



ABDELMALEK ESSAADI UNIVERSITY
NATIONAL SCHOOL OF APPLIED SCIENCES OF TANGIER

Master CyberSecurity and cyberCrime (MCSC)
Department of Mathematics and Computer Sciences

End of Studies Project - PFE

**Architecture dynamique et évolutive pour sécuriser les
réseaux IoT.**

By :

NAJAH ISSAM

Soutenue le : 21/09/2021

Jury members :

Prof Soufiane Mezroui
Prof Mariam Tanana
Prof Said Bouchkaren

Supervisor, ENSA of Tangier
Examiner, ENSA of Tangier
Examiner, ENSA of Tangier

2020/2021

Table des matières

Préambule	1
Remerciements	2
Liste des figures	4
Liste des tables	5
Résumé	6
Introduction générale	7
1 Etat de l'art d'IoT (Internet des Objets)	9
1.1 Introduction	9
1.2 L'Internet des Objets	9
1.3 Historique de l'Internet des objets	10
1.4 Applications IoT	11
1.4.1 Smart Grids	12
1.4.2 Santé	12
1.4.3 Systèmes de transport	12
1.4.4 Villes intelligentes	13
1.4.5 Fabrication	13
1.5 Les avantages de l'Internet des objets	13
1.6 Les enjeux de l'IoT	14
1.7 Architecture de l'IoT	14
1.7.1 La couche perception	15
1.7.2 La couche réseau	15
1.7.3 La couche application	16
1.8 Technologies de communication de l'IoT	16
1.9 Conclusion	17
2 Software Defined Networking (SDN)	18
2.1 Introduction	18
2.2 Le concept du SDN	18
2.3 Différence entre un réseau SDN et un réseau traditionnel	19
2.4 Architecture SDN	20
2.4.1 La couche de transmission	20
2.4.2 La couche de contrôle	21
2.4.3 La couche application	21
2.5 Interfaces de communications	21

2.5.1	La Southbound API	21
2.5.2	Interfaces Nord	21
2.5.3	Interfaces Est/Ouest	21
2.5.4	Protocole OPENFLOW	22
2.6	Mécanisme d'échanges dans un réseau SDN	22
2.7	Conclusion	24
3	LA SÉCURITÉ IOT	25
3.1	Introduction	25
3.2	Défis IoT	25
3.3	Les objectifs de sécurité	26
3.3.1	Confidentialité	26
3.3.2	Intégrité	26
3.3.3	Disponibilité	27
3.3.4	Authentification	27
3.3.5	Solutions légères	27
3.3.6	Hétérogénéité	27
3.3.7	Politiques	28
3.3.8	Systèmes de gestion de clés	28
3.4	Vecteur de menace	29
3.4.1	Attaques de communication	30
3.4.2	Attaques physiques	30
3.4.3	Attaques applicatives/logicielles Vecteur de menace	31
3.5	Problèmes de sécurité dans chaque couche de l'IoT	32
3.5.1	Couche de perception	32
3.5.2	Couche réseau	33
3.5.3	Couche d'application	33
3.6	Mécanismes de sécurité IOT	34
3.6.1	La protection de la vie privée des utilisateurs et La gestion de confiance	34
3.6.2	La gestion des clés	34
3.6.3	La détection d'intrusion	34
3.6.4	Le contrôle d'accès aux services du IoT	34
3.6.5	Solutions basées sur le Fog Computing	35
3.6.6	Solutions basées sur la Blockchain	35
3.6.7	Solutions de chiffrement homomorphiques et interrogeables	37
3.6.8	Solutions légères basées sur la cryptographie	37
3.6.9	Software defined networking (SDN)	38
3.7	Conclusion	39
4	L'architecture proposée	40
4.1	Firewall Intelligent SDN	41
4.2	Analyse, détection et génération des alertes (IDS/IPS)	43
4.3	Sécurisation du lien entre le contrôleur OpenDaylight et l'OVS	43
4.4	Authentification IoT (Lightweight Cryptographie)	44
4.4.1	Poly1305-AES	44
4.4.2	CMAC	44
4.4.3	HMAC	44
4.4.4	Comparaisons	45
4.5	Chiffrement IoT (Lightweight Cryptographie)	46

4.5.1	Advanced Encryption Standard	46
4.5.2	Data Encryption Standard	47
4.5.3	Blowfish	48
4.5.4	Comparaisons	50
4.6	Conclusion	51
5	Implémentation	52
5.1	Introduction	52
5.2	Les outils utilisés	52
5.2.1	Le contrôleur OpenDaylight	52
5.2.2	Open vSwitch (OVS) :	54
5.2.3	Système d'exploitation :	54
5.2.4	Surveillance de trafic et gestion de flux malveillants :	55
5.3	Implémentation	57
5.3.1	Installation d'OpenDaylight	57
5.3.2	Installation d'un commutateur virtuel OpenFlow (OVS)	60
5.3.3	Implémentation d'un pare-feu basé sur SDN	61
5.3.4	Mise en place de Snort	62
5.3.5	La sécurité du lien entre le contrôleur OpenDaylight et l'OVS	64
5.3.6	L'authentification et le chiffrement par poly1305 et AES-128	65
5.4	Quelques exemples d'attaques simulées	69
5.4.1	Déni de services	69
5.4.2	Scan de ports	69
5.4.3	Usurpation d'adresse IP ou MAC	70
5.5	Conclusion	70
	Conclusion et Perspectives	71
	Liste des acronymes	73
	Références	75

Préambule

Nom et prénom :

Najah Issam

Title du projet :

Architecture dynamique et évolutive pour sécuriser les réseaux IoT.

Établissement d'accueil :

ENSA TANGER

Superviseur académique :

SOUFIANE MEZROUI

Remerciements

Je tiens d'abord à remercier le Bon Dieu pour me munir du courage, de la force et de la patience tout au long de mon parcours.

J'exprime mes sincères remerciements et toute ma gratitude au Professeur Mezroui Soufiane pour son excellente qualité d'encadrement et pour les orientations et les consignes pertinentes qu'il m'a accordées, et qui m'ont été très utiles durant les différentes phases de réalisation de projet. Je le remercie également pour sa précieuse disponibilité et pour me permettre de partager ses connaissances et sa grande expérience dans mon domaine de recherche.

Je tiens à exprimer, aussi, mes plus vifs remerciements au professeur Lazaar Saida qui m'a soutenu tout au long de mon cursus au master CSC et pour l'intérêt qu'elle a porté à mon travail.

Je voudrais remercier mes chers parents pour leur accompagnement et leurs encouragements permanents. Ils n'ont préservé aucun effort pour m'aider à mener à succès mes études.

J'adresse toute ma gratitude à Hicham Ziani et pour leur soutien indéfectible, ainsi qu'à tous mes ami(e)s et toutes les personnes qui ont contribué de loin ou de près à la réalisation de ce travail.

Je remercie sincèrement tous les gens honnêtes et justes dont j'ai eu l'honneur de connaître dans ma vie. J'espère qu'ils se reconnaissent en lisant ces mots.

Table des figures

1.1	Architecture d'Internet des Objets	15
1.2	Catégories des technologies de communication	16
2.1	Comparaison entre un réseau traditionnel et un réseau SDN.	20
2.2	Architecture détaillée d'un réseau SDN.	20
2.3	Processus de transmission d'un paquet avec openflow.	22
2.4	Mécanisme d'échanges de flux dans un réseau OpenFlow.	23
2.5	Messages OpenFlow échangés entre le contrôleur SDN et l'OVS.	23
3.1	Structure de blockchain.	35
3.2	Validation de transaction.	36
3.3	Architecture Blockchain pour IoT.	36
3.4	Cryptographie légère pour l'IoT.	38
4.1	Architecture propose pour sécuriser réseau IoT	41
4.2	Pare-feu traditionnel.	42
4.3	Pare-feu basé sur SDN.	42
4.4	Temps d'exécution pendant la génération mac.	45
4.5	Temps d'exécution pendant la validation du mac.	46
4.6	Fonctionnement d'AES	47
4.7	Fonctionnement de DES.	48
4.8	Fonctionnement de Blowfish.	49
4.9	Temps d'exécution pendant le chiffrement.	50
4.10	Temps d'exécution pendant le déchiffrement.	50
5.1	Architecture simplifier d'OpenDaylight.	53
5.2	Maquette de mise en œuvre de sécurisation d'un réseau avec du SDN.	57
5.3	Installation de JAVA 8 JRE.	57
5.4	La sélection du version JAVA par défaut	58
5.5	Le chemin complet de votre exécutable JAVA.	58
5.6	La valeur du JAVA HOME.	58
5.7	Vérification de valeur du JAVA HOME.	58
5.8	Téléchargement OpenDaylight version NEXUS.	58
5.9	Ecran de lancement OpenDaylight.	59
5.10	Interface graphique d'OpenDaylight NEXUS.	60

5.11	Installation de KVM.	60
5.12	Installation de OVS.	60
5.13	Vérification et comparaison des versions des packages.	61
5.14	Installation de SNORT.	62
5.15	L'interface sur laquelle l'outil écouter le réseau.	62
5.16	Décompression des règles présentes sur le site de Snort.	63
5.17	Déplacer le fichier "community.rules".	63
5.18	Fichier /etc/snort/snort.conf	63
5.19	Fonctionnement des alertes	63
5.20	Fonctionnement des alertes	64
5.21	Exemple d'un fichier logs.	64
5.22	Fichier de configuration OpenFlow pour supporter TLS.	65
5.23	Fichier générer par capteur cardiofréquencemètre.	66
5.24	Le script python pour la génération d'une balise MAC	66
5.25	La génération d'une balise MAC avec nonce.	67
5.26	Le script python pour la validation de MAC.	67
5.27	Le fichier Mac générer.	68
5.28	La validation de Mac.	68
5.29	Le cryptage de fichier avec la commande openssl (AES-128).	68
5.30	Le cryptage de fichier avec la commande openssl (AES-128).	68
5.31	Le décryptage de fichier avec la commande openssl (AES-128)	68
5.32	Le contenu de fichier après le décryptage.	69

