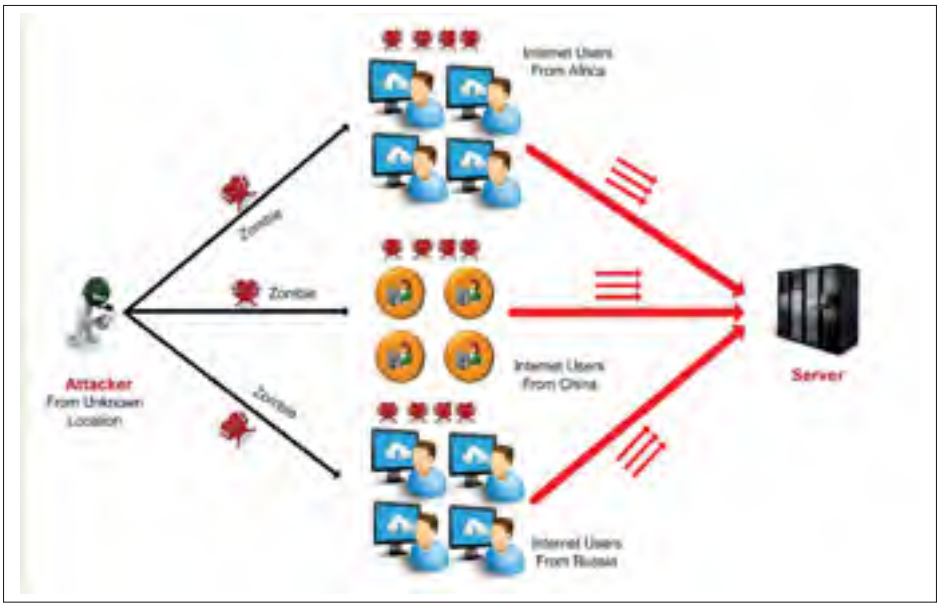

Figure 1.7 L'attaque par déni de service distribué – DDdS
Tirée de BURMABIT (2014)

Il existe aussi des attaques par déni de service distribué (DDdS) dans lesquelle l'attaquant exploite à distance plusieurs ordinateurs zombies infectés, à l'aide d'un virus d'un cheval de Troie ou autres, pour attaquer simultanément la cible (serveur, site web, etc.) (figure 1.7). Ce type d'attaque est très complexe car il est généralement difficile de découvrir la source de l'attaque ou de différencier une requête légitime d'une requête malicieuse (altospam, 2017).

1.3.2 Impacts des attaques de DdS sur le réseau SDN

Due à la centralisation du contrôle dans l'architecture SDN, les attaques de DdS peuvent avoir des graves répercussions sur les performances du réseau conduisant à des cas où tout le réseau devient incapable à répondre aux besoins des utilisateurs. Ces attaques affectent la performance des trois éléments principaux dans le réseau SDN : le contrôleur SDN, la liaison entre le contrôleur et les commutateurs et le plan de transmission (les commutateurs et les liens du réseau).

Impact de l'attaque DdS sur le plan de contrôle

En cas d'une attaque de DdS ou DDdS, l'attaquant va envoyer une grande quantité de flux (simultanément en cas de DDdS) à travers le réseau SDN. Lorsque les commutateurs dans la couche infrastructure reçoivent ces nouveaux flux entrants, ils envoient des demandes au contrôleur pour obtenir des règles de commutation afin de les envoyer à la destination demandée. Par conséquent, le contrôleur SDN sera surchargé à cause de la quantité énorme de demandes provenant du plan de donnée du réseau, menant à des cas où le contrôleur devient totalement paralysé et ne puisse pas prendre aucune décision du routage.

Impact de l'attaque DdS sur le plan de données

Généralement, les commutateurs doivent enregistrer les règles de commutation dans leurs tables de commutation et les utiliser jusqu'à l'expiration des temporisateurs, le *idle timeout* et le *hard timeout*. Dans une situation d'attaque de DdS, où l'attaquant inonde le commutateur avec une quantité du flux importante, tout le trafic de données reçu par les commutateurs, se

traduit en règles de commutation fournit par le contrôleur, afin de les acheminer vers la destination. En effet, la mémoire TCAM du commutateur sera remplie par ces règles envoyées de contrôleur jusqu'à sa saturation. Lorsque cela se produit, les commutateurs sont forcés d'ajouter et de supprimer continuellement les règles de flux et d'envoyer plus des demandes vers le contrôleur ; cela engendre la congestion du plan de transmission ainsi qu'un retard dans le temps de transmission de données.

Impact de l'attaque DdS sur la liaison contrôleur-commutateur

Due à la communication agressive entre le contrôleur SDN et les commutateurs demandant des décisions de routage, la liaison entre le contrôleur et le commutateur (appelé aussi la bande passante du plan de contrôle) sera exténuée et congestionnée ; cela cause la perte de plusieurs messages «*paquet-in*» ainsi que le retard dans le temps de réponse des messages échangés entre le contrôleur et les commutateurs.

1.4 Les systèmes de détection d'intrusions - IDS

Dans cette section, nous allons définir les systèmes de détection des intrusions (IDS) ainsi que ses différents types. De plus, nous allons présenter les deux approches de détection d'intrusion utilisées par les IDSs.

Un système de détection des intrusions (*Intrusion Detection System – IDS*) est un outil capable de surveiller en temps réel les paquets qui transitent dans le réseau afin de détecter ceux qui semblent suspects.

L'objectif principal de l'IDS est de protéger le réseau contre les intrusions et d'assurer sa sécurité. Il se trouve plusieurs IDS open source comme Snort (The Snort Team, 2017), Bro (The Bro Network Security Monitor, 2017) et Suricata (Suricata, 2015). L'utilisation de l'IDS dans les réseaux permet de collecter des informations sur les intrusions, d'assurer une gestion centralisée des alertes et d'effectuer un premier diagnostic sur la nature de l'attaque, permettant

ainsi de réagir activement et rapidement à l'attaque pour la ralentir ou la stopper (Guillaume Lehmann, 2003).

Il existe une possibilité d'erreur de détection de l'IDS qui varie du système à un autre. On parle ici de fausses alarmes positives et de fausses alarmes négatives. Une alerte est dite fausse positive si l'IDS détecte une activité légitime comme une activité malveillante alors qu'une alerte est dite fausse négative si l'IDS accepte une activité malveillante comme une activité légitime.

1.4.1 Types des systèmes de détection d'intrusions

Les systèmes de détection d'intrusions peuvent se classer en deux catégories majeures selon qu'ils s'attachent à surveiller (Guillaume Lehmann, 2003) : soit le trafic réseau (on parle d'IDS réseau ou NIDS (Network based IDS)), soit l'activité des machines (on parle d'IDS Système ou HIDS (Host based IDS)).

- **les systèmes de détection d'intrusions de type réseau (Network IDS - NIDS) :** un NIDS écoute tout le trafic réseau, l'analyse et génère des alertes s'il détecte des paquets semblent malveillants. La plupart des NIDS sont aussi dits IDS *inline* car ils analysent le flux en temps réel. Pour cette raison, la question des performances est très importante. De tels IDS doivent être de plus en plus performants afin d'analyser les volumes de données de plus en plus importants pouvant transiter sur les réseaux (Guillaume Lehmann, 2003).

- **les systèmes de détection d'intrusions de type hôte (Host IDS - HIDS) :** un HIDS analyse en temps réel les flux relatifs à une machine sur laquelle il est installé ainsi que les journaux des événements et génère des alertes s'il détecte des activités semblent malveillants. Ce type d'IDS est très dépendant du système sur lequel il est installé. Il faut donc des outils spécifiques en fonction des systèmes déployés. Ces IDS peuvent s'appuyer sur des fonctionnalités d'audit propres ou non au système d'exploitation, pour en vérifier l'intégrité, et générer des alertes. Il faut cependant noter qu'ils sont incapables de détecter les attaques

exploitant les faiblesses de la pile IP du système, typiquement les dénis de service comme *SYN-FLOOD* ou autre (Guillaume Lehmann, 2003).

- **les systèmes de détection d'intrusions de type hybrides (NIDS+HIDS) :** un IDS hybride rassemble les caractéristiques des NIDS et HIDS. Il permet, en un seul outil, de surveiller les réseaux et les terminaux. Les sondes sont placées en des points stratégiques, et agissent comme NIDS et/ou HIDS suivant leurs emplacements. Toutes ces sondes remontent alors les alertes à une machine qui va centraliser le tout, et agréger les informations d'origines multiples (Guillaume Lehmann, 2003).

1.4.2 Approches de détection des intrusions

Comme le décrivent Thongkanchorn et al. (Thongkanchorn, Kittikhun, Ngamsuriyaroj, Sudsanguan et Visoottiviseth, Vasaka, 2013), les modules de traitement principaux d'un IDS consistent à un module pour capturer des paquets de réseau, un module pour décoder le paquet afin de classifier les champs de ce dernier et un autre module pour détecter si le paquet est légitime ou malveillant basé sur les règles appliquées. L'IDS examine les paquets du trafic via un ensemble des règles. Si la charge utile du paquet correspond à l'une des règles, l'IDS génère des alertes. Par conséquent, tout trafic déviant des caractéristiques normales est supposé malveillant.

Les IDSs disposent de deux approches différentes afin de déceler les intrusions (Cikala, Frédéric, 2017) :

- **IDS à signature :** généralement, les IDSs réseau se basent sur un ensemble des signatures qui représentent chacune le profil d'une attaque. Cette approche consiste à rechercher dans l'activité de l'élément surveillé (un flux réseau) les empreintes d'attaques connues, à l'instar de l'antivirus. Une signature est habituellement définie comme une séquence d'événements et des conditions relatant une tentative d'intrusion. La reconnaissance est alors basée sur le concept de «correspondance des modèles» (analyse des chaînes de caractères présentent dans le paquet à la recherche de correspondance au sein d'une base de connais-

sance). Si une attaque est détectée, une alarme peut être remontée si l'IDS est en mode actif, sinon, il se contente d'archiver l'attaque ;

- **IDS comportemental :** les IDSs comportementaux ont pour principale fonction la détection d'anomalie. En effet, leur déploiement nécessite une phase d'apprentissage pendant laquelle l'outil va apprendre le comportement «normal» des flux applicatifs présents sur son réseau. Ainsi, chaque flux et son comportement habituel doivent être déclarés. L'IDS se chargera d'émettre une alarme, si un flux anormal est détecté. Les IDSs comportementaux sont apparus bien plus tard que les IDSs à signature et ne sont pas encore assez matures. Ainsi, l'utilisation de tels IDS peut s'avérer délicate dans le sens où les alarmes remontées contiendront une quantité importante de fausses alertes. Ce problème peut être résolu en généralisant la déclaration des flux.

1.4.3 Exemples des systèmes de détection d'intrusion

Les systèmes de détection d'intrusion sont indispensables pour la sécurité du réseau. Ils permettent de détecter les tentatives d'intrusions en se basant sur la base d'une signature ou l'apprentissage d'un comportement pour les flux présents dans le réseau. Il existe plusieurs IDS open source qu'on peut les utiliser pour la détection et la prévention contre tout type d'attaque comme Snort (The Snort Team, 2017), Bro (The Bro Network Security Monitor, 2017) et Suricata (Suricata, 2015).

Snort

Snort est un système de détection et de prévention des intrusions (NIDS/NIPS), créé par Martin Roesch en 1998. Ce système s'est imposé comme le système de détection d'intrusions le plus utilisé. Sa version commerciale lui a donné une bonne réputation auprès des entreprises (Jonathan Krier, 2006).

Snort est capable d'effectuer une analyse du trafic réseau en temps réel et il est doté de différentes technologies de détection d'intrusions telles que l'analyse protocolaire et le «correspon-

dance des modèles». Il peut détecter de nombreux types d'attaques comme le déni de service, le débordement de tableau, le scan de ports furtifs et les tentatives de «fingerprinting» de système d'exploitation. Snort se base d'un langage de règles permettant de décrire le trafic qui doit être accepté ou collecté. De plus, son moteur de détection utilise une architecture modulaire de «plug-ins» (Jonathan Krier, 2006).

Snort peut fonctionner en trois modes :

- **sniffer de paquets :** dans ce mode, Snort observe les paquets qui circulent dans le réseau et les affiche d'une façon continue dans l'écran comme «*tcpdump*» ;
- **logger de paquets :** dans ce mode, Snort enregistre les paquets qui circulent sur le réseau dans des répertoires sur le disque. On peut utiliser ce mode pour limiter les logs avec certains critères comme une plage d'adresse IP ou un protocole ;
- **NIDS :** dans ce mode, Snort analyse le trafic du réseau, compare ce trafic à des règles déjà définies par l'administrateur du réseau et établit des actions à exécuter.

Dans ce travail, nous nous intéresserons qu'au mode NIDS.

Bro

Bro est un système de détection d'intrusion réseau (NIDS) open source basé sur UNIX et qui a été fondé par Vern Paxson en 1998. Bro surveille passivement le trafic réseau afin de trouver des flux malveillants. Il détecte les intrusions en analysant d'abord le trafic réseau, puis il exécute des analyseurs orientés évènements qui comparent l'activité avec des motifs jugés malveillants. L'analyse du trafic suspect comprend la détection des attaques spécifiques (signature et évènements) et des activités inhabituelles (anormales). Ce NIDS est capable de donner une analyse détaillée aux événements qui décrivent l'activité observée (Rodfoss, Jonas Tafto, 2011).

Bro contient un ensemble de scripts de politique, qui sont conçus pour détecter les attaques internet les plus courantes, tout en limitant le nombre de faux positifs. Ces scripts de politique sont des programmes écrits dans le langage Bro. Ils contiennent des règles qui décrivent le type

du trafic ou les activités qui sont considérés comme malveillants. Lors de l'analyse de l'activité dans le réseau, Bro lance les actions à exécuter en fonction de l'analyse (Rodfoss, Jonas Tafto, 2011).

Suricata

Suricata est un système de détection et de prévention des intrusions qui est open source et qui a été développé par l'«*Open Information Security Foundation* - OISF» (Open Information Security Foundation, 2017). La version bêta a été diffusée en décembre 2009 alors que la première version stable a eu lieu en juillet 2010. Suricata a été crée pour apporter de nouvelles idées et technologies au domaine de détection d'intrusion. L'OISF fournit à Suricata un ensemble de règles de détection et de prévention des intrusions. Suricata est capable d'utiliser des règles à partir d'autres systèmes tels que Snort pour fournir le meilleur ensemble des règles possible (Rodfoss, Jonas Tafto, 2011).

Comme les autres systèmes de détection d'intrusion réseau, Suricata surveille le trafic réseau et crée des alertes et des journaux d'évènements lorsque le trafic malveillant est détecté (Rodfoss, Jonas Tafto, 2011).

1.5 Revue de littérature

Dans cette section, nous nous concentrons principalement sur les efforts de recherche qui ont abordé les attaques DdS dans les réseaux SDN afin de mitiger leur impact sur la performance du réseau. D'abord, nous exposons les différents techniques de mitigation, qui sont élaborés dans des diverses études. Ensuite, nous avons un recours à des solutions qui exploitent les IDSs dans leurs stratégies de mitigation contre les attaques de DdS telles qu'on a utilisées dans la nôtre. De plus, nous présentons des techniques d'échantillonnage de trafic proposées pour minimiser davantage le trafic analysé dans les réseaux, surtout dans les cas des attaques de DdS. Nous terminons cette section avec une étude comparative qui met en comparaison notre solution proposée, SDN-Guard, à ceux qui existent déjà en fonction de leurs objectifs ciblés à minimiser lors d'attaques DdS.