Liste des tableaux

3.1	Services et mécanismes de sécurité.	29
3.2	Domaines de sécurité et contrôles de sécurité pour l'IOT.	30

Résumé

L'Internet des objets (IoT) est une vague émergente d'Internet qui a beaucoup attiré l'attention ces dernières années. Cela a radicalement changé la vie de la communauté ou les objets dans l'IoT peuvent prendre une grande variété de formes, des objets minuscules aux grands objets. Le déploiement explosif de l'IoT est confronté à des problèmes de sécurité et de confidentialité, qui présenteront des risques importants qui vont de pair avec les avantages potentiels de l'IoT. La sécurisation de tels systèmes pose de nombreux défis, en particulier dans les environnements à ressources limitées, hétérogènes et à grande échelle. L'objectif principal de ce projet est de surmonter les problèmes de sécurité et de confidentialité entourant l'IoT à différentes couches. Dans ce contexte, nous proposons une nouvelle architecture pour sécuriser IoT basée SDN (Software Defined Networking) et des autres mécanismes de sécurité (IDS/IPS, Par-feu intelligent et l'authentification et chiffrement légère) efficaces et robustes pour les systèmes IoT afin de contrecarrer les attaques du monde physique et du cyberspace. Nous évaluons les performances et la sécurité de nos solutions proposées. Les résultats obtenus montrent que nos techniques proposées sont sécurisées, efficaces et adaptées aux systèmes IoT par rapport aux méthodes connexes récentes.

Mots clés : IoT, SDN, Sécurité, IDS/IPS, Par-feu intelligent, réseaux de capteurs sans fil, Authentification, Confidentialité.

Introduction générale

La vision d'un monde intelligent prévoit que l'ensemble de l'espace urbain soit interconnecté afin qu'il puisse interagir avec le monde numérique et par conséquent l'adoption du paradigme de l'Internet des Objets (IoT : Internet of Things). En effet, l'IoT permet d'interconnecter différents objets, machines, capteurs qui peuvent être portés par des êtres humains afin d'offrir une nouvelle gamme de services et d'applications. En interconnectant différents types d'objets grâce à de nouvelles technologies adaptées et en utilisant les capacités offertes par le cloud computing, l'IoT permet de faciliter la vie humaine en automatisant certaines tâches quotidiennes. De même, l'IoT permet d'offrir des services critiques à distance comme les services médicaux en utilisant les capacités du réseau Internet. Ainsi, l'impact de l'IoT sur nos sociétés sera important et permettra l'amélioration de la qualité de vie tout en réduisant les charges de mise en place des services. A titre d'exemple, la prise en charge des services de santé à travers l'IoT permet de réduire les charges qui incombent aux collectivités tout en améliorant le suivi des patients.

En 2030, il est annoncé plus de 500 milliards d'équipements connectés à Internet avec une variété d'usage entraînant des problèmes de sécurité et une augmentation du trafic sur les réseaux qui seront estimée en Zeta octets. Or, actuellement, les architectures de sécurité déployées dans les réseaux sont principalement issues de l'expérience et des travaux sur les réseaux filaires. Ces architectures sont principalement basées sur des équipements centralisés, dont le rôle principal est de contrôler les informations qui sont échangées entre le réseau de l'entreprise et l'extérieur. Il n'est donc pas possible de contrôler les informations échangées entre un équipement terminal qu'un utilisateur va venir connecter sur son ordinateur. L'exemple du téléphone est un des cas les plus problématiques. Sur un réseau d'entreprise, les utilisateurs peuvent connecter leur téléphone sur leur ordinateur, via du Bluetooth par exemple et ainsi, l'ordinateur devient une nouvelle porte d'entrée sur le réseau. Avec l'Internet des Objets où Internet of Things (IoT), nous avons des capteurs, des thermostats, des webcams, des montres connectées à nos téléphones, eux-mêmes éventuellement connectés à Internet ou à nos ordinateurs. Comment pouvons-nous donc effectuer le contrôle des informations qui proviennent de cette grande masse d'équipements hétérogènes ? Avec l'augmentation du nombre de ces équipements hétérogènes, la complexité dans leur administration devient croissante. Ce qui nécessite une vérification de la cohérence des configurations de tous les équipements réseaux d'une entreprise, par exemple les règles de sécurité et les droits utilisateurs.

Depuis ces dernières années, il se développe un nouveau concept de gestion de ré-

seau appelé Software Defined Networking(SDN). Son objectif principal est de simplifier la gestion et la configuration des réseaux et de faciliter l'innovation. Son principe est la séparation du plan de contrôle des équipements réseaux (commutateurs, routeurs...) du plan de données et le placer vers un point de contrôle centralisé. Appelé contrôleur SDN, ce point de contrôle centralisé permet de programmer les équipements réseaux à travers des interfaces programmables. Il permet également d'avoir une vue globale du réseau. Avec le SDN, les administrateurs réseaux n'ont plus à configurer chaque équipement séparément comme dans les réseaux traditionnels où le risque d'erreur est élevé. Du point de vue sécurité, le SDN permet de programmer des politiques de sécurité et de pousser sur les équipements réseaux grâce au contrôleur SDN. Le SDN présente aussi des avantages multiples, entre autres : la programmabilité, l'évolutivité, la flexibilité, la scalabilité, la réduction des coûts d'investissement et d'exploitation des réseaux, mais soulève également des préoccupations liées à la centralisation du contrôle sur un équipement unique, qui devient un point critique et l'introduction des nouvelles interfaces vulnérables aux attaques.

Dans le cadre de ce travail de projet, notre objectif est d'étudier et de réaliser une nouvelle solution pour sécuriser les échanges dans un réseau IOT en exploitant le concept du SDN et des autres mécanismes (IDS/IPS ,FIREWALL , chiffrement et authentification ...).

Chapitre 1

Etat de l'art d'IoT (Internet des Objets)

1.1 Introduction

Internet of Things (IoT), et dans un sens plus large, Internet of Everything (IoE) est un concept relativement récent. Il est considéré comme une innovation technologique et économique majeure dans l'industrie des nouvelles technologies de l'information et de la communication. L'IoT n'a pas une définition unique mais d'une manière générale, il est défini comme étant une extension de l'Internet actuel à tous les objets pouvant communiquer de manière directe ou indirecte avec des équipements électroniques, eux-mêmes connectés à l'Internet.

Ce chapitre introduit l'Internet des Objets (IoT), Historique de l'Internet des objets, Applications IoT et les domaines auxquels nous sommes confrontés sur notre vie quotidienne, Les avantages d'utilisation de l'Internet des objets, notamment les fonctionnalités des technologies d'IoT et son architecture et ses protocoles.

1.2 L'Internet des Objets

L'Internet des Objets (IoT) prend depuis quelque temps une ampleur de plus en plus importante. Il concerne, avec des frontières plus ou moins floues, la mise en connectivité massive d'objets, tel que des capteurs, téléphones, ou plus généralement d'objets du quotidien auparavant déconnectés (serrure, climatiseur, etc.) [1]. Aujourd'hui encore peu présents dans nos domiciles, certaines sources indiquent que le parc de ces objets pourrait atteindre facilement les 24 milliards d'ici 2024 [2]. Si l'intérêt repose généralement sur la mise en place de services avancés pour les différents utilisateurs, ces nouveaux objets fragilisent dangereusement la sécurité de nos domiciles, à la fois numériquement mais également physiquement.

L'IoT, appelé également Web 3.0, représente une extension d'Internet à des choses et à des lieux du monde physique. Chaque objet physique qui peut être une personne, un ordinateur, un smartphone, une voiture, un capteur, une maison intelligente... est associé

à une entité virtuelle qui se comporte comme une entité active dans le système. Selon IBM, neuf milliards d'objets sont aujourd'hui connectés, et le nombre augmente sans cesse et très rapidement. La réussite de ce nouveau paradigme s'explique [3] par la variété des équipements (objets) qu'on utilise dans notre vie quotidienne et leur développement. En IoT, chaque objet doit avoir les caractéristiques suivantes :

L'existence : est le fait d'avoir une réalité et d'avoir une présence en un lieu. Un objet tel qu'un véhicule existe dans le monde physique, mais grâce à un équipement de communication intégré et des technologies spécifiques associées, le véhicule possède également une identité (une existence / une réalité) dans le monde virtuel.

l'autonomie : est la capacité d'un objet à se contrôler soi-même. Elle représente les propriétés d'une entité qui est capable de fonctionner de manière indépendante et sans être commandé de l'extérieur. En effet, chaque objet possède une identité lui représentant, peut librement rassembler, analyser, traiter, générer et échanger des informations. Il peut également prendre des décisions sans aucune intervention humaine.

La connectivité : c'est à dire ce qu'une entité offre comme connexion à d'autres entités de son environnement. Ainsi, n'importe quel objet autorisé peut initier une communication avec d'autres objets et utiliser leurs informations.

L'interactivité et l'interopérabilité : l'internet des objets est un concept général qui en globe différents domaines utilisant des systèmes, architectures et matériels différents. Par conséquent, un objet doit pouvoir interagir et collaborer avec d'autres objets hétérogènes. Ces derniers peuvent être des humains ou des machines, réels ou virtuels, qui produisent et consomment une grande variété de services.

La flexibilité : un objet peut interagir avec d'autres objets à n'importe quel moment (Any time), n'importe où (Any where), et n'importe comment (Any how), et fait des calculs pour n'importe quel objet (Anything), pour n'importe qui (Anyone), et pour n'importe quel service (Any service).

Ce succès s'explique aussi [4] par l'évolution des technologies de communication, notamment celle qui sont sans fils. Ces technologies sont devenues plus fiables, plus optimales, et moins coûteuses en termes de calcul, de mémorisation et de consommation d'énergie.

Cependant, malgré ce succès, à cause de la menace des cyberattaques, la sécurité de l'IoT reste encore un des problèmes majeurs qui freine l'évolution et le déploiement rapide de cette technologie de technologies.

1.3 Historique de l'Internet des objets

Le terme Internet des objets (IoT : Internet of Things) remonte à la fin des années 90 lorsque Kevin Ashton, cofondateur du centre Auto-ID à Massachusetts , l'a utilisé pour définir le lien entre la technologie RFID et l'Internet . Auto-ID est un groupe de laboratoires de recherche travaillant sur les technologies de radiofréquence et les technologies de capteurs émergentes. Selon Ashton, l'IoT n'est pas une vision mais un nouvel environnement capable de collecter des informations indépendamment de l'interaction humaine et de comprendre le monde sans être opéré par les humains [39].

En 2000, la notion d'objets connectés a été abordée pour la première fois par les industriels suite aux expérimentations du fabricant coréen LG (Life's Good) permettant d'interconnecter des appareils de la vie quotidienne à travers l'Internet. Ainsi, LG a annoncé le plan « réfrigérateur interconnecté » en 2000 . Ces expérimentations ont été effectuées bien avant que l'alliance IPSO (Internet Protocol for Smart Objects), alliance regroupant les grands industriels technologiques, promeut l'utilisation du protocole IP (Internet Protocol) pour l'interconnexion des objets en 2008. En revanche, Cisco IBSG (Internet Business Solutions Group) a indiqué que l'IoT est né entre 2008 et 2009, simplement au moment où plus d'objets que d'humains étaient connectés à l'Internet tout en citant la croissance massive du nombre d'appareils connectés. Par ailleurs, l'IoT a été ajouté en 2011 au Gartner Hype Cycle, organisme de recherche et de conseil international, qui suit les cycles de vie des technologies du moment du déclenchement à la production. De plus, l'IoT a atteint le pic des attentes exagérées (Peak of Inflated Expectations) du cycle Gartner Hype en 2014 . Depuis son lancement, l'IoT a fait l'objet d'une immense évolution et il a été considéré par plusieurs organismes de standardisation mais aussi par différentes commissions européennes et mondiales. En effet, l'Union Internationale des Télécommunications – secteur de la normalisation des Télécommunications (UIT-T) a proposé en 2005 un rapport portant sur l'IoT nommé “The Internet of Things” . En outre, la première conférence internationale sur le sujet de l'IoT a pris place à Zurich en 2008 sous le nom « Internet of Things » . En même temps, le NIC (National Intelligence Council) a énuméré l'IoT comme l'une des six technologies civiles qui ont des répercussions potentielles sur les intérêts américains jusqu'en 2025. Par la suite, l'UIT-T a proposé le document Y.2060 présentant une vue d'ensemble de l'IoT en 2012 et des recommandations pour l'environnement IoT à travers le document Y.2066 en 2014 . D'autres organismes de standardisation se sont intéressés à l'environnement IoT ainsi que les technologies associées. Ainsi, l'IETF (Internet Engineering Task Force) a créé le groupe de travail 6Lo en 2014, précédé par le groupe de travail 6LoWPAN , pour adapter les technologies radio à l'environnement IoT. De plus, cet organisme propose plusieurs documents de références décrivant les challenges à considérer dans l'IoT [39].

De nos jours, plus de 26 milliards d'objets connectés existent dans le monde pour offrir différents services de l'IoT dans les différents domaines d'application de ce nouveau paradigme. D'ici 2020, les estimations indiquent que plus de 30 milliards d'objets seront connectés .

1.4 Applications IoT

L'IoT couvre un large éventail d'applications et touchera presque tous les domaines auxquels nous sommes confrontés sur notre vie quotidienne. Parmi ces applications, on peut citer les suivantes :

1.4.1 Smart Grids

L'énergie électrique est un trésor à très haute valeur industrielle et joue un rôle important dans le développement économique. De nos jours, nous utilisons une informatique très moderne technologies pour optimiser la production d'électricité en tenant compte des demandes des utilisateurs tout au long de la ligne de distribution d'électricité. Smart Grids est la technologie derrière cette ligne de distribution. Il se compose d'un réseau intégré, appelé aussi l'infrastructure de comptage (AMI) installée entre les centres de production d'électricité et les clients finaux, dont le rôle important est de coordonner l'électricité production par rapport à la consommation des clients finaux. Smart Grids représentent l'un des domaines les plus attractifs de l'IoT. L'objectif principal est d'améliorer la qualité dès l'expérience des clients finaux et optimiser la production d'électricité [40].

1.4.2 Santé

Les soins de santé intelligents jouent un rôle important dans les applications de soins de santé en intégrant des capteurs et des actionneurs dans le corps des patients à des fins de surveillance et de suivi. L'IoT est utilisé dans le domaine de la santé afin de surveiller l'état physiologique des patients. Les capteurs embarqués ont la capacité de collecter des informations directement à partir du zone du corps du patient et le transmettre au médecin. Cette technologie a la possibilité de détacher complètement le patient du système centralisé qui est l'hôpital tout en maintenant un contact permanent avec le médecin. Actuellement, Les applications IoT basées sur la santé représentent l'une des technologies prometteuses qui impactent énormément la société, principalement en raison du vieillissement de la population. En effet, en France, le pourcentage de personnes de plus de 60 ans atteint environ 24% des populations en 2015 et passera à 32% d'ici 2060 De plus, le budget réservé pour les applications de santé a atteint environ 12% du PIB (produit intérieur brut). Dans ce contexte de vieillissement de la population et de coût lié au traitement, un grand un intérêt émerge pour adopter de nouvelles technologies basées sur l'IoT pour suivre les patients en réel temps [40].

1.4.3 Systèmes de transport

Les systèmes de transport intelligents (STI) représentent la prochaine génération de transport qui vise à relier les personnes, les routes et les véhicules intelligents grâce au développement de systèmes embarqués et de technologies de communication. En vous connectant et la distribution de processeurs intelligents à l'intérieur des véhicules et également par le biais du transport infrastructure, nous pouvons rendre le transport plus sûr, plus vert et plus pratique. ITS emploie quatre composants principaux, à savoir : le sous-système du véhicule (comprend le GPS, Lecteur RFID, OBU et communication), sous-système de station (équipement routier), Centre de surveillance des STI et sous-système de sécurité. Les véhicules connectés deviennent plus importants dans le but de rendre la conduite plus fiable, agréable et efficace. En fait, nous avons trois types de communications dans les réseaux de véhicules : V2V (Vehicle to Vehicle), V2I (Vehicle to Infrastructure)

et V2P (Vehicle to Pedestrian). Cependant, récemment, un nouveau type de communication est apparu, appelé V2G (Vehicle to Grid), dont l'objectif principal est d'assurer la recharge des véhicules électriques basée sur énergie de distribution d'électricité sur un réseau intelligent [40].

1.4.4 Villes intelligentes

Les villes intelligentes sont l'une des applications les plus importantes de l'IoT. Même si, il n'y a pas de définition formelle de la « ville intelligente », il s'agit d'un nouveau paradigme émergent qui vise à améliorer l'utilisation des ressources publiques, à augmenter la qualité de service pour citoyens. Dans ce contexte, des capteurs sont déployés partout sur les routes, les bâtiments, les voitures, etc. pour mieux gérer la circulation, s'adapter à la météo, l'éclairage suit la position du soleil, les incidents domestiques peuvent être évités avec des alarmes, etc [40].

1.4.5 Fabrication

De nos jours, l'IoT joue un rôle important dans l'industrie. Il est considéré comme une solution prometteuse pour automatiser le processus de fabrication et le contrôle de la chaîne de production. L'Internet des objets industriel (IIoT) utilise de nouvelles technologies telles que comme la communication Machine-to-Machine (M2M), les réseaux de capteurs sans fil (WSN), les technologies d'automatisation ainsi que le Big Data pour créer une industrie intelligente écosystème. L'objectif principal de l'IIoT est de fournir une meilleure productivité, efficacité, fiabilité et meilleur contrôle des produits finis [40].

1.5 Les avantages de l'Internet des objets

Nous avons cité dispersément dans différentes parties du présent chapitre quelques avantages de l'Internet des objets. Dans cette section, nous résumons les principaux avantages de l'IoT :

- Accès ubiquitaire à l'information pour un monde plus intelligent et un mode vie sophistiqué et confortable.
- Amélioration de la qualité de service et de la télésurveillance dans différents domaines d'applications, à savoir le domaine industriel, médical, etc.
- Améliorer la productivité et l'expérience-client : les objets connectés envoient des rapports à leurs constructeurs indiquant les préférences et les habitudes des clients aidant davantage les entreprises à agir de manière proactive et adaptée qui satisfait la demande et les exigences de la clientèle.
- Le gain du temps est un autre avantage de l'IoT. Les déplacements inutiles sont dès lors remplacés par une simple navigation sur le web pour commander des produits, contrôler l'état des objets et/ou endroits connectés.
- Dans certaines applications, l'IoT nous permettons même de rationaliser nos dépenses et faire des économies car on ne consomme qu'en cas de besoin, que ça soit pour

les achats ou la consommation énergétique (nécessaire pour l'éclairage ou la climatisation) ou autre.

- Possibilité d'exploitation des ressources géantes de l'Internet pour le stockage et le traitement des données écoulées de l'IoT.