1.5.1 Techniques de mitigation sans l'utilisation de l'IDS

FlowRanger

Lei Wei et Carol Fung (Wei, Lei et Fung, Carol, 2015) proposent FlowRanger, un système de rangement de flux qui permet de détecter et d'atténuer l'impact des attaques de DdS. FlowRanger est implémenté dans le contrôleur SDN. Ce système est constitué de trois composants principaux. Le premier est un composant de gestion de la valeur de confiance qui calcule la valeur de confiance pour chaque message «packet-in» en vérifiant sa source. Le deuxième est responsable à la gestion de la file d'attente du contrôleur, qui va attribuer des règles de commutation à chaque paquet entrant. Ce composant est en charge de mettre les messages «packet-in» dans file d'attente en priorité selon sa valeur de confiance déjà calculée par le premier composant. Et le troisième représente le composant d'ordonnancement de message qui traite les messages selon la stratégie «Weighted Round Robin» (WRR) (Brocade Communications Systems, 2016). La technique FlowRanger peut réduire l'impact des attaques DdS sur les performances du réseau en garantissant que les flux légitimes sont servis en premier dans le contrôleur. Cependant, cette solution n'empêche pas la surcharge du contrôleur et la surcharge des tables de commutation des commutateurs.

Technique de filtrage des adresses IP

Une autre solution a été proposée dans (Dao, Nhu-Ngoc, Park, Junho, Park, Minho et Cho, Sungrae, 2015) pour protéger les réseaux SDN contre les attaques DDdS et qui est basée sur la technique de filtrage IP. Le mécanisme consiste à analyser le comportement et les activités de l'utilisateur dans le réseau afin de calculer le nombre des tentatives de connexion du son adresse IP. Les auteurs définissent une table de stockage dans le contrôleur pour stocker les adresses IP source du chaque paquet transmis par les commutateurs demandant une règle de commutation. Ensuite, le contrôleur incrémente un compteur qui calcule le nombre des paquets arrivés au contrôleur de la même adresse IP. Enfin, le contrôleur va générer l'expiration de la règle de commutation dans le commutateur en se basant sur le nombre de connexions de l'adresse IP. Si le nombre des tentatives de la connexion dépasse le seuil d'une utilisation normale, les

paquets provenant de cette adresse IP sont considérés comme malicieux et le contrôleur va attribuer un court *timeout* aux flux des utilisateurs malveillants. Par contre, si le nombre de connexions ne dépasse pas le seuil défini, l'utilisateur de l'adresse IP est considéré comme légitime et un long *timeout* est assigné à sa règle de commutation. Cette solution force les règles de commutation du trafic malveillant à être rapidement supprimées des tables des flux des commutateurs. Cependant, cela peut conduire à l'envoi de nouveaux messages «packet-in» au contrôleur, demandant des nouvelles règles de commutations pour le même flux, si la durée d'utilisation de flux est supérieure au *timeout* défini dans la règle de commutation. Par conséquent, ceci engendre la surcharge du contrôleur. En outre, cette solution bloque tout le trafic malveillant, ce qui peut être critique pour les flux faux positifs.

Technique de gestion autonome

Dans cette solution (Sahay, Rishikesh, Blanc, Gregory, Zhang, Zonghua et Debar, Hervé, 2015), les auteurs exploitent la centralisation et la programmabilité offertes par la technologie SDN et proposent un système de gestion autonome dans lequel le fournisseur d'accès internet (FAI) et ses clients collaborent pour atténuer l'impact des attaques DdS. Dans cette solution, le FAI collecte des informations sur les menaces déclarées par les clients afin de les utiliser pour appliquer les politiques de sécurité et mettre à jour les tables de flux des commutateurs dans le réseau. Si un flux est considéré comme légitime par les clients, le contrôleur du FAI le marque avec une priorité élevée de sorte qu'il prend un chemin de haute qualité qui assure la rapidité du l'envoi et garantit sa réception. Cependant, en cas de doute sur la légitimité du flux, le contrôleur du FAI lui affectera une faible priorité et le dirigera vers le chemin désigné pour les flux malveillants. Cette proposition réduit l'impact de l'attaque DdS sur la performance du réseau au niveau de plan de transmission en équilibrant la charge sur différents chemins. Cependant, elle n'atténue pas les risques de surcharge du contrôleur et la saturation des tables de commutation des commutateurs à cause du grand nombre de flux reçus par les commutateurs et qui génèrent, à la suite, un nombre élevé des règles de commutation dans leurs tables TCAM ainsi qu'une communication agressive entre ces commutateurs et le contrôleur pour attribuer les règles des commutations nécessaires.

FloodGuard

FloodGuard est une nouvelle plateforme de défense contre les attaques DdS dans les réseaux SDN proposée par Wang et al. (Wang, Haopei, Xu, Lei et Gu, Guofei, 2015). L'objectif des auteurs est d'éviter la saturation des tables de commutation des commutateurs du plan de transmission ainsi que la surcharge du contrôleur. Pour atteindre ces objectifs, ils implémentent deux modules de supervision et de gestion des flux dans le contrôleur. Le premier est un module d'analyse proactive des règles de commutation qui génère et installe les règles de commutation durant l'attaque de DdS. Le deuxième est un module de migration de paquet qui gère le trafic inondé sans perdre le trafic légitime. Lorsqu'une attaque de DdS est détectée, le module de migration des paquets commence à les rediriger vers un cache installé dans le plan de données. Par la suite, le module d'analyse proactive des règles de commutation génèrera les règles de commutation suivant l'état actuel telles que la quantité de données dans le réseau et l'état des liens. Dans le même temps, le cache du plan de données commence à gérer les paquets déjà collectés par l'envoi des messages «packet_in» au contrôleur avec un taux d'envoi limité en utilisant l'algorithme du tourniquet (Round Robin - RR) comme technique d'ordonnancement des paquets (Luc, De Mey, 2017). Lorsque les règles de flux sont prêtes, le contrôleur les attribue aux commutateurs. Les résultats prouvent que cette technique est très efficace pour diminuer le taux d'utilisation de la bande passante entre le contrôleur et le commutateur et il évite la saturation de sa mémoire TCAM. Cependant, FloodGuard n'évite pas la surcharge du plan de contrôle parce qu'elle va augmenter la charge du traitement au niveau du contrôleur.

AVANT-GUARD

AVANT-GUARD (Shin, Seungwon, Yegneswaran, Vinod, Porras, Phillip et Gu, Guofei, 2013) est une nouvelle plateforme qui permet au plan de contrôle d'être plus résistant et évolutif contre les attaques qui engendrent la saturation du plan de contrôle comme l'attaque des inondations TCP-SYN (DdS). Dans ce travail, le but des auteurs est d'éviter la saturation de la bande passante de commutateur au contrôleur et d'augmenter le taux de réponse du contrôleur aux demandes arrivant des commutateurs du plan de données. Pour atteindre cet objectif,

Shin et al. mettent en place deux nouveaux modules dans le plan de données ; le «*connection migration*» et l'«*actuating trigger*». Le premier module protège le contrôleur de la saturation lors des attaques d'inondation TCP-SYN en limitant les demandes de flux envoyées au plan de contrôle. Il inspecte les sessions TCP au plan de données avant de les envoyer au contrôleur. Le deuxième module, appelé «*actuating trigger*», permet d'augmenter la réactivité entre le contrôleur et le plan de données (les commutateurs OF). Il utilise des déclencheurs pour collecter les statistiques du réseau et signaler toutes les conditions existantes dans les commutateurs (la capacité de stockage des tables TCAM, le nombre des règles des flux installées, la charge dans les files d'attente, etc.) au contrôleur afin de répondre aux menaces. Les résultats montrent que les extensions de sécurité AVANT-GUARD réduisent la charge dans le plan de contrôle et l'utilisation de la table des règles de flux du commutateur dans le plan de données. Cependant, seuls les flux TCP sont considérés, l'approche proposée n'est pas conçue pour empêcher les attaques basées sur des autres protocoles tels que UDP ou ICMP. De plus, pour assurer une sécurité stricte, les paquets du flux interagissent avec le contrôleur seulement après avoir passé le mécanisme d'établissement d'une connexion TCP. Cette méthode peut aider à détecter les flux malveillants, mais elle ne tient pas compte de la charge de traitement dans le plan de données lors de l'attaque de DdS. Cette solution peut augmenter le délai de réponse pour les flux légitimes. En outre, pour implémenter cette solution, il est nécessaire d'effectuer des modifications à la spécification du commutateur OpenFlow afin de gérer les opérations des nouveaux modules.

SLICOTS

SLICOTS est une autre solution, proposée par Mohammadi et al. (Mohammadi, Reza, Javidan, Reza et Conti, Mauro, 2017), pour atténuer l'impact des attaques d'inondation TCP-SYN dans les réseaux SDN. Elle se présente comme un module de sécurité implanté dans le plan de contrôle. Ce module surveille toutes les requêtes TCP entrantes au réseau afin de détecter et prévenir les attaques par inondation TCP-SYN et bloquer les hôtes malveillants. Pour chaque demande de connexion TCP, SLICOTS installe des règles de flux temporaires pendant le processus de connexion TCP, et après la validation d'une connexion, il installe des règles de flux

permanentes entre le client et le serveur. De plus, SLICOTS bloque l'attaquant qui envoie un grand nombre de connexions TCP à demi ouvert (*half-Open*). La solution proposée est mise en œuvre en tant que module d'extension au contrôleur OpenDaylight (Linux foundation, 2017). Les résultats des simulations ont montré que SLICOTS est une solution efficace pour atténuer l'impact des attaques d'inondation SYN en réduisant le temps de détection d'attaque, en minimisant la taille de la table de flux et la charge du contrôleur. Cette solution offre un temps de réponse faible pour les demandes légitimes. Cependant, lors d'une attaque DDdS, le temps passé avant de détecter le trafic malveillant et de le bloquer, peut surcharger le contrôleur et le plan de données. De plus, durant la période de détection, les tables TCAM des commutateurs peuvent être saturées vu le grand nombre des règles de flux temporaires installées.

Combinaison OpenFlow et sFlow

K.Giotis et al. proposent dans (Giotis, Kostas, Argyropoulos, Christos, Androulidakis, Georgios, Kalogeras, Dimitrios et Maglaris, Vasilis, 2014), une méthode combinant les spécifications du protocole OpenFlow et la technologie standard de la surveillance du trafic sFlow (sFlow.org, 2017). Cette méthode présente un mécanisme efficace de détection des anomalies et d'atténuation de leurs impacts sur les environnements SDN en temps réel. La solution est constituée de trois modules principaux. Un module collecteur collecte périodiquement les informations et les statistiques des flux installées dans les tables de commutations des commutateurs. Pour accomplir cette tâche, les auteurs choisissent d'utiliser sFLow qui est un mécanisme de surveillance de flux tout en utilisant l'échantillonnage de paquets. Ce module envoie périodiquement (chaque 30 secondes) les paquets échantillonnés des flux collectés vers un module de détection d'anomalie afin d'identifier les attaques potentielles en se basant sur les informations des flux. Lorsqu'une anomalie est détectée, ce module de détection communique avec un module de mitigation qui est en charge de prendre les contre-mesures nécessaires pour atténuer les effets des attaques sur le réseau SDN. Ce module distribue et installe les règles de commutation appropriées dans chaque commutateur dans le réseau afin de neutraliser les attaques identifiées en bloquant le trafic malveillant. Les résultats ont démontré que la solution proposée détecte avec succès les attaques de DDdS, la propagation des vers et les attaques de

scan de port. Grâce à la capacité d'échantillonnage des paquets du sFlow, cette solution réduit la communication requise entre les commutateurs et le contrôleur SDN ainsi que la charge du traitement au niveau du contrôleur. Cependant, la collecte périodique des données à partir de l'ensemble de la table de flux qui peut contenir des milliers d'entrées ne s'arrange pas pour des environnements à trafic réseau élevés. De plus cette solution ne résout pas le problème de l'épuisement des tables de commutation des commutateurs durant l'attaque. En outre, cette technique ne prend pas en considération la possibilité des alertes fausses positives, puisqu'il bloque directement le trafic qui est considéré comme malveillant.

1.5.2 Techniques de mitigation avec l'utilisation de l'IDS

En outre, notre solution exploite un système de détection des intrusions (IDS) pour analyser le trafic et détecter les attaques de DdS dans le réseau SDN. Dans cette section, nous exposons quelques travaux de recherche qui proposent des stratégies de mitigation qui utilisent des IDSs pour l'analyse du trafic et la détection des attaques de DdS. Ces travaux se basent sur les alertes fournies par un IDS afin de bien gérer le trafic et atténuer l'impact des attaques de DdS sur la performance du réseau SDN. Ces propositions s'appuient sur l'utilisation d'un IDS centralisé, c'est-à-dire un IDS unique qui analyse le trafic de l'ensemble du réseau, ou un IDS distribué où plusieurs IDS sont répartis sur le réseau et envoient des alertes à un serveur centralisé pour une analyse plus approfondie. Nous présentons ces recherches qui utilisent le système IDS pour analyser le trafic de l'ensemble du réseau. Ainsi, nous abordons des solutions proposées pour réduire la quantité de trafic analysé en cas de l'utilisation d'IDS unique afin de ne pas dépasser sa capacité de traitement.

BroFlow

Le mécanisme BroFlow présenté dans (Lopez, Martin Andreoni, Mattos, Diogo Menezes Ferrazani et Duarte, Otto Carlos MB, 2016), est un système de détection et de prévention des intrusions (IDPS) distribué pour les environnements SDN. Ce système utilise l'IDS Bro (The Bro Network Security Monitor, 2017) pour surveiller et analyser les paquets en réseau par

des capteurs implémentés dans les commutateurs réseau. L'objectif des auteurs est l'optimisation du placement des capteurs Bro en minimisant leur nombre et maximisant la couverture réseau pour chaque capteur. Deux types de capteur Bro sont déployés dans l'architecture, les capteurs de réseau virtuel BroFlow déployés en réseau virtuel et le capteur d'infrastructure BroFlow installé dans une machine physique. Le rôle de ces capteurs est la surveillance du réseau, la détection des attaques et l'alerte du contrôleur. Chaque capteur BroFlow compte les paquets pendant une période donnée. Lorsqu'un seuil est atteint, il envoie une alarme au contrôleur. De plus, une application BroFlow, a été développée et hébergée dans un contrôleur POX (Confluence, 2015) pour recevoir les messages d'alarme provenant de ces capteurs, vérifier la correspondance entre les informations du message avec ses tables de flux et indiquer au contrôleur la décision appropriée à prendre dans tous les commutateurs de réseau (par exemple, supprimer un flux ou modifier le port de sortie). Par ailleurs, afin d'équilibrer la charge entre les commutateurs en réseau, les auteurs utilisent la technique de l'élasticité qui consiste à activer ou désactiver des machines virtuelles en fonction de la charge du réseau. Les résultats prouvent que l'algorithme proposé couvre 95% du réseau avec seulement 7 capteurs. De plus, cette solution réduit le temps nécessaire pour détecter et mitiger l'attaque jusqu'à 90% par rapport au cas sans BroFlow. Malgré ces résultats positifs, cette solution ne prend pas en compte le cas de la défaillance d'un capteur BroFlow qui entraîne la surcharge des autres capteurs.

Approche collaborative

Le but de cette approche est d'assurer une alerte rapide contre les menaces potentielles avec une grande précision, Tommy Chin et al. (Chin, Tommy, Mountrouidou, Xenia, Li, Xiangyang et Xiong, Kaiqi, 2015). Les auteurs fournissent une nouvelle technique de détection de l'attaque. Cette technique consiste à la coordination entre les moniteurs (IDS) répartis sur les réseaux, un corrélateur, qui est le contrôleur centralisé, et les commutateurs. Avec cette solution, quand l'IDS détecte des anomalies dans le trafic, il envoie une alerte au corrélateur (contrôleur SDN) pour vérifier les suspects en examinant d'une façon plus détaillée les paquets réseau pour trouver des signatures d'attaque supplémentaires, prendre la décision appropriée et mettre à jour la table des flux de commutateurs pour atténuer l'impact d'une attaque. Les

résultats des simulations montrent que la solution évolue bien au fur et à mesure que le trafic de l'utilisateur augmente, mais la duplication du trafic de commutateur vers l'IDS nécessite beaucoup du temps ainsi que celle prise par le contrôleur pour envoyer la décision du routage (la règle de commutation) au commutateur. En outre, cette solution ne tient pas compte de la consommation de capacité de traitement des moniteurs, des commutateurs et du corrélateur.