1.6 Les enjeux de l'IoT

De l'objet qui collecte l'information au réseau qui transmet les données à la plateforme qui agrège et traite les données, les enjeux de l'IoT sont nombreux. Il y a deux tendances dans l'approche même de l'IoT, d'un côté, celle de EPCglobal (une organisation d'adoption et de standardisation à l'échelle mondiale de la technologie du code de produit électronique) qui soutient les solutions RFID comme la base de l'IoT, et de l'autre, IPSO Alliance (Internet Protocol for Smart Objects), acteurs actuels de l'Internet et pour qui, si un objet ne communique pas le protocole IP, il n'a pas sa place dans l'IoT. Il y a aussi les solutions M2M (Machine-to-Machine) qui s'appuient sur des concepts relativement différents, mais qui se réclament de l'Internet des Objets. La variété de ces éléments du réseau est un grand challenge en matière d'architecture et de sécurité pour l'Internet du futur et pose des questions assez diverses. En plus, le nombre des objets connectés augmentent chaque jour et devrait atteindre les 500 milliards d'ici à 2030 selon les estimations de Cisco 2, ce qui entrainera de facto une augmentation du trafic sur les réseaux qui sera estimé en Zeta (1021) octets. Cela suppose qu'il faut plus d'équipements réseaux et d'outils applicatifs pour gérer ces réseaux IoT. C'est pourquoi on constate le développement des réseaux dédiées aux objets connectés et l'existence d'une multitude de protocoles de communication tels que NFC (Near Field Communication), RFID, Bluetooth LE (Low Energy), SigFox, Long Rang (LoRa), réseaux cellulaires, etc. avec chacun ses propres caractéristiques et contrainte [41].

L'avènement de l'IoT a engendré un nouveau marché permettant de créer des emplois et des métiers d'aide à la croissance des entreprises. Selon la GSM Association (GSMA), l'IoT est une industrie en plein essor tant sur les aspects matériels que logiciels et devrait rapporter aux opérateurs mobiles un revenu autour de 1200 milliards de dollars à l'horizon 2020 [41].

Il y a certainement de nombreux enjeux socio-économiques qui touchent tous les domaines de la vie quotidienne (santé, éducation, agriculture...) mais nous nous focaliserons sur les enjeux techniques, notamment d'architecture et de sécurité. C'est ce que nous allons évoquer dans la prochaine section.

1.7 Architecture de l'IoT

Les appareils IoT ont une architecture unique qui peut être définie en couches. Semblable à TCP / IP, les appareils IoT sont considérés d'avoir trois couches opérationnelles différentes : le couche de perception, couche réseau et couche application. Chaque couche offre des fonctionnalités différentes ; par conséquent, chaque couche a ses propres

menaces uniques. De plus, chaque couche est connectée et s'appuie sur les autres couches pour fonctionner. Par conséquent, pour l'une de ces couches pour être sécurisée, les autres couches doivent également être sécurisée [5].

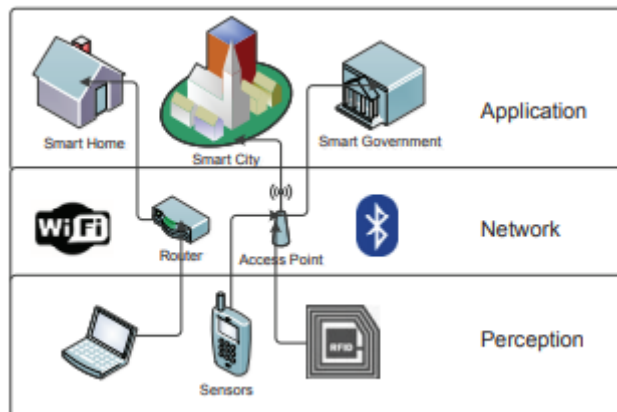


FIGURE 1.1 – Architecture d'Internet des Objets

1.7.1 La couche perception

Cette couche est constituée d'objets physiques composés de capteurs, de dispositifs RFID, de mobiles etc. Le rôle principal de la couche perception est essentiellement d'identifier et de collecter les informations spécifiques aux objets. Selon le type d'objets physiques, les informations peuvent être des informations de type localisation, humidité, température, qualité de l'air, etc. Ainsi les informations collectées sont ensuite transmises à la couche réseau pour leurs transmissions via des canaux sécurisés. Des mécanismes plug-and-play normalisés doivent être utilisés pour configurer les objets physiques hétérogènes. La grande quantité de données générées par l'IoT provient de cette couche, Le cout des objets connectés et leur autonomie sont les contraintes de cette couche, en dehors des données massives qu'ils génèrent [5].

1.7.2 La couche réseau

La couche réseau est responsable de la transmission des données vers la couche perception et vers la couche application. Pour assurer le transfert de données, la couche réseau utilise différents types de technologies de mise en réseau et de protocoles. Ces solutions de connexions réseaux sont, d'une part, les technologies déjà largement déployées, capables de transporter de gros volumes d'informations, mais gourmandes en énergie, telles que Bluetooth, WiFi, 3G, 4G et LTE, et d'autre part, les technologies conçues spécifiquement pour le monde de l'IoT mais encore moins d'éployées comme les ZigBee, LoRa, Sigfox etc. Les réseaux de type SigFox et LoRa permettent de connecter des objets sur une dizaine de kilomètres mais avec des débits très faibles de l'ordre de quelques octets. La WiFi et le Bluetooth ont une portée limitée mais très utile dans un environnement restreint et pourvue de sources d'énergie (une usine par exemple). L'avantage de la 2G/3G/4G est sa disponibilité. Elle peut servir par exemple pour les véhicules connectés. Les passerelles

font partie de cette couche réseau et sont responsables de la transmission des données., la passerelle IoT est l'un des composants le plus important de l'architecture IoT et sert de pont entre la couche de perception et la couche réseau. Il doit être suffisamment flexible pour gérer les ressources disponibles et connecter le système IoT local au reste du système IoT global. La passerelle IoT se heurte à de nombreux problèmes tels que l'hétérogénéité des différents protocoles et composants de l'IoT. Les contraintes liées à cette couche réseau sont la couverture, le cout et la fiabilité [5].

1.7.3 La couche application

La couche application permet l'interaction directe avec les utilisateurs finaux. Les applications peuvent être déployées pour différents domaines de la vie quotidienne tels que la surveillance sanitaire, l'agriculture intelligente, le transport intelligent, les maisons intelligentes, la surveillance routière, l'industrie intelligente etc. Elle comprend également des infrastructures serveur et cloud partageant du contenu et fournissant des services en temps réel. Le traitement des données et la fourniture de services sont deux des fonctions les plus importantes de cette couche. La couche application fournit une gestion globale du système IoT. Il revoit des informations de la couche réseau qui permet aux développeurs de créer diverses applications en utilisant une abstraction et des interfaces ouvertes et de haut niveau. Et c'est dans cette couche que toutes les décisions de contrôle, de sécurité et de gestion des applications sont prises [5].

1.8 Technologies de communication de l'IoT

Selon la portée, le débit, la consommation d'énergie, et le domaine d'application, on peut classer ces technologies en 4 grandes catégories :

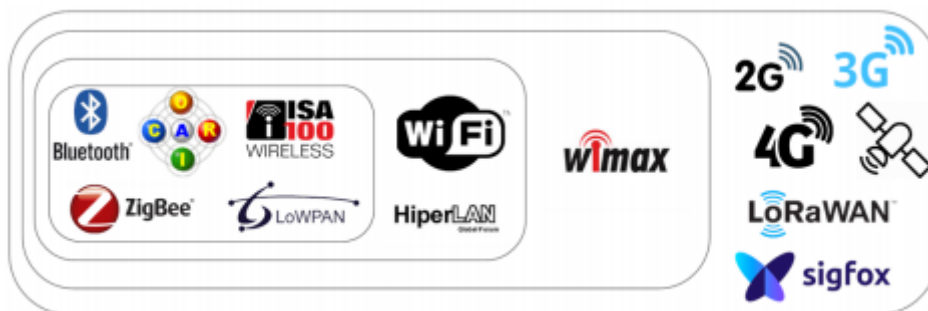


FIGURE 1.2 – Catégories des technologies de communication

Réseaux étendus sans fil (WWAN) : ces réseaux sont considérés comme étant les réseaux les plus étendus. Ils représentent généralement les réseaux à liaisons sans fil à faible consommation énergétique (Low Power Wide Area Network (LPWAN)) tel que LoRaWAN et Sigfox (ayant un débit théorique de 0.3 Kbi t s/S jusqu'à 50 Kbi t s/S), et les réseaux cellulaires tel que GSM, UMTS, et LTE. Ils peuvent avoir un débit théorique de 2 Mbi t s/S (cas de la 2G) jusqu'à 100 Mbi t s (cas de la 4G*). Les WWANs incluent aussi

les réseaux satellitaires tel que le système mondial de positionnement (Global Positioning System (GPS)).

Réseaux métropolitains sans fil (WMAN) : sont basés sur la norme IEEE 802.16. Cette catégorie peut avoir une portée de 4 à 10 Km et offre un débit utile de 1 à 70 Mbit/s. La technologie WMAN la plus connue est WiMax.

Réseaux locaux sans fil (WLAN) : ces réseaux ont une portée d'une centaine de mètres, soit l'équivalent d'un local d'entreprise. En théorie, ils peuvent fournir un débit de plus de 50 Mbits/s. Les technologies les plus connus sont Wi-Fi et High Performance radio LAN (HiperLAN) [50], qui suivent la norme IEEE 802.11.

Réseaux personnels sans fil (WPAN) : concernent les réseaux sans fil à faible portée, de l'ordre de quelques dizaines de mètres. Tout comme la portée qui varie d'une technologie WPAN à une autre, le débit varie aussi. Ce dernier peut être à 250 Kbit/s (ZigBee) jusqu'à 1 Mbit/s (cas du Bluetooth). Ces technologies suivent la famille IEEE 802.15, les plus connues celles de la sous norme IEEE 802.15.1 (Bluetooth), et celles qui sont utilisées dans le domaine des réseaux de capteurs sans fil (WSN pour Wireless Sensor Networks) qui suivent principalement la sous norme IEEE 802.15.4 tel que ZigBee, OCARI, ISA100, 6LoWPAN, etc. Ces technologies sont connues par leurs optimalités en énergie et résistances aux interférences dans les zones industrielles.