SnortFlow

SnortFlow est un système proposé dans (Xing, Tianyi, Huang, Dijiang, Xu, Le, Chung, Chun-Jen et Khatkar, Pankaj, 2013) et qui est dédié à la prévention des environnements virtuels tels que les nuages informatiques contre les intrusions détectées. Ce système vise à détecter les intrusions, à sélectionner des contre-mesures appropriées et à reconfigurer les réseaux de nuage pour atténuer ou prévenir les attaques. Pour atteindre cet objectif, Xing et al. combinent le mode de fonctionnement du Snort comme un IDS réseau (NIDS) avec la puissance de la programmabilité offerte par la technologie SDN pour déployer SnortFlow. Cette solution est basée sur le composant serveur SnortFlow qui collecte les alertes provenant des agents Snort. Ces agents sont installés dans le réseau afin d'analyser le trafic suspect détecté et envoyer les alertes au serveur SnortFlow. À la réception des alertes, le serveur SnortFlow les analyse, génère les règles des flux appropriées pour le trafic suspect et transmet ces règles au contrôleur OpenFlow. Ce dernier reconfigure le réseau en fonction des nouvelles règles générées. L'évaluation de SnortFlow a présenté une bonne performance en termes d'analyse du trafic et de prévention contre l'intrusion. Malgré cela, avec SnortFlow, la détection de l'ensemble du trafic généré lors d'une attaque DDdS peut être difficile pour l'agent Snort à cause de ses ressources limitées (capacité de traitement, mémoire, etc.) ce qui minimise sa précision de la détection. De plus, la consommation de la bande passante entre l'agent SnortFlow et le serveur SnortFlow peut engendrer un nombre élevé des alertes durant ce type d'attaque.

1.5.3 Techniques d'échantillonnage de trafic

L'attaque de DdS ainsi que le DDdS, visent à perturber, ou paralyser totalement, le fonctionnement du réseau SDN en le bombardant avec des requêtes erronées. Ce type d'attaque peut affecter la performance du réseau en saturant ses ressources comme la bande passante, l'espace de stockage dans les tables TCAM du commutateur et la capacité de traitement du contrôleur. Dans le contexte du notre travail, nous exploitons un IDS pour analyser le trafic dans le réseau et détecter s'il est malveillant ou pas, en se basant sur son comportement. Cette tâche serait très difficile à l'IDS durant une attaque de DdS ou de DDdS, vu que la quantité de paquets du trafic à analyser est beaucoup plus grande de sa capacité de traitement. Alors, pour minimiser la charge de l'IDS, nous avons recours à l'échantillonnage de trafic analysé par l'IDS.

Plusieurs recherches ont étudié la précision de détection des anomalies dans le réseau en utilisant le processus d'échantillonnage du trafic (Androulidakis, Georgios et Papavassiliou, Symeon, 2008), (Kawahara, Ryoichi, Ishibashi, Keisuke, Mori, Tatsuya, Kamiyama, Noriaki, Harada, Shigeaki et Asano, Shoichiro, 2007), (Ha, Taejin, Kim, Sunghwan, An, Namwon, Narantuya, Jargalsaikhan, Jeong, Chiwook, Kim, JongWon et Lim, Hyuk, 2016). Ce processus consiste à faire des observations partielles des flux provenant au réseau et à tirer des conclusions sur le comportement complet de ces flux à partir de ces observations limitées. Cette technique est très utilisée dans les cas où le réseau est menacé par une attaque de DdS dont lequel les nombres des flux reçus est beaucoup plus grand que la capacité de traitement du réseau, ce qui engendre des difficultés dans le stockage et traitement de ces flux.

L'objectif de toutes ces études est de réduire le volume de données collectées tout en gardant la précision d'observation et de détection des anomalies afin de rendre ce processus réalisable et évolutif. Par exemple, Androulidakis et Papavassilou dans (Androulidakis, Georgios et Papavassiliou, Symeon, 2008) ont proposé une technique d'échantillonnage du flux sélectif qui permet d'analyser seulement des petits flux, car ils supposent que ces derniers sont prédominants dans le trafic malveillant. Une autre solution d'échantillonnage de trafic à analyser par le réseau a été proposée par Kawahara et al. (Kawahara, Ryoichi, Ishibashi, Keisuke, Mori,

Tatsuya, Kamiyama, Noriaki, Harada, Shigeaki et Asano, Shoichiro, 2007) qui consiste de partitionner le trafic en groupes en fonction des adresses IP sources des flux. Par la suite, les auteurs analysent chaque groupe individuellement pour obtenir des statistiques de flux afin de détecter les anomalies dans le trafic.

Un autre travail est présenté dans (Ha, Taejin, Kim, Sunghwan, An, Namwon, Narantuya, Jargalsaikhan, Jeong, Chiwook, Kim, JongWon et Lim, Hyuk, 2016), qui a pour objectif d'analyser les paquets par l'IDS dans les réseaux SDN sans dépasser sa capacité d'inspection maximale. Ha et al. proposent une stratégie d'échantillonnage de trafic suspect implémentée dans les commutateurs. Les auteurs proposent un algorithme qui détermine les taux d'échantillonnage des paquets pour chacun des commutateurs SDN pour s'assurer que le total des paquets échantillonnés ne dépasse pas la capacité d'inspection maximale de l'IDS. La mise en œuvre de la solution indique que l'algorithme proposé ajuste les taux d'échantillonnage des commutateurs et maintient la probabilité d'échec de l'inspection IDS proche de zéro. Cependant, avec cette solution, il n'y a aucune garantie que l'IDS peut capturer des paquets appartenant à tous les flux parce qu'il peut manquer d'analyser beaucoup des paquets qui peuvent être malveillants.

1.5.4 Discussion

Cette sous-section présente une étude comparative entre notre solution (SDN-Guard) et les différents travaux déjà couverts dans la revue de la littérature. Le tableau 1.1 met en comparaison notre solution proposée, SDN-Guard, à celles existantes par rapport à leurs objectifs de performance ciblés lors des attaques de DdS.

Aucune des solutions existantes ne peut réduire simultanément la charge du contrôleur, la bande passante entre le contrôleur et le commutateur (notée par BW du P.C) et éviter d'inonder les tables de commutation (TCAM) des commutateurs. Dans notre travail, nous visons à atteindre simultanément ces objectifs afin de maintenir une performance de réseau acceptable lors des attaques DdS.

Tableau 1.1 SDN-Guard versus les solutions existantes

Approche	Gestion du trafic		Mitigation d'attaque en minimisant		
	Echantillon de trafic	BW du réseau	Charge du contrôleur	BW du P.C	Stockage TCAM
SDN-Guard	✓	✓	✓	✓	✓
FlowRanger	✗	✗	✓	✗	✗
Filtrage des adresses IP	✗	✗	✓	✓	✓
Technique de gestion autonome	✗	✗	✗	✗	✗
FloodGuard	✗	✓	✗	✓	✓
AVANT-GUARD	✗	✗	✓	✓	✓
SLICOTS	✗	✗	✓	✓	✓
Combinaison OF et sFlow	✓	✓	✓	✓	✗
BroFlow	✗	✓	✗	✗	✓
Approche collaborative	✗	✗	✗	✗	✓
SnortFlow	✗	✗	✗	✓	✓
échantillonnage de trafic suspect	✓	✗	✗	✗	✗

En revanche, SDN-Guard minimise le trafic inspecté par l'utilisation de la technique d'échantillonnage des paquets. De plus, SDN-Guard propose un emplacement optimal de l'IDS utilisé dans la solution pour minimiser le trafic entre les commutateurs et l'IDS ainsi que la bande passante consommée par ce trafic dans le réseau (noté par BW du réseau).

1.6 Conclusion

Dans ce chapitre, nous avons présenté la technologie SDN, les attaques de DdS et leurs effets sur la performance du réseau SDN et les systèmes de détection d'intrusion qui représentent l'un des moyens de sécurité les plus utilisés contre ces attaques. L'architecture SDN offre une intelligence centralisée dans le plan du contrôle qui est séparé de plan de transmission de données. Cette centralisation engendre une simplicité de contrôle et une gestion plus efficace du réseau afin d'optimiser sa performance. SDN offre aussi la programmabilité au niveau applicatif qui communique avec le contrôleur à traves des APIs pour qu'il s'adapte aux besoins en évolution rapide des entreprises facilement. Cependant, malgré tous ces avantages, les attaques par déni de service (DdS) sont considérées comme une menace majeure pour tel réseau,

car elles peuvent facilement surcharger la capacité de traitement du contrôleur et les tables de commutation des commutateurs avec les règles de commutation des flux, ce qui entraîne une dégradation critique de la performance globale du réseau.

Les techniques existantes ont réussi à réduire l'impact de ces attaques sur la performance du réseau. Cependant, aucune des solutions mentionnées n'est capable de réduire simultanément la charge de contrôleur, la congestion de la liaison de communication entre les commutateurs et le contrôleur et également éviter de submerger les tables des flux des commutateurs. Dans ce travail, nous visons à atteindre ces objectifs simultanément afin de maintenir une performance de réseau acceptable durant une attaque DdS.

CHAPITRE 2

SOLUTION PROPOSEÉ : SDN-GUARD

2.1 Introduction

Dans ce chapitre, nous présentons l'architecture de la solution que nous avons proposons et que nous avons baptisée, SDN-Guard. Ainsi, nous discutons ses composantes et ses principales raisons de conception qui ont été considérées afin d'atteindre les objectifs visés pour réduire la charge de traitement du contrôleur, la consommation de la bande passante du plan de contrôle et éviter le dépassement de la capacité de stockage dans les tables de commutation (TCAM) des commutateurs OpenFlow. De plus, vu que les commutateurs du réseau SDN reçoivent une énorme quantité du trafic qui doit être dupliquée vers l'IDS pour l'analyse et la détection, nous proposons un Programme Linéaire en Nombres Entiers (PLNE) qui permet d'optimiser l'emplacement de l'IDS dans le réseau afin de minimiser la quantité du trafic à analyser qui sera dirigée vers l'IDS et de réduire davantage les risques de congestion du réseau durant les attaques de DdS. Par ailleurs, nous avons recours à l'échantillonnage de flux afin de réduire davantage le trafic dupliqué par les commutateurs vers l'IDS et les risques de congestion dans le plan de données du réseau SDN.

2.2 Architecture du SDN-Guard

Comme le montre la figure 2.1, SDN-Guard est considérée comme une application SDN qui peut être hébergée au-dessus du contrôleur SDN. Elle utilise un IDS qui permet d'analyser le trafic du réseau et de générer des alertes chaque fois qu'un flux de trafic malveillant est détecté. En se basant sur les alertes provenant de l'IDS et de l'état actuel du réseau, SDN-Guard prend la décision appropriée pour chaque flux afin de minimiser l'impact des attaques de DdS sur la performance du réseau SDN.

SDN-Guard se compose de trois modules qui collaborent pour guider le contrôleur à prendre la bonne décision pour chaque flux et attribuer les règles de commutation aux commutateurs.

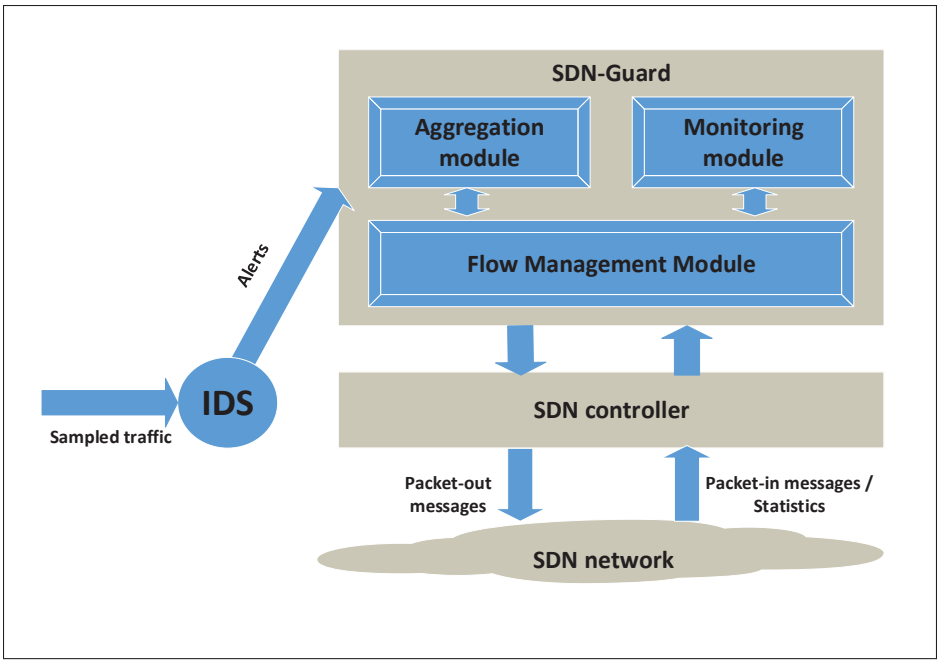

Figure 2.1 L'architecture du SDN-Guard

Ces modules s'assurent de bien gérer le trafic dans le réseau et d'éviter sa congestion lors des attaques de DdS.

2.2.1 Module de gestion des flux

Le module de gestion des flux représente le module principal du SDN-Guard qui communique avec les deux autres modules pour prendre les informations nécessaires à propos l'état du réseau afin de prendre la bonne décision.

Ce module est responsable de l'attribution des règles de commutation pour chacun des flux et décide son délai de temporisation *hard timeout*, qui représente le temps nécessaire pendant lequel l'entrée de la règle de flux reste installée dans les tables TCAM du commutateur. Le module prend ses décisions en se basant sur les alertes de menaces envoyées par l'IDS afin d'atténuer l'impact de l'attaque de DdS sur la performance du réseau.

Ce module est également chargé de sélectionner le taux d'échantillonnage de trafic à dupliquer par les commutateurs vers l'IDS pour l'analyse et la détection des attaques. Le taux d'échan-

tillonnage doit être choisi d'une façon à minimiser la charge du traitement de l'IDS tout en garantissant la précision de la détection des attaques. Le module de gestion des flux choisit également les chemins pris par le trafic dupliqué vers l'IDS d'une manière qui minimise l'utilisation des liens dans le plan de données.

2.2.2 Module d'agrégation des règles de commutation

Ce module est en charge d'agréger les entrées des règles de commutation des flux malveillants qui sont installées dans les tables de flux des commutateurs et qui ont des propriétés partagées (par exemple, le port TCP source et de destination, les adresses IP, le protocole). Ce module vise à réduire le nombre des entrées utilisées dans les TCAMs des commutateurs. Par conséquent, il permet d'éviter le problème de dépassement de la mémoire TCAM du commutateur.

2.2.3 Module de surveillance

Ce module est responsable de la surveillance et la collecte des statistiques sur les flux, les commutateurs et les liens du réseau (par exemple, le débit, l'utilisation du TCAM et de la bande passante des liens dans le plan de donnée). Ces statistiques seront utilisées par les deux autres modules. Le module de gestion des flux utilise les statistiques du ce module pour l'attribution des règles du flux convenable ainsi que pour la sélection du taux d'échantillonnage approprié. Par ailleurs, ce module communique avec le module d'agrégation des règles de commutation pour l'informer de l'état de stockage des tables de flux pour chaque commutateur dans le réseau.

2.3 Mitigation des attaques de déni de service

Dans cette section, nous détaillons les principales considérations prises dans la conception de la solution proposée afin de mitiger l'impact des attaques de DdS et DDdS sur la performance du réseau SDN. Le module de gestion des flux du SDN-Guard se base sur trois points principaux

pour attribuer et gérer les règles de flux aux commutateurs : le routage selon le type de flux, la gestion de *timeout*, et l'agrégation des règles de commutation.

2.3.1 Routage selon le type du flux

Afin d'atténuer l'impact des attaques de DdS sur la consommation de la bande passante et les délais d'attente des paquets dans la file d'attente du contrôleur, SDN-Guard, et plus précisément le module de gestion des flux, redirige le trafic malveillant à travers les chemins avec les liens les moins utilisés en termes de consommation de bande passante et de mémoire TCAM du commutateur. Ces deux métriques sont connues par le module de gestion des flux grâce aux statistiques collectées par le module de surveillance. Le chemin sélectionné pour transmettre les flux malveillants, en utilisant ces paramètres, ne doit pas être le chemin le plus court dans le réseau, qui est réservé au trafic légitime. Cependant, il assure un impact minimal de l'attaque sur la performance des flux légitimes. En même temps, le trafic malveillant atteindra la destination (ce qui est important dans le cas des alarmes fausses positives) où il peut potentiellement être analysé par des systèmes de détection ou de prévention des intrusions.

Dans notre solution, le trafic malveillant ne sera pas bloqué ou supprimé afin de s'assurer que les flux malveillants faux positifs ont la chance d'atteindre la destination, même avec des délais plus élevés. En utilisant ce concept, nous pouvons réduire l'impact du trafic malveillant sans prendre le risque de le bloquer.

Pour les flux légitimes, ils sont toujours acheminés par les chemins les plus courts entre la source et la destination afin d'assurer un délai de temps aller-retour minimum et garantir une bonne qualité de service (QoS) au trafic légitime dans le réseau.

En recevant une alerte de la part d'IDS, le module de gestion du flux prend la décision concernant l'acheminement du chacun des flux et attribue la temporisation d'activité (*timeout*) appropriée pour ses entrées correspondantes dans les tables TCAM des commutateurs. Deux cas peuvent être identifiés :

Cas 1 - Flux malicieux

Chaque fois que le module de gestion des flux reçoit une alerte de la part de l'IDS sur un flux malicieux, il attribue une valeur élevée du *hard timeout* à sa règle de flux correspondante pour s'assurer que cette règle reste une longue période dans la table de flux du commutateur et éviter les demandes de type PACKET_IN successives, envoyées par les commutateurs (demandant une règle de commutation pour le même flux), en cas de DdS. De plus, le module sélectionne les liens les moins utilisés en termes de consommation de bande passante et de mémoire TCAM du commutateur pour s'assurer que ce flux ne rivalise pas avec les flux légitimes et influe sur leur performance.

Le module d'agrégation analyse les règles générées pour ce trafic malveillant et essaie de les fusionner, lorsque cela est possible, afin de réduire leurs nombres et de réduire ainsi l'utilisation des tables de flux des commutateurs.

Cas 2 - Flux légitime

Quand il n'y a pas d'alerte pour un flux particulier, ce flux est considéré comme légitime. Le module de gestion des flux l'oriente alors vers le chemin le plus court et lui attribue une valeur de temporisation du *hard timeout* régulière qui doit être adaptée au besoin de ce trafic légitime.

Dans ce cas, l'agrégation des règles de flux est optionnelle. Grâce à l'attribution d'une valeur de temporisation appropriée pour ce type de trafic, la plupart des entrées de ses règles de flux ne restent plus longtemps installées dans les tables de commutation des commutateurs.

2.3.2 Gestion du *timeout*

Le module de gestion des flux attribue la valeur de la temporisation d'activité *timeout* de chacune des règles de commutation en fonction des alertes reçues par l'IDS. Ce *timeout* représente la durée pendant laquelle la règle de commutation doit être conservée dans la table TCAM du commutateur.

Dans notre travail, nous nous concentrons sur la valeur de temporisation «*hard timeout*» qui indique le délai maximal d'utilisation de la règle de commutation après lequel l'entrée doit être supprimée quelque soit son activité. Chaque fois que le *hard timeout* est expiré, l'entrée de la règle de flux sera supprimée de la table de commutation du commutateur.

Généralement, dans une architecture SDN, lorsqu'un nouveau flux est reçu, les commutateurs SDN doivent communiquer avec le contrôleur pour demander des règles de commutation afin de les utiliser pour transmettre ce flux vers la destination. Dans le cas d'une attaque de DdS, ces commutateurs reçoivent une grande quantité de flux. Cela sera traduit en plusieurs règles de commutation et engendre une communication excessive entre le contrôleur et les commutateurs qui demandent une nouvelle règle pour le même flux. L'attribution d'une faible valeur au *hard timeout* entraîne beaucoup plus du trafic de communication avec le contrôleur. Cela augmente non seulement la consommation de la bande passante entre les commutateurs et le contrôleur, mais aussi la surcharge du contrôleur.

Afin d'éviter ce problème, si le flux entrant est considéré comme malveillant, SDN-Guard attribue un *hard timeout* élevé à ses règles de commutation. Avec cette considération, nous nous assurons que le même flux ne déclenche pas de nombreuses communications entre le commutateur et le contrôleur. D'autre part, lorsque le contrôleur ne reçoit pas une alerte concernant un flux, il le considère comme légitime et il lui affecte une valeur *hard timeout* appropriée selon son besoin et son activité dans le réseau.

2.3.3 Agrégation des règles de commutation

Comme il est déjà mentionné dans la section précédente, les règles de commutation des flux malveillants sont attribuées des délais *hard timeout* élevés. Ainsi, ces entrées de flux resteront longtemps dans la table TCAM du commutateur. Durant une attaque de DdS, cela pourrait augmenter le nombre des entrées stockées dans les tables de commutation des commutateurs et pourrait les surcharger.

Pour résoudre ce problème, nous proposons d'agréger les entrées de règles de commutation des flux malveillants pour chaque commutateur. Ces règles de commutation sont automatiquement agrégées par le module d'agrégation des flux s'ils ont des propriétés partagées (par exemple, même source et destination, même port, etc.) et transmises au même lien de sortie. Dans ce qui suit, nous présentons un exemple pour bien comprendre le concept d'agrégation.

Comme il est déclarée dans le chapitre précédent, chaque entrée dans la table de flux est composée de trois champs (règle, action et statistiques). Le champ «règle» (appelée aussi champ d'en-tête) est utilisé pour définir la condition de correspondance à un flux, le champ «action» définit l'action à appliquer à un flux et le dernier champ «statistiques» est utilisé pour compter l'occurrence de la règle à des fins de gestion. Dans notre exemple, nous nous concentrons sur les deux premiers champs (règle et action). Le tableau (2.1) représente un exemple d'une table de flux d'un commutateur OpenFlow qui contient des règles de commutation de différents flux. Soit les cinq adresses MAC suivantes que nous allons utiliser dans notre exemple :

- A = 00 :0a :25 :33 :45 :4a ;
- B = 00 :0b :25 :33 :45 :4b ;
- C = 00 :0b :25 :33 :45 :4b ;
- D = 00 :0b :25 :33 :45 :4b ;
- E = 00 :0b :25 :33 :45 :4b.

Nous considérons aussi les cinq adresses IP suivantes correspondantes à chaque adresse MAC :

- a = 10.0.0.1 ;
- b = 10.0.0.2 ;
- c = 10.0.0.3 ;
- d = 10.0.0.4 ;
- e = 10.0.0.5.

Tableau 2.1 Exemple d'une table de commutation

Num	**Règle (en-tête)**						**Action**
	MAC src	MAC dst	IP src	IP dst	TCP sport	TCP dport	
1	A	B	a	b	17226	80	port 2
2	C	B	c	b	25350	25	port 3
3	C	B	c	b	23511	80	port 2
4	A	E	a	e	44530	80	port 4
5	E	D	e	d	37458	25	port 3
6	A	B	a	b	26532	80	port 2
7	C	B	c	b	17890	25	port 3
8	C	B	c	b	44896	25	port 3
9	A	B	a	b	17878	80	port 2
10	C	B	c	b	26734	25	port 3

Durant une attaque de DdS, les commutateurs reçoivent une énorme quantité des flux provenant d'une seule ou plusieurs sources qui visent à bloquer une victime (destination) bien déterminée. Ces flux se traduisent en règles de commutation qui doivent être installées dans les tables de flux des commutateurs. Ces dernières peuvent ainsi être saturées à cause de la capacité limitée de stockage de leurs TCAM. Afin de minimiser ce problème de dépassement de stockage, le module d'agrégation des flux va comparer les informations des entrées installées dans la table et lorsqu'il trouve une correspondance entre deux entrées ou plus, il va les fusionner en une seule entrée.

Tableau 2.2 Résultat de l'agrégation des régles de flux

Num	**Règle (en-tête)**						**Action**
	MAC src	MAC dst	IP src	IP dst	TCP sport	TCP dport	
1	A	B	a	b	*	80	port 2
2	C	B	c	b	*	25	port 3
3	C	B	c	b	23511	80	port 2
4	A	E	a	e	44530	25	port 4
5	E	D	e	d	37458	25	port 3

En appliquant ce principe sur la table de commutation présentée dans le tableau 2.1, nous obtenons le résultat de l'agrégation présenté dans le tableau 2.2. Dans notre exemple, le module d'agrégation vérifie la correspondance entre les lignes dans la table de commutation (tableau 2.1). Chaque ligne dans ce tableau représente une règle de commutation. Lorsqu'il trouve deux règles ou plus ayant les mêmes propriétés (adresse MAC source, adresse MAC destination, adresse IP source, adresse IP destination, port TCP destination et action (port de sortie)), il les fusionne dans une seule règle de flux dans la table. Nous remarquons que le nombre des flux a été diminué ; ce qui évite le débordement de la mémoire dans les commutateurs.

2.3.4 Résumé des décisions du SDN-Guard

Le tableau 2.3 ci-dessous résume les principales décisions prises par SDN-Guard selon le type de flux reçu.

Tableau 2.3 Décisions du module de gestion des flux

Type de flux	**Valeur du *Timeout***	**Chemin du routage**	**Agrégation des règles**
Légitime	approprié	plus court	optionnel
Malicieux	élevée	moins utilisé	obligatoire

Lorsque le flux est considéré comme légitime, le module de gestion des flux lui attribue une règle de commutation avec un délai du *timeout* régulier et le dirige vers le chemin le plus court afin d'atteindre la destination rapidement. Avec ce type de flux, l'agrégation des entrées (les règles de commutation) dans la table TCAM de commutateur n'est pas obligatoire. Par contre, si le flux est considéré comme malicieux, le module de gestion des flux va assigner une valeur de *timeout* élevée pour la règle de commutation de ce flux et il le redirige vers le chemin le moins utilisé en termes de consommation de la bande passante et des tables TCAM des commutateurs (ce chemin est différent de celui utilisé pour les flux légitimes). En outre, la tâche d'agrégation des règles de commutation est nécessaire dans le cas où le trafic est malicieux afin de minimiser le nombre des entrées installées dans la table de commutation

pour le même flux et d'éviter le risque du dépassement de la capacité de stockage de TCAM. Avec ces considérations, SDN-Guard maintient le bon fonctionnement du réseau SDN durant les attaques de DdS.

2.4 Emplacement de l'IDS et la gestion du trafic

SDN-Guard exploite un système de détection d'intrusions (IDS) qui analyse les flux du trafic de l'ensemble du réseau et génère des alertes chaque fois qu'un trafic malveillant est détecté, en se basant sur des règles de détection (figure 2.1).

À la réception de ces alertes, SDN-Guard va prendre les décisions appropriées pour atténuer l'impact des attaques sur la performance du réseau SDN. L'un des plus grands défis lors du déploiement de l'IDS est de minimiser le taux de trafic envoyé par les commutateurs vers l'IDS, à raison de l'analyser, car il peut consommer une énorme bande passante dans le plan de données qui engendre sa congestion. De plus, l'IDS peut être surchargé vu que la quantité de données reçues est beaucoup plus grande que sa capacité de traitement.

En pratique, il existe deux options pour déployer les systèmes de détection d'intrusion (IDS) permettant d'analyser tout le trafic réseau. La première option consist e à déployer des IDS multiples. Il consiste à connecter un IDS à chaque commutateur dans le réseau. Par la suite, chaque IDS analyse seulement le trafic traversant son commutateur associé, ce trafic représente une partie de la totalité du trafic dans le réseau. Cependant, la limitation de cette solution est que certaines attaques comme les attaques DdS distribuées (DDdS) ne peuvent pas être facilement détectées, car chaque IDS n'est pas au courant de tout le trafic réseau. Dans ce cas, des solutions plus sophistiquées doivent être élaborées pour permettre de détecter tous les types d'attaques comme la coordination et la synchronisation entre les IDS. Ces solutions consomment plus de la bande passante à cause des messages de synchronisation qui doivent être échangés entre les commutateurs et les IDS ainsi que les IDS entre eux. En plus, cette solution est très coûteuse.

La deuxième option consiste à déployer un IDS unique qui analyse tout le trafic du réseau. Dans ce travail, nous adoptons cette solution parce qu'elle a l'avantage de détecter tous les

types des attaques comme le DDdS ainsi qu'elle est moins coûteuse. En outre, cette option élimine la complexité de la synchronisation de la première option.

Cependant, il existe deux défis à relever pour implémenter cette solution. Premièrement, il n'est pas pratique que les commutateurs du plan de transmission dupliquent tout le trafic réseau vers un seul emplacement (où se trouve l'IDS), car cela consomme une grande bande passante du réseau, ce qui résulte la congestion du plan de données. Deuxièmement, l'IDS peut avoir des ressources limitées et il ne peut pas être en mesure de traiter tout le trafic réseau avec un temps de traitement acceptable. Pour résoudre ces problèmes, nous proposons un Programme Linéaire en Nombres Entiers (PLNE) qui optimise l'emplacement d'IDS afin d'éviter le premier problème et nous utilisons la technique d'échantillonnage de trafic pour résoudre le deuxième problème.

2.4.1 Emplacement optimal de l'IDS et duplication du trafic

Notre objectif est de trouver l'emplacement optimal pour l'IDS et déterminer les commutateurs qui devraient dupliquer les flux d'une manière qui minimise le volume du trafic dupliqué à l'IDS et la quantité de la bande passante consommée pendant cette opération (c.-à-d. en minimisant le nombre des liens traversés par le trafic dupliqué). Dans ce qui suit, nous exposons notre PLNE proposé pour modéliser ce problème.

Soit $G = (N, L)$ un graphe représentant le réseau, où N est l'ensemble des commutateurs et L est l'ensemble des liens qui les relient. Nous définissons $p_{n\bar{n}}$ comme le coût de plus court chemin de commutateur $n \in N$ au commutateur $\bar{n} \in N$, ce qui correspond au nombre des sauts entre les deux commutateurs.

Soit $i \in I$ désigne un flux traversant le réseau. Nous indiquons par f_i le débit du flux i. Nous définissons $r_{in\in\{0,1\}}$ comme une variable booléenne qui égale à 1 si le flux $i \in I$ a franchi le commutateur $n \in \bar{N}$.

Il est intéressant de noter que les valeurs de toutes ces variables déjà définies ($p_{n\bar{n}}$, f_i and r_{in}) sont connues par le contrôleur, car il est au courant de la topologie du réseau. Il reçoit régulièrement les statistiques des flux grâce à la caractéristique de la vue globale offerte par la technologie SDN.