Les réseaux sans fils représentent des mécanismes de transport de données entre objets, et entre objets et réseaux filaires classiques. Ces technologies servent à recevoir et transmettre des informations à l'aide d'ondes électromagnétiques. Les réseaux sans fil permettent aux périphériques d'être déplacés à différents degrés de liberté tout en conservant la communication entre eux. Ils offrent également une plus grande flexibilité que les réseaux câblés et réduisent considérablement le temps et les ressources nécessaires pour mettre en place de nouveaux réseaux et facilite la création, la modification ou la démolition des réseaux.

1.9 Conclusion

Dans ce chapitre, nous avons introduit l'Internet des Objets, Historique de l'Internet des objets, Applications IoT et les domaines auxquels nous sommes confrontés sur notre vie quotidienne, Les avantages d'utilisation de l'Internet des objets, aborde de manière succincte ses enjeux, son architecture et les technologies utilisées pour les échanges entre les dispositifs IoT pour mieux comprendre l'approche globale.

Chapitre 2

Software Defined Networking (SDN)

2.1 Introduction

Le SDN – Software-Defined Networking – est certainement le sujet chaud qui agite le monde du réseau depuis ces dernières années. Le SDN est reconnu aujourd’hui comme une architecture permettant d’ouvrir le réseau aux applications. On est donc loin de la définition rigide initiale dans laquelle il était seulement question de dissocier les plans de contrôle et de données. Openflow n’est donc qu’une composante du SDN et le marché semble aujourd’hui davantage réfléchir sur les solutions offrant réellement la programmation et la simplification des infrastructures. A ce niveau plusieurs modèles de SDN se développent en parallèle pour mieux répondre à des usages spécifiques. Les travaux portent globalement sur la programmabilité intrinsèque des équipements (nouveaux composants programmables...), les contrôleurs SDN et orchestrateurs qui ont pour mission de fournir une couche d’abstraction du réseau, et la virtualisation des fonctions réseau qui vise à s’affranchir partiellement de la complexité du réseau physique sous-jacent. Le Software Defined Networking offrira de nouveaux usages. L’enjeu pour les administrateurs réseau est d’accompagner cette nouvelle étape afin de pouvoir tirer profit de ces nouvelles capacités [6].

2.2 Le concept du SDN

Selon le RFC7149 (Boucadair et Jaquenet, 2014), Le SDN (Software-Defined Networking) est un ensemble de techniques visant à faciliter l’architecture, la livraison et l’opération de services réseaux de manière déterministe, dynamique et pouvant être déployé à grande échelle.

L’ONF (Open Networking Foundation, 2013) définit quant à elle le SDN comme étant une architecture qui sépare le plan de contrôle du plan de données, et unifie les plans de contrôle de plusieurs périphériques dans un seul software de contrôle externe appelé « Contrôleur », qui voit le réseau dans sa totalité pour gérer l’infrastructure via des interfaces

de communications appelées APIs. Le contrôleur en question fait abstraction de la couche physique pour les applications qui communiquent en langage développeur, permettant la programmation du réseau [42].

Globalement un réseau est dit SDN en considération des 5 caractéristiques suivantes :

- Séparation du plan de données et du plan de contrôle.
- Périphériques simplifiés.
- Contrôle centralisé.
- Automatisation du réseau et virtualisation.
- Open source.

2.3 Différence entre un réseau SDN et un réseau traditionnel

Le principe de fonctionnement des réseaux actuels a été défini dans les années 50-60, par l'armée des Etats-Unis dans le cadre d'un projet de recherche. L'objectif était de mettre en place un réseau de communication résilient. C'est ainsi que les chercheurs ont inventé le système d'architecture des réseaux distribués. Avec une architecture distribuée, par exemple, les équipements réseaux tels que commutateurs ou routeurs sont actifs et capables de prendre une décision pour l'acheminement des paquets en cas d'indisponibilité d'un de leurs voisins. Cette manière de gérer les réseaux est certes résiliente mais avec le besoin d'intégrer des nouveaux services (sécurité, la qualité de service...), la nécessité d'ajouter des équipements tels que firewalls, IDS/IPS... pour assurer ces services s'est fait sentir. Ainsi, à chaque ajout d'un nouvel équipement, l'architecture se complexifie et devient plus difficile à maintenir. De plus, avec cette approche traditionnelle de gestion des réseaux, chaque équipement doit être configuré indépendamment. Ce qui est non seulement lourd et fastidieux, mais induit aussi le risque de ne pas avoir l'uniformité dans les configurations. D'où le SDN pour permettre la centralisation des configurations. L'idée du SDN a été de simplifier la gestion des réseaux comme nous l'avions annoncé précédemment et de quitter l'architecture distribuée, pour revenir à un système de gestion centralisé du réseau afin de gérer plus facilement les réseaux. La figure 2.1 donne un aperçu sur la différence entre l'architecture traditionnelle et le SDN.

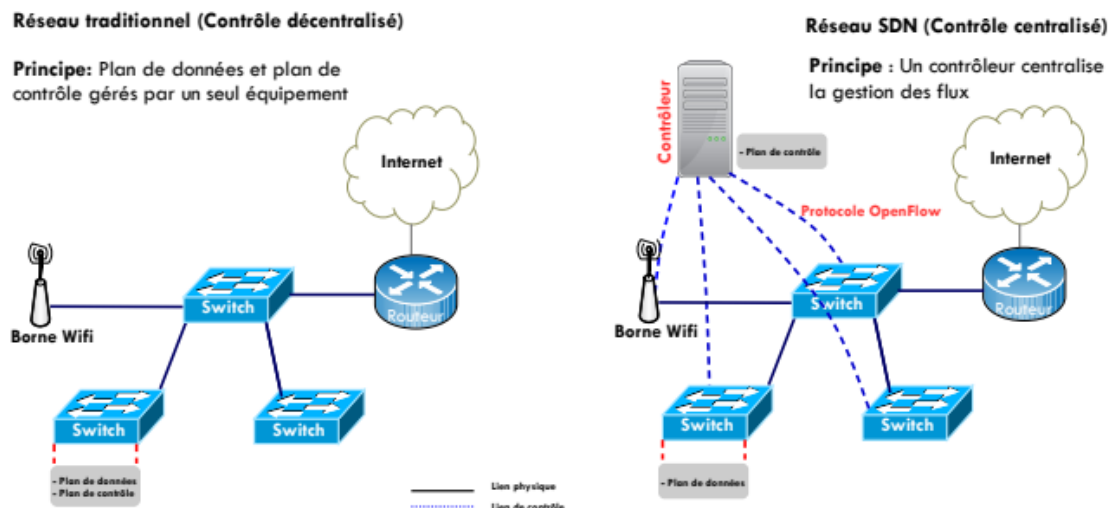


FIGURE 2.1 – Comparaison entre un réseau traditionnel et un réseau SDN.

2.4 Architecture SDN

Un réseau traditionnel est composé généralement des équipements d'interconnexions tels que des switches et des routeurs. Ces équipements incorporent à la fois la partie transmission et la partie de contrôle de réseau. Dans ce modèle d'architecture, il est difficile de développer de nouveaux services, en raison du fort couplage qui existe entre le plan de contrôle et le plan de transmission. Afin d'ouvrir les équipements réseaux aux innovations, l'architecture SDN, a vu le jour. Elle permet de découpler la partie de contrôle de la partie transmission des équipements d'interconnexions. Le SDN est composé principalement de trois couches et d'interfaces de communication Figure 2.2, nous décrivons dans ce qui suit ces couches, ainsi que les interfaces de communications [11] :

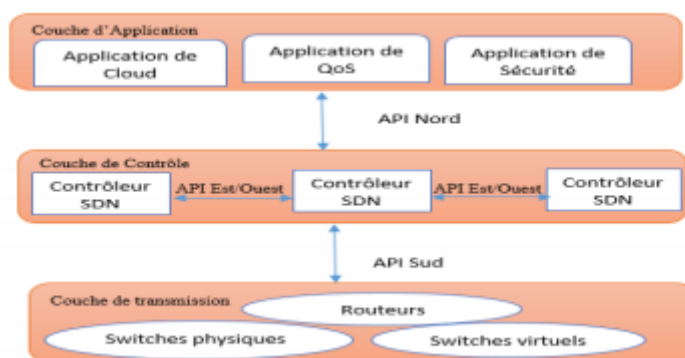


FIGURE 2.2 – Architecture détaillée d'un réseau SDN.

2.4.1 La couche de transmission

Appelée aussi « plan de données », elle est composée des équipements d'acheminement tels que les switches ou les routeurs, son rôle principal est de transmettre les données,

et collecter les statistiques.

2.4.2 La couche de contrôle

Appelée aussi « plan de contrôle », elle est constituée principalement d'un ou plusieurs contrôleurs SDN, son rôle est de contrôler et de gérer les équipements de l'infrastructure à travers une interface appelée " south-bound API ".

2.4.3 La couche application

Représente les applications qui permettent de déployer de nouvelles fonctionnalités réseau, comme l'ingénierie de trafic, QoS, la sécurité, etc. Ces applications sont construites moyennant une interface de programmation appelée " north-bound API ".

2.5 Interfaces de communications

2.5.1 La Southbound API

Les interfaces Sud ou (Southbound APIs) représentent les interfaces de communication, qui permettent au contrôleur SDN d'interagir avec les équipements de la couche d'infrastructure, tel que les switches, et les routeurs. Le protocole le plus utilisé, et le plus déployé comme interface Sud est le protocole OpenFlow, qui a été standardisé par l'ONF, sa dernière version est 1.5, plus de détails sur ce protocole sera donnée dans la prochaine section. Il existe dorénavant d'autres alternatives d'interface Sud, tels que ForCes, ou Open vSwitch Database (OVSDB), mais le protocole openflow est actuellement le standard de facto, qui est largement accepté et répandu dans les réseaux SDN.