En outre, un flux i ne peut pas être transféré à partir d'un commutateur n s'il ne franchit pas ce commutateur. Par conséquent, nous avons :

$$x_{in} \leq r_{in} \quad \forall n \in N \forall i \in I \tag{2.1}$$

Nous définissons également la variable de décision $x_{in} \in \{0,1\}$ comme une variable booléenne qui indique si le flux i est dupliqué par le commutateur n vers l'IDS. Chaque flux i est dupliqué une seule fois à l'IDS. Nous devons donc satisfaire la contrainte suivante :

$$\sum_{n \in N} x_{in} = 1 \quad \forall i \in I \tag{2.2}$$

Le coût de duplication de tous les flux vers un IDS connecté à un commutateur $\bar{n} \in N$ correspond à la quantité totale du trafic dupliqué qui est transféré par tous les commutateurs du réseau vers l'IDS. Ce coût peut être calculé comme suit :

$$\mathscr{C}_{\bar{n}} = \sum_{i \in I} \sum_{n \in N} x_{in} p_{n\bar{n}} f_i \qquad \forall n \in N \tag{2.3}$$

Enfin, notre objectif ultime est de trouver le commutateur approprié $\bar{n} \in N$ qui minimise ce coût de duplication du flux. Ainsi la fonction objective peut s'écrire comme suit :

$$\min_{\bar{n} \in \bar{N}} \quad \mathscr{C}_{\bar{n}} \tag{2.4}$$

Ce programme linéaire proposé fournit la position optimale de l'IDS ($\bar{n}$) ainsi que les commutateurs qui doivent transmettre le trafic à analyser vers l'IDS (en utilisant la variable de décision x_{in}) afin de réduire le taux de duplication de trafic par les commutateurs et éviter le problème

de congestion dans le plan de transmission du réseau SDN. Par contre, notre PLNE ne peut pas résoudre le problème de l'emplacement de l'IDS dans un réseau à grande échelle. Ceci est développé pour un réseau à petite échelle avec un nombre limité des flux. L'élaboration d'un algorithme PLNE capable de trouver l'emplacement optimal pour l'IDS dans un réseau à plus grande échelle sera notre prochain objectif à poursuivre dans le futur.

2.4.2 Échantillonnage de trafic dupliqué

Pour réduire davantage la quantité de trafic dupliquée par les commutateurs vers l'IDS, nous échantillonnons le trafic avant qu'il ne soit transmis à l'IDS. L'échantillonnage va réduire la charge du traitement de l'IDS et la quantité de trafic analysée sans nuire à sa performance et sa précision de détection des attaques. De plus, ce processus permet d'éviter les risques de congestion dans le plan de transmission du réseau SDN.

Avec cette solution, on minimise la quantité de données analysée par l'IDS toute en garantissant la précision de la détection pour que le contrôleur prenne, par la suite, les contre-mesures nécessaires afin de mitiger l'impact de l'attaque de DdS sur le réseau SDN. Dans le chapitre suivant, nous étudions davantage l'impact du taux d'échantillonnage pour déterminer le taux d'échantillonnage optimal qui ne nuirait pas à la performance et à la précision de l'IDS.

2.5 Conclusion

Dans ce chapitre, nous avons présenté notre solution SDN-Guard qui permet de mitiger l'impact de l'attaque de DdS sur la performance de réseau SDN. La solution SDN-Guard est considérée comme une application SDN qui utilise le contrôleur SDN. Pour la détection des attaques et l'identification des flux, SDN-Guard s'appuie sur un IDS pour analyser le trafic et décider des degrés de la menace pour chaque flux entrant au réseau. SDN-Guard se compose en trois modules : le module de gestion des flux, le module de surveillance et le module d'agrégation des règles de commutation. Ces modules collaborent afin de prendre la décision appropriée pour chaque flux et aider le contrôleur à l'assignation des règles de commutation pour chaque

flux pour qu'il soit transmis par les commutateurs du plan de transmission. De plus, pour assurer le bon fonctionnement de SDN-Guard, nous prenons en considération plusieurs aspects tels que l'emplacement de l'IDS et sa façon de gérer le trafic analysé pour qu'il consomme le minimum de bande passante pour atteindre l'IDS. Afin d'atteindre cet objectif, nous proposons un Programme Linéaire au Nombres Entiers (PLNE) qui localise l'emplacement idéal de l'IDS. De plus, nous avons eu recours à la technique d'échantillonnage de trafic pour minimiser la quantité de trafic dupliqué dans le réseau. Avec SDN-Guard, la décision de routage du chaque flux est basée sur son type. Si le flux est considéré comme légitime, le contrôleur va lui attribuer une règle de commutation adaptée avec l'état du réseau. Sinon, lorsque SDN-Guard reçoit une alerte de la part d'IDS, le flux sera considéré comme malveillant, il va lui attribuer une valeur élevée du *hard timeout* afin de minimiser la communication entre le contrôleur et les commutateurs demandant des règles de commutation pour le même flux malicieux et le diriger vers le chemin le moins utilisé en termes de consommation de bande passante et de stockage de TCAM des commutateurs pour éviter la congestion dans le plan de transmission. Enfin, notre solution propose l'agrégation des règles de commutation installées dans les TCAMs des commutateurs afin d'éviter le problème de dépassement de la mémoire TCAM des commutateurs.

SDN-Guard offre une solution fiable et adaptable qui résout les problèmes considérés et maintient une performance du réseau SDN acceptable durant une attaque DdS. Dans le chapitre suivant, nous allons décrire les expériences réalisées pour tester la solution proposée. De plus, nous allons présenter et discuter les résultats obtenus qui évaluent la précision et la performance de l'IDS ainsi que l'efficacité du SDN-Guard.

CHAPITRE 3

EXPÉRIMENTATIONS ET RÉSULTATS

3.1 Introduction

Dans ce chapitre, nous décrivons l'environnement expérimental utilisé pour émuler un réseau SDN et les outils utilisés pour générer le trafic et les attaques DdS. Ensuite, nous commençons par investiguer l'impact de l'emplacement d'IDS sur la quantité de trafic dupliqué ainsi que celui de l'échantillonnage du trafic sur la précision et la performance de l'IDS. Ensuite, nous évaluons l'efficacité du SDN-Guard pour atténuer l'impact des attaques DdS en le comparant avec l'approche de base (sans SDN-Guard).

3.2 Environnement expérimental

Pour réaliser les expériences requises, nous utilisons Mininet 2.3.0 (Mininet Team, 2017) pour créer des réseaux SDN composés de commutateurs SDN contrôlés par un contrôleur. Mininet est un émulateur du réseau SDN populaire qui permet de créer un réseau virtuel qui se compose par des hôtes, des liens et des commutateurs *OpenVSwitch* (OVS) (A Linux Foundation Collaborative Project, 2016). Les hôtes dans Mininet exécutent un logiciel de réseau Linux standard et ses commutateurs prennent en charge le protocole OpenFlow pour la communication avec le contrôleur.

Le réseau créé est contrôlé par Floodlight 1.2, qui est un contrôleur SDN basé sur java et qui est largement utilisé (Project Floodlight, 2016) dans les environnements SDN. Floodlight est offert par Big Switch Networks (Big Switch Networks, 2017) qui utilise le protocole OpenFlow pour gérer les trafics dans un environnement SDN. Ce contrôleur SDN est à source ouverte (open source) et il offre aux développeurs la possibilité d'adapter les logiciels facilement et de développer des applications SDN (SDx central, 2017). Pour assurer la communication entre les commutateurs OVS et le contrôleur Floodlight, nous utilisons la version 1.3 du protocole OpenFlow (Brocade Communications Systems, 2017).

Nous utilisons également Snort (version 2.0.6), comme un système de détection d'intrusion, qui analyse le trafic du réseau et envoie des alertes à SDN-Guard au cas où il détecte un flux malveillant dans le réseau.

Toutes les expériences sont effectuées sur un serveur exécutant Ubuntu 14.04 comme système d'exploitation et ayant un processeur Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz x 8 et 8 GB de mémoire vive.

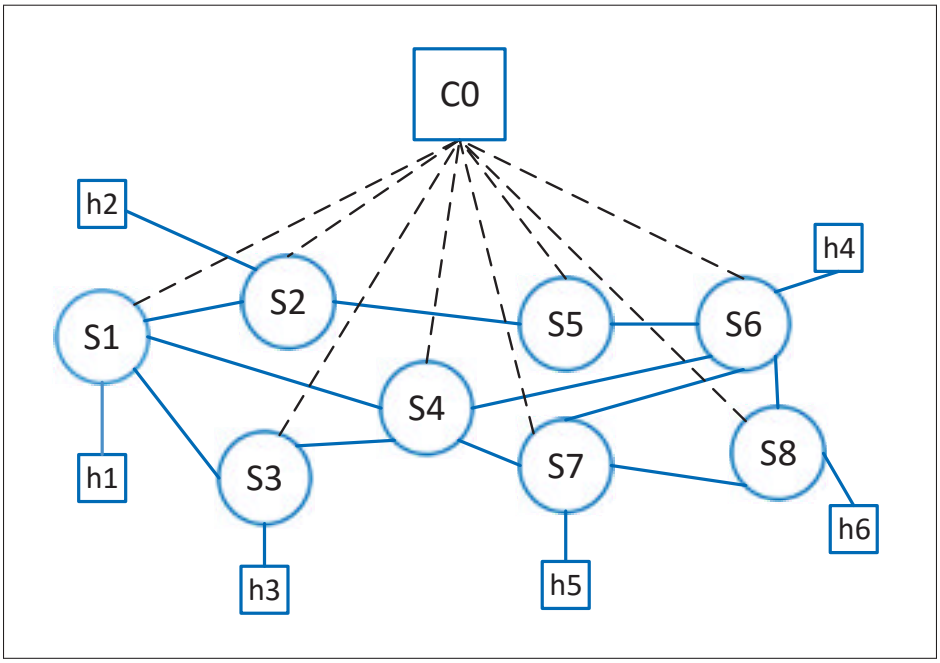

Figure 3.1 La topologie emulée

3.2.1 Topologie émulée

Afin d'évaluer notre proposition, nous avons émulé une topologie composée de huit commutateurs OVS et six hôtes, comme indiqué dans la figure 3.1. Cette topologie est sous la responsabilité d'un contrôleur Floodlight c_0 qui va gérer le flux circulant entre les commutateurs et les hôtes, attribuer les règles de commutation pour chaque nouveau flux et garder le bon fonctionnement du réseau. Chaque commutateur est directement connecté au contrôleur c_0 afin qu'il puisse être informé des décisions de transmission déterminées par SDN-Guard. Nous définis-

sons les hôtes h_1, h_2 et h_3 comme des attaquants alors que l'hôte h_6 est le serveur ciblé par ces derniers qui vont déclencher une attaque de DdS afin de le saturer.

3.2.2 Description des expériences effectuées

Nous avons divisé notre expérience en trois intervalles de temps. La durée du chaque intervalle était de dix minutes. La première phase consiste à envoyer un trafic normal avec des flux TCP provenant de toutes les sources vers toutes les destinations dans le réseau. Une attaque de DdS commence par la suite par les hôtes attaquants et dure dix minutes pendant lesquelles le serveur h_6 est inondé avec un grand nombre de nouveaux flux TCP. Après la phase d'attaque, le trafic retourne à son comportement normal et seule une quantité normale de flux TCP est envoyée au serveur ciblé.

Pour lancer l'attaque DdS, nous envoyons le trafic TCP en utilisant l'outil réseau *hping3* (die.net, 2017) qui inonde le serveur h_6 avec un grand nombre de paquets TCP SYN, ICMP et UDP avec différentes adresses IP. Ce type de trafic est équivalent à une attaque de DdS distribuée provenant de sources multiples.

3.3 Évaluation de la performance et de la précision d'IDS

Dans la première partie de notre étude, nous cherchons à atteindre l'objectif d'obtenir une détection précise par l'IDS sans dépasser sa capacité de traitement. Pour cela, nous évaluons la précision de l'IDS à la détection des attaques lorsqu'il analyse des échantillons de trafic du réseau et non sa totalité. Nous analysons également les performances de l'IDS en termes de temps de traitement des paquets avec des taux d'échantillonnage différents. De plus, nous investiguons sur l'emplacement idéal d'IDS qui minimise la quantité du trafic dupliquée et envoyée à l'IDS pour être analyser afin d'éviter la congestion en bande passante dans le réseau.

Dans notre expérience de l'évaluation de la performance de l'IDS, nous générons trois types d'attaques DdS : les inondations TCP-SYN, les inondations UDP et les inondations ICMP. Ces attaques sont lancées simultanément pendant 30 minutes pour observer le comportement et la

précision de détection de l'IDS lorsque le réseau est ciblé de plusieurs types d'attaques DdS en même temps. Nous exécutons ce scénario avec des taux de suppression de paquets (*dropping rate*) différents. Un taux de suppression de paquets de p% signifie que p% du trafic dupliqué est supprimé aléatoirement par les commutateurs avant d'être transmis à l'IDS.

3.3.1 Précision de l'IDS versus le taux d'échantillonnage

Dans ce paragraphe, nous commençons par évaluer la précision de l'IDS à la détection des attaques lorsqu'un trafic échantillonné est utilisé au lieu de la totalité du trafic. Le taux d'échantillonnage représente la quantité du trafic restante (aprés la suppression des paquets), qui va être dupliquée par les commutateurs et analysée par l'IDS.

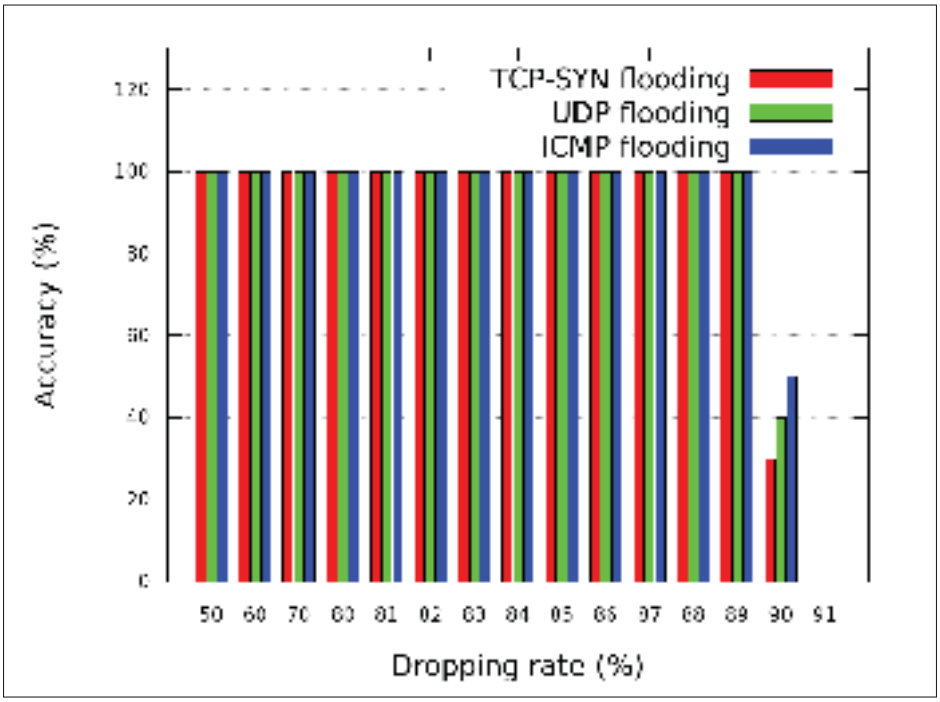

Figure 3.2 La précision du Snort par rapport au taux de suppression de paquets pour chaque type d'attaque de DdS

La précision est mesurée par le pourcentage des attaques détectées, qui représente le nombre des attaques détectées avec succès lorsque le trafic échantillonné est analysé divisé par le

nombre total des attaques détectées lorsque la totalité du trafic est analysée (sans échantillonnage).