2.5.2 Interfaces Nord

Les interfaces Nord servent à programmer les équipements de transmission, en exploitant l'abstraction du réseau fourni par le plan de contrôle. Il est noté que contrairement à la Southbound API qui a été standardisé, l'interface nord reste encore une question ouverte. Bien que la nécessité d'une telle interface standardisée constitue un débat considérable au sein de l'industrie, l'avantage d'une API nord ouverte est aussi important, une API nord ouverte permette plus d'innovation et d'expérimentation. Plusieurs implémentations de cette interface existent, chaque une de ces implémentations offre des fonctionnalités bien différents. Le RESTful considéré comme l'API nord le plus répandue dans les réseaux SDN [11].

2.5.3 Interfaces Est/Ouest

Les interfaces Est/Ouest sont des interfaces de communication qui permettent la communication entre les contrôleurs dans une architecture multi-contrôleurs pour synchroniser l'état du réseau. Ces architectures sont très récentes et aucun standard de communication inter-contrôleur n'est actuellement disponible [11].

2.5.4 Protocole OPENFLOW

Openflow est le protocole utilisé pour la communication entre la couche transmission et la couche de contrôle, il a été initialement proposé et implémenté par l'université de Stanford, et standardisé par la suite par l'ONF, sa dernière version est 1.5. Nous détaillons par la suite la structure d'openflow, son fonctionnement, ses différentes spécifications, ainsi que quelques contrôleurs openflow [11].

Fonctionnement Openflow

Lorsqu'un paquet arrive à un commutateur, le commutateur vérifie s'il y a une entrée dans la table de flux qui correspond à l'en-tête de paquet. Si c'est le cas, le commutateur exécute l'action correspondante dans la table de flux. Dans le cas contraire, c'est-à-dire il n'y a pas une entrée correspondante, le commutateur génère un message asynchrone vers le contrôleur sous la forme d'un (Packet-in), puis le contrôleur décide selon sa configuration une action pour ce paquet, et envoie une nouvelle règle de transmission sous la forme d'un (Packet-out) et (Flow-mod) au commutateur, et enfin, la table de flux du commutateur est actualisée, pour prendre en compte la nouvelle règle installée par le contrôleur. La Figure 2.3 décrit le processus de transmission d'un paquet avec openflow [11].

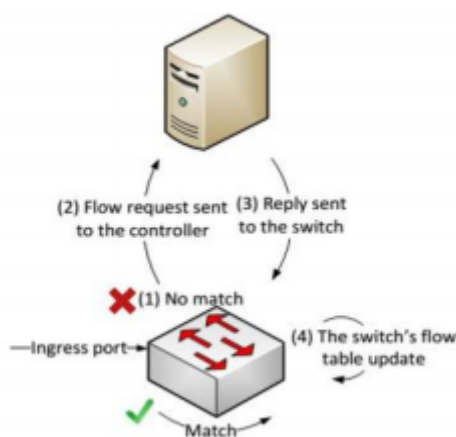


FIGURE 2.3 – Processus de transmission d'un paquet avec openflow.

L'échange d'informations entre le commutateur et le contrôleur s'effectue par l'envoi de messages via un canal de contrôle sécurisé en utilisant TLS (Transport Layer Security).

2.6 Mécanisme d'échanges dans un réseau SDN

Afin d'expliquer plus précisément le fonctionnement d'un réseau SDN, nous présentons ci-dessous un exemple d'échanges de paquets entre deux machines. Nous avons utilisé seulement deux machines et un commutateur pour cette explication mais notre plateforme offre la possibilité de lancer plusieurs machines avec un ou plusieurs commutateurs. Lors d'un échange ICMP dans un réseau SDN avec un commutateur OpenFlow appelé Open

vSwitch (OVS), deux machines h1 et h2 et un contrôleur OpenFlow de type OpenDaylight, comme l'indique la figure 2.4, les différentes phases d'échanges entre h1 et h2 sont :

1. Envoi de paquet de h1 vers l'OVS ;
2. L'OVS va recevoir le paquet ICMP de h1 et le transmet au contrôleur ;
3. Le contrôleur analyse le paquet reçu et installe le flux sur l'OVS ;
4. h2 reçoit le paquet ICMP de h1 ;
5. Echange de flux entre h1 et h2.

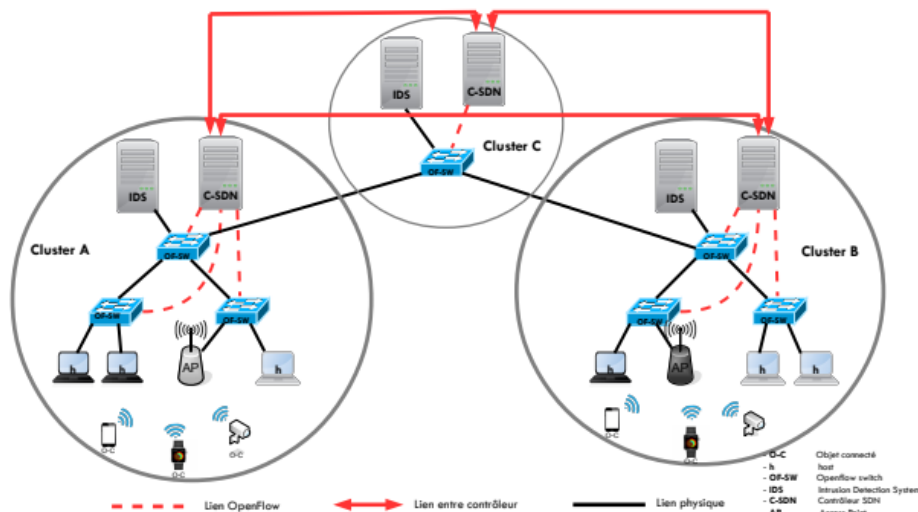


FIGURE 2.4 – Mécanisme d'échanges de flux dans un réseau OpenFlow.

La figure 2.5 ci-dessous montre les types de messages OpenFlow échangés entre le contrôleur SDN et l'OVS. Ces messages sont capturés avec Wireshark, un logiciel d'analyse des paquets réseaux.

Si h1 ou h2 se retrouve compromise après l'installation du flux, ce défaut de sécurité risque de se propager sur la totalité du réseau. C'est pour cette raison que nous avons proposé dans le précédent chapitre une combinaison du SDN avec un IDS, pour améliorer la sécurité dans un réseau en fonction des événements détectés de manière automatisée.

No.	Time	Source	Destination	Protocol	Length	Info
32	64.000751	192.168.1.9	192.168.1.12	OpenFlow	82	Type: OFPT_HELLO
53	112.000803	192.168.1.9	192.168.1.12	OpenFlow	82	Type: OFPT_HELLO
90	162.093014	192.168.1.9	192.168.1.12	OpenFlow	98	Type: OFPT_FEATURES_REPLY
92	162.168260	192.168.1.12	192.168.1.9	OpenFlow	98	Type: OFPT_ROLE_REQUEST
93	162.168514	192.168.1.9	192.168.1.12	OpenFlow	98	Type: OFPT_ROLE_REPLY
95	162.165128	192.168.1.12	192.168.1.9	OpenFlow	74	Type: OFPT_BARRIER_REQUEST
96	162.165305	192.168.1.9	192.168.1.12	OpenFlow	74	Type: OFPT_BARRIER_REPLY
97	162.167858	192.168.1.12	192.168.1.9	OpenFlow	98	Type: OFPT_BARRIER_REQUEST
98	162.168039	192.168.1.9	192.168.1.12	OpenFlow	98	Type: OFPT_ROLE_REPLY
99	162.168116	192.168.1.9	192.168.1.12	OpenFlow	74	Type: OFPT_BARRIER_REPLY
101	162.177513	192.168.1.12	192.168.1.9	OpenFlow	82	Type: OFPT_MULTIPART_REQUEST, OFPMP_TABLE_FEATURES
102	162.177759	192.168.1.9	192.168.1.12	OpenFlow	94	Type: OFPT_ERROR
103	162.182588	192.168.1.12	192.168.1.9	OpenFlow	98	Type: OFPT_MULTIPART_REQUEST, OFPMP_DESC
104	162.182788	192.168.1.9	192.168.1.12	OpenFlow	274	Type: OFPT_MULTIPART_REPLY, OFPMP_PORT_DESC
105	162.182974	192.168.1.9	192.168.1.12	OpenFlow	1138	Type: OFPT_MULTIPART_REPLY, OFPMP_DESC
107	162.261281	192.168.1.12	192.168.1.9	OpenFlow	82	Type: OFPT_MULTIPART_REQUEST, OFPMP_GROUP_FEATURES
108	162.261519	192.168.1.9	192.168.1.12	OpenFlow	94	Type: OFPT_ERROR
109	162.273315	192.168.1.12	192.168.1.9	OpenFlow	82	Type: OFPT_MULTIPART_REQUEST, OFPMP_METER_FEATURES
110	162.273522	192.168.1.9	192.168.1.12	OpenFlow	98	Type: OFPT_MULTIPART_REPLY, OFPMP_METER_FEATURES
112	162.322471	192.168.1.12	192.168.1.9	OpenFlow	194	Type: OFPT_FLOW_MOD
113	162.334773	192.168.1.12	192.168.1.9	OpenFlow	242	Type: OFPT_FLOW_MOD
115	162.700065	192.168.1.12	192.168.1.9	OpenFlow	78	Type: OFPT_SET_CONFIG
117	162.844049	192.168.1.12	192.168.1.9	OpenFlow	219	Type: OFPT_PACKET_OUT

FIGURE 2.5 – Messages OpenFlow échangés entre le contrôleur SDN et l'OVS.