La technique d'échantillonnage a été implémentée en utilisant le protocole OpenFlow et la commande *tc* à l'interface du commutateur OVS. Le taux de suppression de paquets est spécifié par le contrôleur, qui envoie un paquet OpenFlow spécifique (que nous avons créé) indiquant la valeur du pourcentage (p%) du trafic dupliqué qui doit être supprimé par le commutateur OVS. Ce dernier utilise cette valeur pour supprimer de manière aléatoire les paquets du trafic dupliqué à l'IDS selon ce taux d'échantillonnage (p%) spécifié.

La figure 3.2 montre la précision de l'IDS pour des différents taux de suppression de paquets pour trois types d'attaques. Elle indique clairement que la précision de Snort reste à 100% pour un taux de suppression de paquets inférieur à 89% et elle commence à diminuer lorsque le taux de suppression de paquets dépasse 89%.

Cela signifie que l'IDS est capable de détecter les attaques DdS avec une précision de 100% en analysant seulement 11% du trafic global du réseau. Cela prouve qu'on peut diriger seulement 11% du trafic vers l'IDS pour être analysé. Par conséquent, cela permet de réduire significativement la consommation de la bande passante dans le réseau.

3.3.2 Temps du traitement des paquets par IDS

L'échantillonnage offre un autre avantage très important pour notre solution. Il réduit la charge du travail de l'IDS, vu qu'il n'analyse pas la totalité du trafic ce qui conduit donc à réduire son temps de traitement pour les paquets. Le temps de traitement des paquets est le temps nécessaire pour Snort afin d'analyser un paquet. Intuitivement, cela dépend du nombre de règles de sécurité vérifiées par Snort pour chaque paquet ainsi que sa charge du travail.

La figure 3.3 montre le temps moyen de traitement des paquets dans Snort avec des taux de suppression de paquets différents. Les résultats prouvent que sans échantillonnage de trafic (le taux de suppression de paquets est de 0%), c-à-d. lorsque Snort reçoit tout le trafic réseau,

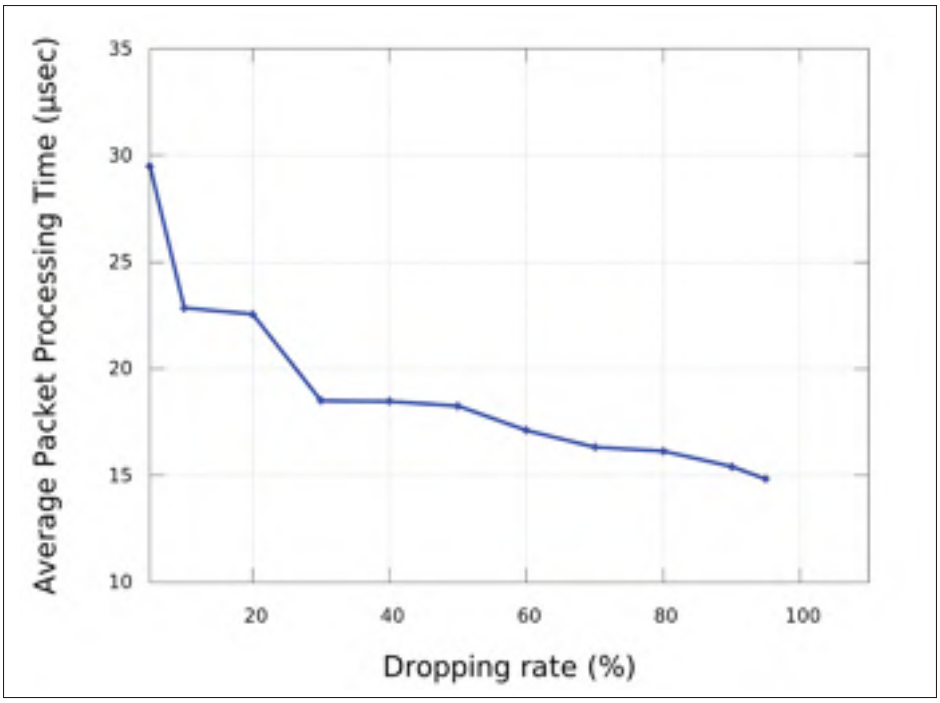

Figure 3.3 Temps moyen de traitement des paquets par Snort

le temps moyen de traitement d'un paquet est d'environ 30 μsec. Ensuite, ce temps diminue progressivement jusqu'à 16 μsec lorsque le taux de suppression de paquets augmente à 91%. Cette diminution du temps de traitement de paquet permet d'assurer une communication plus rapide entre l'IDS et SDN-Guard ce qui minimise le temps de prise de décision et de protection par SDN-Guard aussi.

À partir de ces résultats, on peut conclure qu'avec un taux d'échantillonnage de 11%, SDN-Guard réduit non seulement la quantité de trafic dupliqué dans l'IDS mais, aussi réduit le temps de traitement des paquets jusqu'à 50% tout en gardant la précision d'IDS à 100%.

3.4 Emplacement optimal de l'IDS

Nous avons également investigué sur le meilleur emplacement pour l'IDS dans le réseau. Cet emplacement doit être idéal de façon à minimiser la quantité de trafic dupliquée par les commutateurs vers l'IDS et réduire la quantité de bande passante consommée par ce trafic.

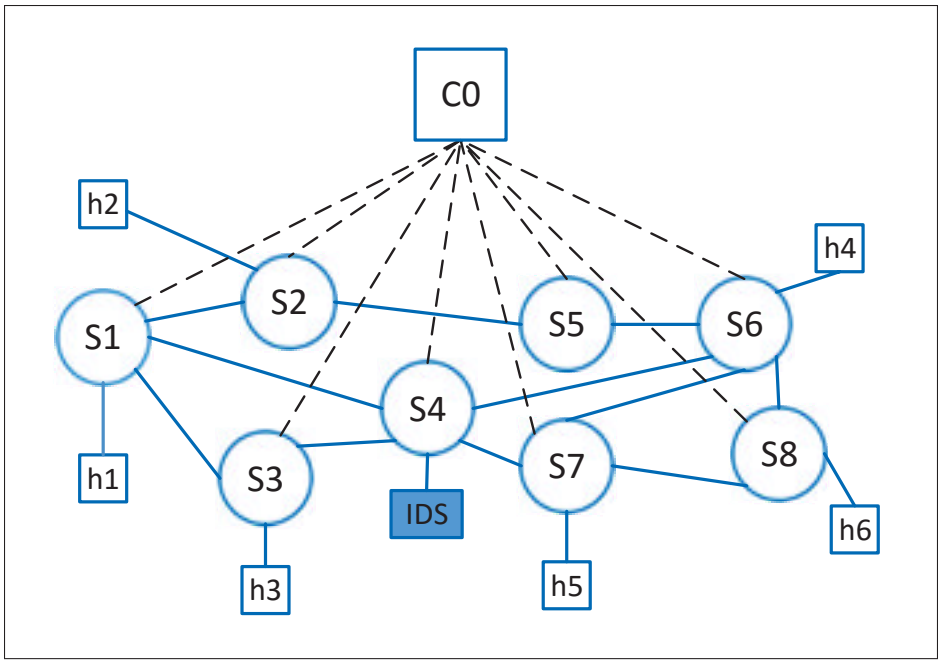

Figure 3.4 Emplacement optimal de l'IDS

Pour trouver l'emplacement optimal de l'IDS dans notre topologie, nous avons recours à ILP solver *lpsolve* (Sourceforge, 2017) dont lequel nous implémentons notre programme linéaire d'optimisation proposé dans le chapitre précèdant. Le commutateur S_4 est choisi par *Ipsolve* comme l'emplacement optimal de l'IDS (figure 3.4) afin de minimiser le trafic analysé ainsi que la bande passante consommée dans le plan de données du réseau SDN.

Pour confirmer le choix de cet emplacement, nous exécutons plusieurs expériences tout en variant à chaque fois l'emplacement de l'IDS. Les expériences réalisées utilisent la même configuration décrite à la section 3.1 sans l'utilisation de la technique d'échantillonnage du trafic.

Dans chaque expérience, nous connectons l'IDS à l'un des commutateurs de la topologie et nous calculons la quantité de trafic qui a été dupliquée à l'IDS par les autres commutateurs. Ce paramètre est pertinent pour la performance de l'IDS, car ce trafic consomme une énorme quantité de bande passante dans le réseau pour accéder à l'IDS.

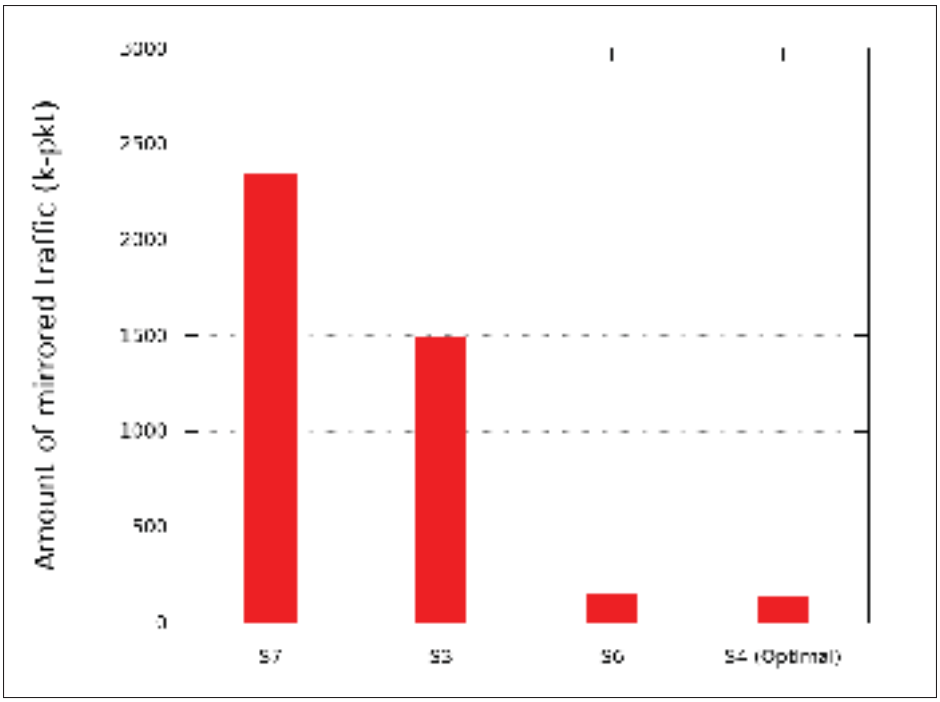

Figure 3.5 Quantité de trafic dupliquée selon l'emplacement de l'IDS

La figure 3.5 représente la quantité de trafic dupliquée reçu par l'IDS dans des différents emplacements (commutateurs). Cette quantité ne tient pas compte du trafic qui est naturellement transmis par le commutateur directement connecté à l'IDS. Ce trafic atteint l'IDS sans coût donc le commutateur n'a pas besoin d'une duplication de son trafic. Par exemple, si l'IDS est connecté au commutateur S_4, le trafic traversant naturellement S_4 sera directement transmis à l'IDS sans coût, par conséquent, nous ne le considérons pas comme trafic additionnel dirigé vers l'IDS. Nous considérons uniquement le trafic transféré par tous les autres commutateurs vers le commutateur S_4 parce que ce trafic consomme la bande passante du réseau pour atteindre l'IDS.

A partir de la figure 3.5, nous pouvons constater que, lorsque l'emplacement optimal de l'IDS est utilisé, c'est-à-dire le commutateur S_4, la quantité de trafic est plus basse par rapport aux autres emplacements tels que les commutateurs S_7, S_3 et S_6. En particulier, la quantité de trafic dupliquée dans cet emplacement optimal est d'environ 90% inférieure à d'autres emplacements comme le commutateur S_7 par exemple.

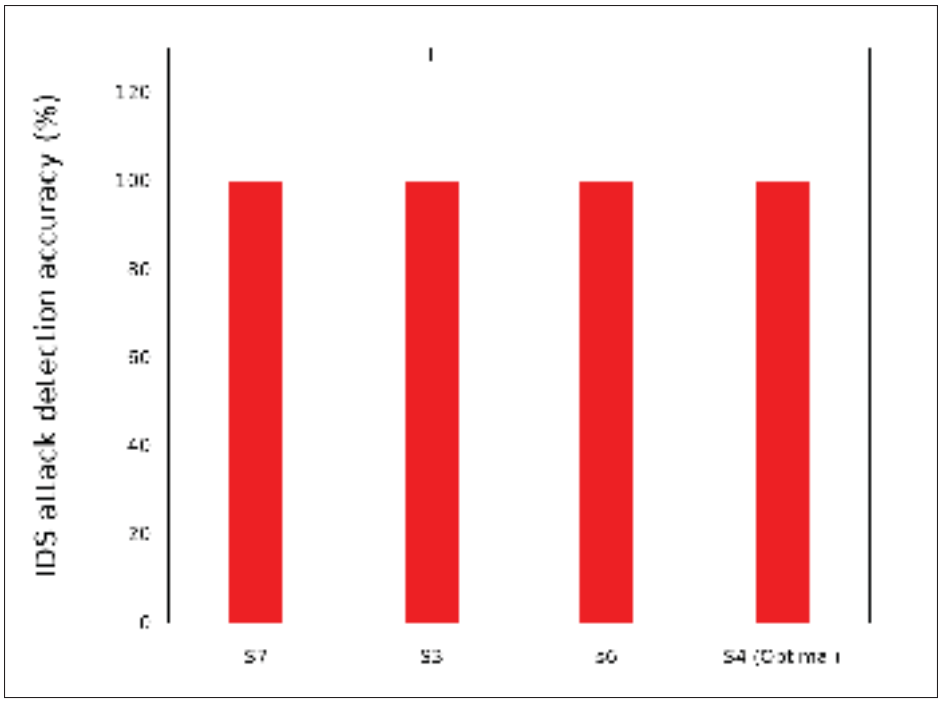

Figure 3.6 Précision de l'IDS dans ces différents emplacements

Il est également intéressant de noter que la précision de l'IDS reste à 100% indépendamment de son emplacement (figure 3.6). Ceci est attendu car l'IDS reçoit le même trafic, quel que soit son emplacement. Cependant, l'emplacement optimal de l'IDS permet un minimum de trafic dupliqué et permet ainsi d'éviter la congestion du réseau.

3.5 Évaluation de l'efficacité du SDN-Guard

Dans cette deuxième partie de notre étude, nous nous concentrons sur l'étude de l'efficacité du SDN-Guard pour mitiger l'impact des attaques de déni de service sur la performance des réseaux SDN. Nous analysons principalement les métriques de performance suivantes : le taux de traitement du contrôleur, la taille moyenne des tables TCAM des commutateurs, le taux de perte des paquets et le temps moyen de réponse pour les paquets (Round Trip Time - RTT).

Pour vérifier les avantages du mécanisme proposé, nous exécutons la même expérience deux fois : une fois avec l'algorithme de routage de base qui utilise le plus court chemin pour transmettre les paquets quel que soit leurs degrés de menace (c.-à-d., sans SDN-Guard et sans IDS)

et une autre fois où SDN-Guard, l'IDS et les contre-mesures pour atténuer l'impact de DdS sont activés.

3.5.1 Taux de traitement du contrôleur

Pour étudier le comportement du contrôleur lors d'une attaque de DdS, nous analysons le débit des messages de type PACKET_IN reçus par le contrôleur depuis tous les commutateurs du réseau. Ces messages sont envoyés par les commutateurs pour demander des règles de commutation afin de transmettre un nouveau flux entrant qui n'a pas d'entrée dans leurs tables de commutations.

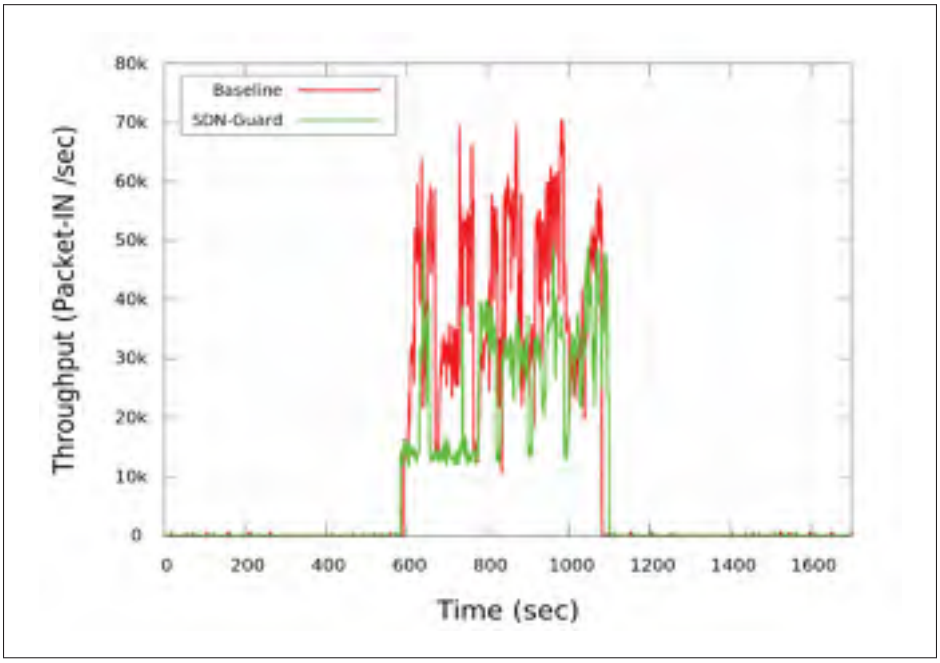

Figure 3.7 Débit entrant du contrôleur

Le débit des messages PACKET_IN reçus par le contrôleur Floodlight est présenté dans la figure 3.7. Il est clair que, lors de l'attaque, il y'a une augmentation du nombre des PACKET_IN reçus par Floodlight. Cependant, comparant à la technique de base (sans SDN-Guard), nous pouvons voir que SDN-Guard réussit à réduire ce débit jusqu'à 32% en moins. Cela est principalement dû au fait que SDN-Guard définit des valeurs élevées au temporisateur *hard timeout*

pour les règles de commutation associées aux flux malveillants, ce qui réduit considérablement la nécessité de reconsulter le contrôleur pour les nouvelles règles de flux. Bien que la charge du contrôleur diminue, nous pouvons conclure que la charge dans la bande passante du plan de contrôle (le lien de communication entre le contrôleur et les commutateurs) sera également réduite.

3.5.2 Taille des tables TCAM des commutateurs

La table de commutation peut être facilement inondée avec les attaques DdS à cause de la quantité énorme des nouveaux flux envoyés aux commutateurs. Cela nécessite un grand nombre de règles de flux à stocker.

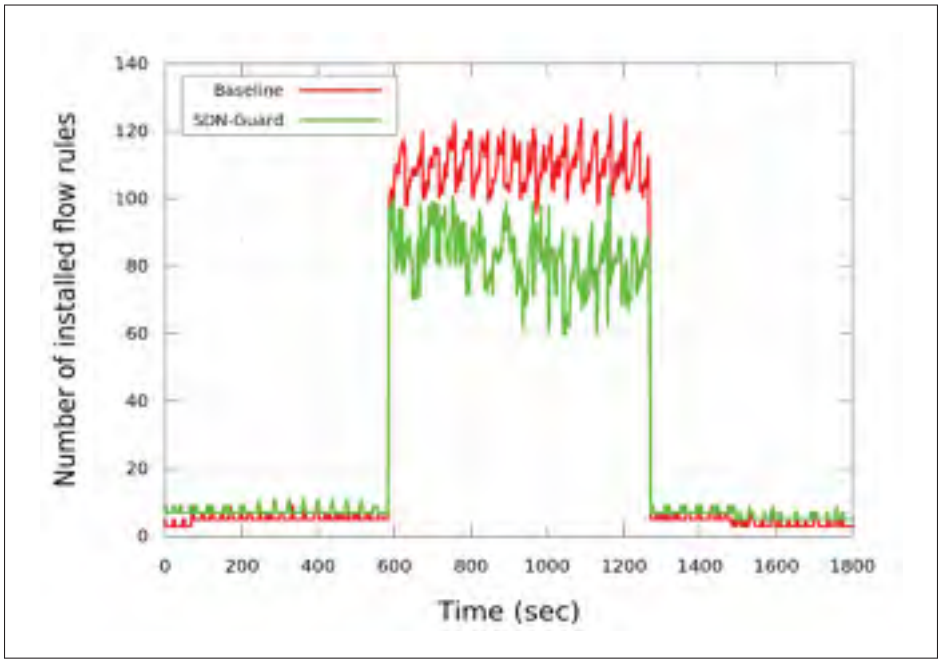

Figure 3.8 Taux de stockage moyenne de la table TCAM du commutateur S_1

La figure 3.8 prouve qu'avec SDN-Guard, le nombre de règles de flux dans la table TCAM du commutateur S_1 est réduit jusqu'à 26% par rapport à la technique sans SDN-Guard. Nous notons également que des résultats similaires ont été trouvés dans les autres commutateurs

utilisés dans la topologie de l'expérience. Ce résultat est obtenu à cause des deux considérations de conception qui sont prises :

1) le trafic malveillant est transmis par les liens les moins utilisés en termes de consommation de bande passante et de capacité de stockage dans les tables TCAM des commutateurs. Cela signifie que les entrées des règles des flux seront partagées par les commutateurs dans les différents chemins (c.-à-d. on ne se limite pas aux commutateurs du chemin le plus court) ;
2) le module d'agrégation s'assure de minimiser le nombre des règles de commutation des flux malveillants installées dans les tables TCAM en les regroupant à l'aide des propriétés communes (même source et destination, même prochain saut, etc.).

3.5.3 Taux de perte des paquets

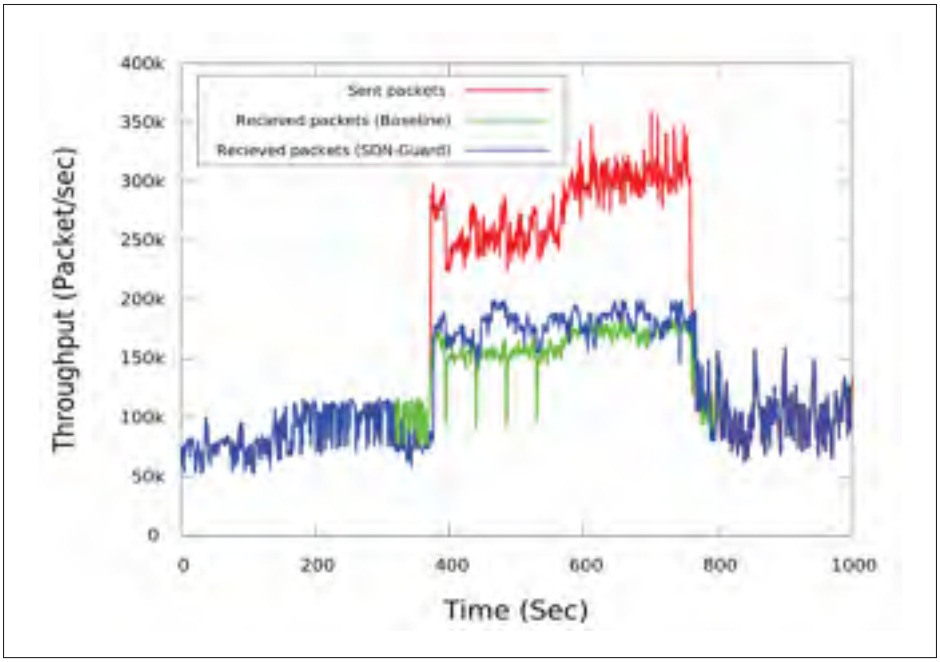

Figure 3.9 Nombre de paquets envoyés par h_1 et reçus par h_6

Nous évaluons également l'efficacité de SDN-Guard sur le taux de perte des paquets dans le réseau pendant l'attaque de DdS. Pour étudier cette métrique de performance, nous mesurons le nombre de paquets envoyés à partir de la source h_1 et reçus à la destination h_6.

A partir des résultats dans la figure 3.9, nous constatons qu'avant l'attaque, il n'y a presque pas de perte de paquet car le nombre de paquets envoyés par la source est égal à ceux reçus à la destination. Cependant, lors de l'attaque, le débit reçu est considérablement inférieur à celui qui a été envoyé. Cela implique un taux de perte de paquets élevé.

Avec SDN-Guard, le débit reçu est relativement moins affecté par l'attaque. En analysant les résultats obtenus lors de l'attaque, nous trouvons que, sans SDN-Guard, il y a 40% de perte de paquets, alors qu'avec SDN-Guard le taux de perte de paquets est de 35% seulement. Cette diminution est atteinte par SDN-Guard car le trafic malveillant est équilibré sur les liens les moins utilisés. Cela réduit les risques de congestion dans le trafic.

3.5.4 Temps moyen de réponse

Nous analysons aussi le temps de réponse entre la source et la destination (*Roud Trip Time* - RTT). Cette métrique est très importante pour évaluer la performance du réseau SDN.

Nous extrayons donc la valeur moyenne de RTT durant notre expérience afin d'étudier l'impact de l'attaque DdS sur ce paramètre dans les cas avec et sans SDN-Guard.

Nous pouvons voir dans la figure 3.10 que la valeur moyenne de RTT diminue jusqu'à 23% lorsque SDN-Guard est activé. Cette diminution a lieu à cause des délais élevés de *hard timeout* pour le trafic malveillant. Grâce à cette décision, les commutateurs demandent moins souvent de nouvelles règles de commutation pour le même flux en cas de DdS. Ainsi, cela élimine le temps nécessaire pour envoyer la requête PACKET_IN au contrôleur et l'attente de la règle de commutation afin de transmettre ce flux.

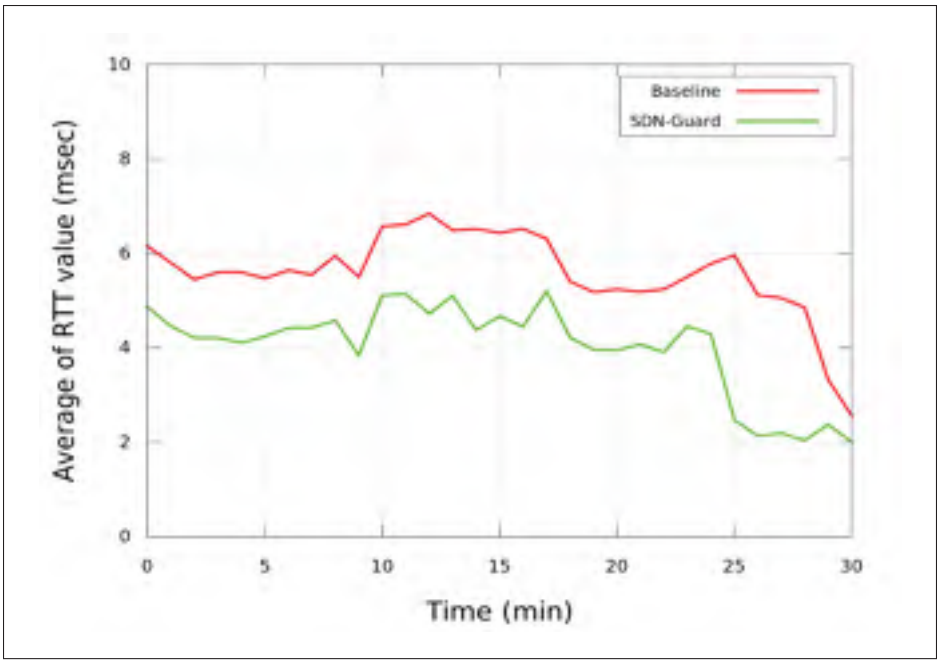

Figure 3.10 Temps de réponse moyen entre h_1 et h_6

3.5.5 Discussion

Ce paragraphe résume tous les résultats obtenus lors de l'évaluation de la performance de SDN-Guard.

La figure 3.11 illustre les avantages de l'utilisation du SDN-Guard dans le réseau SDN pour mitiger l'impact des attaques de DdS. La figure compare les métriques de performance, qui ont été évaluées précédemment dans cette section, dans le cas avec SDN-Guard et dans le cas sans SDN-Guard (noté par Baseline). Nous calculons le taux de variation de chaque métrique par rapport à celui-ci du *Baseline*. Nous pouvons remarquer qu'avec SDN-Guard le réseau a réussi de minimiser le débit de traitement des requêtes au niveau du contrôleur (notée par Throughput (CP)), la capacité de stockage dans les tables TCAM des commutateurs (notée par TCAM size) et le temps moyen de réponse entre la source et la destination (noté par RTT). De plus, nous observons aussi que le taux de perte de paquet a diminué jusqu'à 5% par rapport au *Baseline*. La figure montre aussi que le nombre des paquets reçus à la destination (noté Throughput (DP)) avec SDN-Guard est supérieur à celui du *Baseline*.

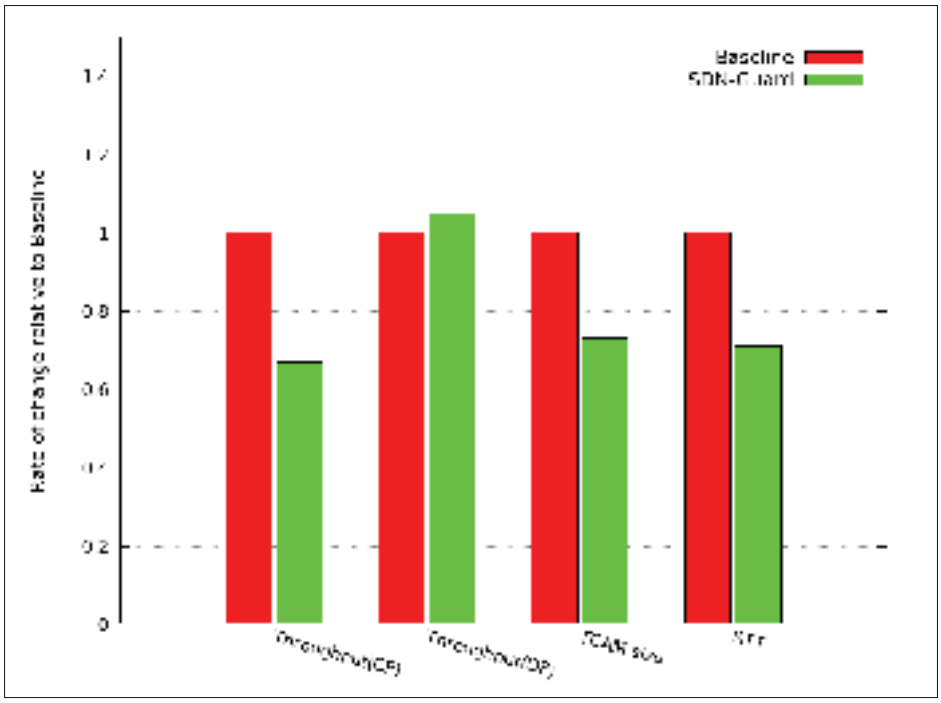

Figure 3.11 L'efficacité du SDN-Guard en termes des métriques de performance

3.6 Conclusion

Dans ce chapitre, nous avons présenté les expériences réalisées et avons discuté les résultats obtenus. La première partie du chapitre étudie l'emplacement de l'IDS dans le réseau SDN ainsi que l'impact de cet emplacement sur sa performance et sa précision dans la détection des attaques. Nous avons ensuite étudié l'utilisation de l'échantillonnage pour réduire la quantité de trafic dupliquée et envoyée à l'IDS à partir des commutateurs et de réduire le temps nécessaire pour traiter les paquets. Nous avons constaté que le fait de placer l'IDS dans un emplacement optimal pourrait diminuer jusqu'à 90% du trafic dupliqué vers l'IDS. Nous avons également trouvé qu'avec l'analyse de 11% du trafic, Snort peut détecter avec une précision élevée les attaques DdS. Dans la deuxième partie, nous avons évalué l'efficacité de SDN-Guard. Les expériences réalisées en utilisant Mininet ont montré que SDN-Guard réussit à minimiser l'impact des attaques DdS et réduit la charge de traitement du contrôleur de 32%, ce qui mène à réduire la communication entre le contrôleur et les commutateurs. De plus, SDN-Guard diminue jus-

qu'à 26% l'utilisation de la mémoire de stockage des règles de flux dans les commutateurs. En outre, les résultats prouvent que SDN-Guard réduit considérablement la perte de paquets et le temps de réponse moyen dans le réseau SDN durant les attaques DdS.