2.7 Conclusion

Dans ce chapitre, nous avons introduit le concept du SDN en expliquant les différentes couches de son architecture ainsi que les API utilisées. Il était question de montrer que cette approche a permis de rendre les réseaux programmables, évolutifs et faciles à innover. Les protocoles standards OpenFlow et OpFlex ont aidé à la concrétisation du SDN. C'est aussi ce qui a nous permis d'expérimenter notre approche de sécurisation des échanges dans un réseau, que nous allons expliquer dans la partie suivante de ce document.

Chapitre 3

LA SÉCURITÉ IOT

3.1 Introduction

Objectifs de sécurité typiques de confidentialité, d'intégrité et La disponibilité (CIA) s'applique également à l'IoT. Cependant, l'IoT a de nombreuses restrictions et limitations en termes de composants et les périphériques, les ressources de calcul et de puissance, et même le nature hétérogène et omniprésente de l'IoT qui introduit préoccupations supplémentaires. Cette section se compose de trois parties : Défis IoT, les problèmes de sécurité spécifiques à chaque couche de l'IoT, et les solutions existences pour sécuriser IoT.

3.2 Défis IoT

L'IoT est une grande industrie qui devrait apporter de nombreuses opportunités commerciales et avantages. L'IoT accentue l'impact positif considérable déjà produit dans notre société, et donc un réel changement de modèles socio-économiques et culturels. Cependant, ces avantages ne peuvent être obtenus en fin de compte sans aborder de nombreux défis et problèmes délicats. Nous énumérons dans ce qui suit les principaux défis auxquels l'IoT est confronté :

- **Évolutivité** : le très grand nombre d'objets connectés à Internet doit être pris en compte dans de nombreux protocoles IoT. Le nombre d'appareils connectés dépassé le nombre de la population humaine en 2010. Par conséquent, il est important pour concevoir une architecture évolutive et construire des protocoles efficaces qui peuvent gérer Problèmes d'évolutivité de l'IoT.

- **Limitation des ressources** : la plupart des appareils IoT sont limités en termes de capacités de stockage et de calcul. Ainsi, il est important de concevoir des poids légers protocoles prenant en charge cette limitation des ressources et répondant aux exigences des clients.

- **Fiabilité** : les systèmes IoT sont vulnérables à de nombreux problèmes de sécurité et de fiabilité cela peut causer d'énormes dégâts. Il est obligatoire de concevoir des systèmes fiables qui fonctionne correctement en toutes circonstances, surtout en cas d'urgence et

des applications critiques telles que la fabrication, le transport et la santé demandes [13].

- **Gestion** : l'une des tâches les plus difficiles de l'IoT est de savoir comment fournir protocoles de gestion en temps réel, légers et sécurisés pour gérer les pannes, la configuration, la comptabilité, les performances et la sécurité (FCAPS) des personnes connectées dispositifs.

- **Mobilité** : est un autre défi important car la plupart des appareils IoT sont mobiles. Il est très important que les services IoT soient fournis à l'utilisateur mobile final avec respectant leurs exigences.

- **Interopérabilité** : les applications, plates-formes, appareils et protocoles IoT sont peut prendre en charge l'interconnexion entre tous les systèmes IoT hétérogènes [13]. Par conséquent, il y a un réel besoin de prendre en compte les aspects d'hétérogénéité de l'IoT pour créer des applications qui peuvent être facilement maintenues, étendues et intégré à d'autres systèmes et applications.

- **Sécurité et confidentialité** : ce défi est probablement le plus difficile qui gêne l'évolution de l'IoT. Ces dernières années, le défi de la sécurité et de la confidentialité est considéré comme l'un des domaines de recherche les plus importants en IoT.

3.3 Les objectifs de sécurité

La sécurité comprend toutes les techniques qui visent à préserver, restaurer et garantir la protection des informations des systèmes informatiques contre les attaques malveillantes. Du quotidien l'actualité met la sécurité au premier plan des préoccupations : fuite de données personnelles et économiques espionnage, infection de systèmes informatiques sensibles, vol d'identité et craintes les paiements par carte ne sont que quelques exemples de menaces. La sécurité des réseaux informatiques et les systèmes d'information en général, consiste à fournir les services suivants [14] :

3.3.1 Confidentialité

Il est très important de s'assurer que les données sont sécurisées et uniquement disponible pour les utilisateurs autorisés. Dans l'IoT, un utilisateur peut être humain, machines et services, et objets internes (périphériques qui sont partie du réseau) et des objets externes (appareils qui ne sont pas partie du réseau). Par exemple, il est crucial de s'assurer que les capteurs ne révèlent pas les données collectées aux voisins nœuds [15]. Un autre problème de confidentialité qui doit être est la manière dont les données seront gérées. C'est important pour les utilisateurs de l'IoT pour être au courant de la gestion des données mécanismes qui seront appliqués, le processus ou la personne responsable de la gestion et de veiller à ce que les données soient protégé tout au long du processus [16].

3.3.2 Intégrité

L'IoT est basé sur l'échange de données entre de nombreux appareils, c'est pourquoi il est très important de garantir l'exactitude des données ; qu'il vient du bon expéditeur

comme ainsi que pour s'assurer que les données ne sont pas altérées pendant le processus de transmission en raison d'ingérence. La fonction d'intégrité peut être imposée par maintenir la sécurité de bout en bout dans la communication IoT. Le trafic de données est géré par l'utilisation de pare-feu et de protocoles, mais cela ne garantit pas la sécurité aux points de terminaison en raison de la nature caractéristique de la faible puissance de calcul à l'IoT nœuds.

3.3.3 Disponibilité

La vision de l'IoT est de connecter autant d'appareils intelligents possible. Les utilisateurs de l'IoT doivent disposer de toutes les données disponibles quand ils en ont besoin. Cependant, les données ne sont pas les seules composant utilisé dans l'IoT ; les appareils et les services doivent être également joignable et disponible en cas de besoin en temps opportun afin de répondre aux attentes de l'IoT.

3.3.4 Authentification

Chaque objet de l'IoT doit être capable d'identifier et d'authentifier d'autres objets. Cependant, ce processus peut être très difficile en raison de la nature de l'IoT ; de nombreuses entités sont impliqués (appareils, personnes, services, prestataires de services et unités de traitement) et une autre chose est que parfois les objets peuvent avoir besoin d'interagir avec les autres pour la première fois (objets qu'ils ne sais pas) [17]. À cause de tout cela, un mécanisme pour l'authentification des entités dans chaque interaction dans l'IoT est nécessaire.

3.3.5 Solutions légères

Les solutions légères sont une fonction de sécurité unique introduit en raison des limites du calcul et capacités d'alimentation des appareils impliqués dans l'IoT. Ce n'est pas un objectif en lui-même plutôt une restriction qui doit être considérée tout en concevoir et mettre en œuvre des protocoles soit en cryptage ou authentification des données et des appareils dans l'IoT. Depuis ces les algorithmes sont destinés à être exécutés sur des appareils IoT avec une capacités, ils doivent donc être compatibles avec l'appareil capacités [18].

3.3.6 Hétérogénéité

L'IoT connecte différentes entités avec différents capacités, complexité et différents fournisseurs. Les appareils ont même des dates et des versions de sortie différentes, utilisez des interfaces techniques et débits binaires, et sont conçus pour une fonction totalement différente, donc les protocoles doivent être conçu pour fonctionner dans tous les appareils différents ainsi que dans différentes situations [17]. L'IoT vise à connecter un appareil à appareil, humain à appareil et humain à humain, il fournit ainsi connexion entre choses hétérogènes et réseaux. Un autre défi qui doit être pris en compte dans l'IoT est que le l'environnement est en constante évolution (dynamique), à un moment l'appareil peut être

connecté à un ensemble complètement différent de appareils que dans un autre temps. Et pour assurer une sécurité optimale un système de cryptographie est nécessaire avec une clé adéquate protocoles de gestion et de sécurité.

3.3.7 Politiques

Il doit y avoir des politiques et des normes pour garantir que les données être géré, protégé et transmis de manière efficace, mais plus important encore, un mécanisme pour appliquer de telles politiques est nécessaire pour garantir que chaque entité applique les normes. Les accords de niveau de service (SLA) doivent être clairement identifiés chaque service impliqué. Politiques actuelles utilisées dans la sécurité des ordinateurs et des réseaux peut ne pas s'appliquer à l'IoT, en raison de sa nature hétérogène et dynamique. La mise en application de telles politiques introduira la confiance des utilisateurs humains dans l'IoT paradigme qui aboutira finalement à sa croissance et évolutivité [18].

3.3.8 Systèmes de gestion de clés

Dans l'IoT, les appareils et les capteurs IoT doivent échanger matériel de cryptage pour assurer la confidentialité des données. Pour à cet effet, il doit y avoir une gestion des clés légère système pour tous les frameworks qui peuvent permettre la confiance entre différentes choses et peut distribuer des clés en consommant des appareils » capacités minimales [18].

Services de sécurité	Mécanismes de sécurité	Quelques exemples
Confidentialité	Cryptage des messages / Cryptage des signes	Cryptographique symétrique (AES, CBC, etc.) ; asymétriques (RSA, DSA, IBE, ABE, etc.).
Intégrité	Fonctions de hachage, message Signature	Fonctions de hachage (SHA-256, MD5, etc.) ; Authentification des messages Codes (HMAC)
Authentification	Chaîne de hachage, Message Code d'identification	HMAC, CBC-MAC, ECDSA
Non-répudiation	Signature de message	ECDSA, HMAC
Disponibilité	Sauts de fréquence pseudo- aléatoires, contrôle d'accès, Systèmes de prévention des intrusions, Pare-feu	Détection d'intrusion basée sur la signature, détection d'intrusion basée sur des anomalies statistiques
Privacy	Pseudonymat, dissociabilité, K-anonymat, preuve de connaissance zéro (ZKP)	EPID, DAA

TABLE 3.1 – Services et mécanismes de sécurité.