CONCLUSION ET PERSPECTIVES

La réseautique définie par logiciel (SDN) a récemment émergé comme une nouvelle technologie de réseau annonçant des changements importants sans précédent qui permet aux opérateurs de réseau de gérer dynamiquement leurs infrastructures. Cela sera permis grâce à l'ouverture, la virtualisation et l'orchestration offerte par le nouveau paradigme de gestion du réseau avec SDN. Le concept de SDN consiste à séparer la partie contrôle de la partie infrastructure du réseau qui traite et achemine les données.

Malgré ses avantages, l'attaque de déni de service (DdS) est considérée comme la menace majeure pour ces réseaux. À cause de la gestion centralisée dans l'architecture SDN, ce type d'attaque peut facilement surcharger le contrôleur SDN et les tables de commutation (TCAM) des commutateurs, ce qui entraîne une dégradation critique de la performance du réseau. L'attaque DdS est un fléau qui trouble la sécurité et le bon fonctionnement du réseau SDN. Dans ce mémoire, nous nous sommes intéressés à diminuer l'impact de ce type d'attaque sur la performance du réseau SDN. Pour atteindre cet objectif, nous avons proposé SDN-Guard, une nouvelle approche destinée à protéger les réseaux SDN contre les attaques de DdS et mitiger leurs répercussions. SDN-Guard est considéré comme une application SDN implémentée dans le contrôleur SDN et qui permet à ce dernier de mieux gérer les flux dans le plan de transmission et le plan de contrôle lors d'une attaque de déni de service.

SDN-Guard exploite un système de détection d'intrusion (IDS) pour détecter les éventuelles attaques de DdS. En effet, il s'appuie sur les alertes générées par l'IDS, qui analyse l'ensemble du trafic, afin de prendre la bonne décision concernant les règles des commutations attribuées aux commutateurs. À la réception d'une alerte, SDN-Guard peut atténuer efficacement l'impact de l'attaque par le réacheminement du trafic malveillant, l'ajustement des délais de l'utilisation des règles de commutation de ces flux et l'agrégation des règles de commutation installées dans les tables TCAM des commutateurs.

Par ailleurs, nous avons proposé des solutions permettant de minimiser le trafic entre commutateurs et l'IDS sans avoir un impact sur sa précision de détection. Nous avons proposé donc d'utiliser l'échantillonnage du trafic. De plus, nous avons étudié l'impact de l'emplacement d'IDS sur sa performance en termes de la quantité de données analysée, sa précision de détection et son temps de traitement de paquets. Un Programme Linéaire au Nombres Entières (PLNE) a été élaboré pour permettre de trouver l'emplacement optimal pour l'IDS et de déterminer les commutateurs qui devraient dupliquer les flux de manière à minimiser la consommation de bande passante dans le réseau.

À travers des expériences approfondies utilisant l'émulateur réseau Mininet et Snort comme un IDS, nous avons trouvé que SDN-Guard réussit à réduire l'impact des attaques DdS et réduit de 32% la charge de traitement du contrôleur ainsi que la bande passante consommée entre le contrôleur et les commutateurs. Les résultats ont prouvé aussi que SDN-Guard diminue l'utilisation de la mémoire de stockage des commutateurs jusqu'à 26%. Par conséquent, SDN-Guard réduit considérablement la perte de paquets et le temps de réponse moyen dans le réseau SDN pendant les attaques DdS. Nous avons également montré que la précision de l'IDS lors de la détection des attaques DdS reste à 100% en analysant seulement 11% du trafic réseau et que cette quantité de trafic envoyé vers l'IDS peut être encore réduite de jusqu'à 90% lorsque l'IDS est situé au meilleur emplacement qui minimise le trafic échangé entre les commutateurs et l'IDS.

Il existe de nombreuses directions prometteuses que nous pouvons poursuivre dans le futur. Nous prévoyons d'évaluer les performances du SDN-Guard pour des déploiements plus réalistes et à plus grande échelle. Une autre voie intéressante serait de proposer un algorithme capable de trouver un emplacement optimal pour l'IDS dans un réseau à plus grande échelle, puisque le PLNE proposé ne peut être appliqué que dans un réseau à petite échelle et avec un nombre limité du flux.

BIBLIOGRAPHIE

A Linux Foundation Collaborative Project. (2016). «OpenVswitch». Repéré à <http://openvswitch.org/>.

Ali Elamrani Joutei. (2014). «Inside OpenFlow ! ConfigNetworks». Repéré à <http://confignetworks.com/inside-openflow/>.

altospam. (2017). «Qu'est ce qu'une attaque DoS, par déni de service ?». Repéré à <http://www.altospam.com/glossaire/deni-de-service.php>.

Androulidakis, Georgios et Papavassiliou, Symeon. (2008). Improving network anomaly detection via selective flow-based sampling. *IET communications*, 2(3), 399–409.

Big Switch Networks. (2017). «Big Switch Networks, Inc». Repéré à <http://www.bigswitch.com/>.

Brocade Communications Systems. (2016). «Weighted Round Robin scheduling». Repéré à <http://www.brocade.com/content/html/en/configuration-guide/nos-700-l2guide/GUID-B75BD370-B251-45A3-B3FD-87F54E35DC6E.html>.

Brocade Communications Systems. (2017). «OpenFlow 1.3». Repéré à <http://www.brocade.com/content/html/en/configuration-guide/nos-601-sdnguide/GUID-FEF22796-0D22-4BDD-96F1-28589E2A0E6B.html>.

BURMAT BIT. (2014). «DoS Attack | BURMA BIT». Repéré à <https://burmabit.wordpress.com/2014/04/22/dos-attack>.

Chin, Tommy, Mountrouidou, Xenia, Li, Xiangyang et Xiong, Kaiqi. (2015). Selective packet inspection to detect dos flooding using software defined networking (SDN). *IEEE International Conference on, Distributed Computing Systems Workshops (ICDCSW)*, pp. 95–99.

Cikala, Frédéric. (2017). «Les IDS : Présentation». Repéré à <http://www-igm.univ-mlv.fr/~dr/XPOSE2004/IDS/IDSPres.html>.

Clarke, Salgueiro et Pignataro. (2016). «RFC 7922 - Interface to the Routing System (I2RS) Traceability : Framework and Information Model». Repéré à <https://tools.ietf.org/html/rfc7922>.

Confluence. (2015). «POX Wiki - Open Networking Lab - Confluence». Repéré à <https://openflow.stanford.edu/display/ONL/POX+Wiki>.

Dao, Nhu-Ngoc, Park, Junho, Park, Minho et Cho, Sungrae. (2015). A feasible method to combat against DDoS attack in SDN network. *International Conference on, Information Networking (ICOIN)*, pp. 309–311.

die.net. (2017). «Hping3 Network tool». Repéré à <http://linux.die.net/man/8/hping3/>.

Dridi, Lobna et Zhani, Mohamed Faten. (2016). SDN-Guard : DoS Attacks Mitigation in SDN Networks. *IEEE International Conference on, Cloud Networking (Cloudnet)*, pp. 212–217.

EFORT. (2016). «Le Protocole OpenFlow dans l'Architecture SDN». Repéré à <http://www.efort.com/r_tutoriels/OPENFLOW_EFORT.pdf>.

Enns, Bjorklund, Schoenwaelder et Bierman. (2011). «RFC 6241 - Network Configuration Protocol (NETCONF)». Repéré à <https://tools.ietf.org/html/rfc6241>.

Giotis, Kostas, Argyropoulos, Christos, Androulidakis, Georgios, Kalogeras, Dimitrios et Maglaris, Vasilis. (2014). Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments. *Computer Networks*, 122–136.

Greg Sowell Consulting. (2013). «OpenFlow And Mikrotik». Repéré à <http://gregsowell.com/?p=4442>.

Guillaume Lehmann. (2003). «Les différents types d'IDS». Repéré à <http://lehmann.free.fr/RapportMain/node10.html>.

Ha, Taejin, Kim, Sunghwan, An, Namwon, Narantuya, Jargalsaikhan, Jeong, Chiwook, Kim, JongWon et Lim, Hyuk. (2016). Suspicious traffic sampling for intrusion detection in software-defined networks. *Computer Networks*, 172–182.

Halpern et Hadi, Salim. (2010). «RFC 5812 Forwarding and Control Element Separation (ForCES) Forwarding Element Model». Repéré à <https://tools.ietf.org/html/rfc5812>.

Jonathan Krier. (2006). «Les systèmes de détection d'intrusions». Repéré à <http://dbprog.developpez.com/securite/ids/#LIV-A-1>.

Kandoi, Rajat et Antikainen, Markku. (2015). Denial-of-service attacks in OpenFlow SDN networks. *IFIP/IEEE International Symposium on, Integrated Network Management (IM)*, pp. 1322–1326.

Kawahara, Ryoichi, Ishibashi, Keisuke, Mori, Tatsuya, Kamiyama, Noriaki, Harada, Shigeaki et Asano, Shoichiro. (2007). Detection accuracy of network anomalies using sampled flow statistics. *IEEE. Global Telecommunications Conference (GLOBECOM)*, pp. 1959–1964.

Linux foundation. (2017). «The OpenDaylight Platform | OpenDaylight» [Format]. Repéré à <https://www.opendaylight.org/>.

Lopez, Martin Andreoni, Mattos, Diogo Menezes Ferrazani et Duarte, Otto Carlos MB. (2016). An elastic intrusion detection system for software networks. *Annals of Telecommunications*, 595–605.

Luc, De Mey. (2017). «La gestion des processus». Repéré à <http://www.courstechinfo.be/OS/Processus.html#Algorithmes>.

Margaret, Rouse. (2013). «What is ternary content-addressable memory (TCAM) ? Definition from WhatIs.com». Repéré à <http://searchnetworking.techtarget.com/definition/TCAM-ternary-content-addressable-memory>.

MediaWiki. (2010). «CS244 : Overview - CS244 Wiki». Repéré à <http://yuba.stanford.edu/cs244wiki/index.php/Overview>.

Mininet Team. (2017). «Mininet Overview». Repéré à <http://mininet.org/overview/>.

Mohammadi, Reza, Javidan, Reza et Conti, Mauro. (2017). SLICOTS : An SDN-Based Lightweight Countermeasure for TCP SYN Flooding Attacks. *IEEE Transactions on Network and Service Management*, 487–497.

Open Information Security Foundation. (2017). «Open Information Security Foundation». Repéré à <https://oisf.net/>.

Open Networking Foundation. (2017a). «Software-Dedined Networking (SDN) Definition». Repéré à <https://www.opennetworking.org/sdn-resources/sdn-definition>.

Open Networking Foundation. (2017b). «OpenFlow-Open Networking Fundation». Repéré à <https://www.opennetworking.org/sdn-resources/openflow>.

Open Networking Fundation. (2012). *OpenFlow Switch Specification version1.3.0*. Open Networking Fundation : Open Networking Fundation.

Open Networking Fundation. (2013). *SDN Architecture Overview version 1.0*. Open Networking Fundation : Open Networking Fundation.

Peter, Mell et Timothy, Grance. (2011). *«The NIST Definition of Cloud Computing : Recommendations of the National Institute of Standards and Technology»*. Computer Security Division Information Technology Laboratory National Institute of Standards and Technology Gaithersburg, MD 20899-8930 : National Institute of Standards and Technology.

Project Floodlight. (2016). «Floodlight controller». Repéré à <http://www.projectfloodlight.org/>.

Rodfoss, Jonas Tafto. (2011). *Comparison of Open Source Network Intrusion Detection Systems*. (Mémoire de maîtrise, Université de OSLO, OSLO-Norvège). Repéré à <https://www.duo.uio.no/bitstream/handle/10852/8951/Rodfoss.pdf?sequence=1&isAllowed=y>.

Sahay, Rishikesh, Blanc, Gregory, Zhang, Zonghua et Debar, Hervé. (2015). Towards autonomic ddos mitigation using software defined networking. *NDSS Workshop on, Security of Emerging Networking Technologies, SENT 2015*, pp. 1–7.

Scott-Hayward, S., O'Callaghan, G. et Sezer, S. (2013). SDN Security : A Survey. *IEEE Conference on, Network Softwarization (NetSoft)*, pp. 1-7.

SDx central. (2017). «What is a Floodlight Controller ?». Repéré à <https://www.sdxcentral.com/sdn/definitions/sdn-controllers/open-source-sdn-controllers/what-is-floodlight-controller/>.

sFlow.org. (2017). «About sFlow Overview sFlow.org». Repéré à <http://www.sflow.org/about/index.php>.

Shin, Seungwon et Gu, Guofei. (2013). Attacking Software-defined Networks : A First Feasibility Study. *ACM SIGCOMM Workshop on, Hot Topics in Software Defined Networking*, pp. 165–166.

Shin, Seungwon, Yegneswaran, Vinod, Porras, Phillip et Gu, Guofei. (2013). AVANT-GUARD : scalable and vigilant switch flow management in software-defined networks. *ACM SIGSAC conference on, Computer & communications security*, pp. 413–424.

Sourceforge. (2017). «lpsolve». Repéré à <http://lpsolve.sourceforge.net/5.5/>.

Suricata. (2015). «Suricata | Open Source IDS / IPS / NSM engine». Repéré à <https://suricata-ids.org/>.

TechTerms. (2017). «API (Application Programming Interface) Definition». Repéré à <https://techterms.com/definition/api>.

The Bro Network Security Monitor. (2017). «The Bro Network Security Monitor». Repéré à <https://www.bro.org/contact/index.html>.

The Snort Team. (2017). «Snort - Network Intrusion Detection Prevention System». Repéré à <https://www.snort.org/>.

Thongkanchorn, Kittikhun, Ngamsuriyaroj, Sudsanguan et Visoottiviseth, Vasaka. (2013). Evaluation studies of three intrusion detection systems under various attacks and rule sets. *TENCON IEEE Region 10 Conference (31194)*, pp. 1–4.

Wang, Haopei, Xu, Lei et Gu, Guofei. (2015). FloodGuard : a dos attack prevention extension in software-defined networks. *45th Annual IEEE/IFIP International Conference on, Dependable Systems and Networks (DSN)*, pp. 239–250.

Wei, Lei et Fung, Carol. (2015). FlowRanger : A request prioritizing algorithm for controller DoS attacks in Software Defined Networks. *IEEE International Conference on, Communications (ICC)*, pp. 5254–5259.

Xing, Tianyi, Huang, Dijiang, Xu, Le, Chung, Chun-Jen et Khatkar, Pankaj. (2013). Snortflow : A openflow-based intrusion prevention system in cloud environment. *Second GENI, Research and Educational Experiment Workshop (GREE)*, pp. 89–92.