3.4 Vecteur de menace

Un vecteur de menace est un moyen pour un attaquant de pénétrer dans les appareils IoT et exécuter des fonctions malveillantes nuire à un appareil ou à un système. Dans l'IoT, la surface d'attaque présente plusieurs vulnérabilités allant des attaques de communication à attaques dynamiques.

Syed Rizvi et al [19] ont divisée le vecteur de menace en trois principales voies d'attaques : attaques de communication, attaques physiques, et les attaques applicatives ou logicielles. Le tableau 3.2 présente un résumé des domaines de sécurité de premier niveau et de leurs contrôles de sécurité.

<i>Domaine de sécurité de premier niveau</i>	<i>Sous-domaine</i>
Domaines	Commercial
	Un service
	Consommateur
	Infrastructure
Contrôles de sécurité	Confidentialité
	Intégrité
	Disponibilité

TABLE 3.2 – Domaines de sécurité et contrôles de sécurité pour l'IOT.

3.4.1 Attaques de communication

Les attaques de communication peuvent être effectuée sur le réseau ou dans l'environnement de l'IoT. Celles-ci les attaques incluent, mais sans s'y limiter, Dos et Distributed Dos (DDoS), spoofing, man-in-the-middle, réseau injection (SQL Injects) et attaques par inondation. La quantité des dommages que ces attaques peuvent causer avec le grand nombre d'IoT appareils existants est stupéfiant. Si un attaquant malveillant pouvait exploiter la puissance de ces appareils, ils pourraient faire des dommages extraordinaires. Nous discuterons de chacun de ces attaques de communication brièvement [19].

- Dos et DDoS : une attaque DoS se produit lorsqu'un attaquant utilise son ou ses propres ressources pour demander des connexions à un serveur qui permet des services à différents utilisateurs, inévitablement brouillage et l'arrêt du serveur qui active les services. Une attaque DDoS est l'endroit où un attaquant (botmaster) utilise des bots (ordinateurs contrôlés par l'attaquant qui peuvent être inconnu du propriétaire de l'ordinateur) pour demander un serveur qui active les serveurs pour différents utilisateurs. Cela crée une surcharge de demandes qui fait inévitablement baisser le serveur. Dans l'IoT, la menace d'une attaque DDoS persiste.

- Injection de réseau : cette attaque peut se produire lorsque n'importe quel appareil ou domaine utilise une base de données SQL. Les attaquants peuvent injecter code malveillant pour faire tomber la base de données, changer informations, ou enregistrer des informations. Dans l'IoT, de nombreux appareils enregistrer et stocker des informations dans diverses bases de données. Les attaques par injection de réseau peuvent être préjudiciables si un l'attaquant accède aux bases de données via des appareils IoT.

3.4.2 Attaques physiques

Les attaques physiques sont limitées à attaquer le réseau via une connexion filaire ou sans fil, ou l'appareil individuellement. Ces attaques physiques sont répertoriées mais ne sont pas limités à l'ingénierie inverse, Brouillage, interférence radio et falsification. Ces attaques se font en utilisant physiquement les appareils à des fins malveillantes l'intention d'endommager l'appareil pour enregistrer, bloquer ou transmettre des messages à un ou

plusieurs autres appareils IoT. Celles-ci les attaques physiques se situent au niveau des couches physiques et MAC du TCP / IP alors qu'au niveau des couches de perception et d'application de l'architecture du réseau IoT [19].

- Ingénierie inverse : cette attaque est utilisée pour prendre un l'appareil et le décomposer en une série d'étapes et effectuer une évaluation de la vulnérabilité pour trouver ces vulnérabilités du périphérique physique. Cela permet à l'attaquant pour exploiter les vulnérabilités connues et inconnues. Après avoir effectué cette ingénierie inverse, on pourrait répliquer une attaque sur chaque appareil connecté à un réseau.

- Brouillage et interférences radio : le brouillage se produit lorsqu'un l'attaquant utilise un appareil pour arrêter la connectivité d'un IoT appareil. L'attaquant se trouve à proximité des personnes vulnérables appareil. L'interférence radio est le brouillage involontaire d'une la connectivité de l'appareil. Cela peut être dû à l'environnement ou la fiabilité de l'appareil lui-même également.

- Falsification : la falsification de l'appareil IoT est possible lorsque l'appareil est en phase de pré-déploiement ou en les phases de développement / fabrication / conditionnement. C'est lorsqu'un attaquant altère faussement processus de développement ou de fabrication de l'appareil. Cette peut être atténué en contenant les appareils du fabricant dans un boîtier inviolable lors de l'expédition et vendu.

3.4.3 Attaques applicatives/logicielles Vecteur de menace

Les menaces logicielles sont toujours impliquées dans le traitement de tout appareil exécutant un logiciel. Tout ce qui va d'une mauvaise configuration dans le code à les scripts malveillants peuvent entraîner des failles de sécurité. Lorsque traitant de logiciels IoT, nous avons principalement affaire à des API et les applications Web en particulier. Il est très important de sécuriser ce vecteur logiciel car il y a plus de possibilités et opportunités de menaces. De plus, les données stockées ou transférés dans ce vecteur ne sont pas des données brutes et pourraient fournir des informations plus utiles [19].

- Injection SQL : l'un des types de code les plus courants l'injection est l'injection SQL. Les injections SQL se produisent lorsqu'un un acteur malveillant entre une requête SQL dans un champ non sécurisé qui sera traité par une base de données SQL. Ce type de la menace est répandue dans tous les types de systèmes, y compris l'IoT. Une préoccupation majeure pour une injection SQL est que cela peut provoquer des élévations de privilèges, accordant à l'attaquant plus accès au système. Pour éviter une attaque par injection SQL, l'application doit valider toutes les données fournies par le client avant de l'utiliser avec des API spécifiques.

- Cross Site Scripting : le Cross Site Scripting (XSS) est un autre type d'injection de code qui exécute un script malveillant via le navigateur Web de la victime. Il provient d'un attaquant et puis est renvoyé par l'application. Essentiellement, il redirige la victime sur un autre site Web et peut faire en sorte que la victime participer à DDoS ou même voler la session de l'utilisateur. Semblable aux injections SQL, la validation des données est obligatoire pour empêcher les attaques XSS.

- Exploitation d'une mauvaise configuration : applications dans l'IoT nécessitent la

configuration de nombreux systèmes et composants pour fonctionner correctement. Par conséquent, chacun de ces composants nécessite une configuration de sécurité appropriée. S'ils ne le sont pas correctement configurés, ils peuvent être facilement exploités par un acteur malveillant. Systèmes d'exploitation, serveurs, frameworks, systèmes de gestion de base de données et toute autre application doivent être correctement configurés pour environnement un IoT sécurisé .

3.5 Problèmes de sécurité dans chaque couche de l'IoT

Chaque couche IoT est sensible aux menaces de sécurité et aux attaques. Ceux-ci peuvent être actifs ou passifs et peuvent provenir de sources externes ou réseau interne suite à une attaque Inside. Une attaque active arrête directement le service tandis que le type passif surveille les informations du réseau IoT sans entraver son service. À chaque couche, les appareils et services IoT sont sensibles aux attaques par déni de service (DoS), qui font l'appareil, la ressource ou le réseau indisponible pour les personnes et les utilisateurs autorisés. Les sections suivantes présentent une analyse détaillée des problèmes de sécurité concernant chaque couche [18].

3.5.1 Couche de perception

Il existe trois problèmes de sécurité dans la couche de perception IoT. D'abord est la force des signaux sans fil. La plupart des signaux sont transmis entre les nœuds de capteur de l'IoT à l'aide des technologies sans-fil dont l'efficacité peut être compromise par vagues dérangeantes. Deuxièmement, le nœud de capteur dans les appareils IoT peut être intercepté non seulement par le propriétaire mais aussi par les attaquants car les nœuds IoT fonctionnent généralement en externe et en environnements extérieur, conduisant à des attaques physiques sur les capteurs IoT et appareils dans lesquels un attaquant peut altérer le composant matériel de l'appareil. Troisièmement, la nature inhérente de topologie de réseau qui est dynamique car les nœuds IoT sont souvent déplacé à différents endroits. La couche de perception IoT se compose principalement de capteurs et de RFID, grâce auxquels leur capacité de stockage, consommation d'énergie et calcul les capacités sont très limitées, ce qui les rend sensibles à de nombreux types de menaces et d'attaques [18].

La confidentialité de cette couche peut être facilement exploitée par Replay Attack qui peut être effectué par usurpation, modification ou rejouer les informations d'identité de l'un des appareils dans l'IoT. Ou l'attaquant peut obtenir la clé de chiffrement en analysant le temps nécessaire pour effectuer le chiffrement, ce que l'on appelle Attaque chronométrée. Une autre attaque menaçant la confidentialité est lorsque l'attaquant prend le contrôle du nœud et capture tous des informations et des données qui sont essentiellement une attaque de capture de nœud. L'attaquant peut ajouter un autre nœud au réseau qui menace l'intégrité des données de cette couche en envoyant des données malveillantes. Cela peut également conduire à une attaque DoS, en consommant l'énergie des nœuds du système [18].

Les problèmes de sécurité énumérés ci-dessus au niveau de la couche de perception

peuvent être adressé à l'aide du cryptage (qui peut être point à point ou de bout en bout), authentification (pour vérifier la véritable identité de l'expéditeur) et contrôle d'accès. Autres mesures et protocoles de sécurité sont donnés .

3.5.2 Couche réseau

Comme mentionné précédemment, la couche réseau de l'IoT est également sensible aux attaques DoS. Outre les attaques DoS, l'adversaire peut également attaquer la confidentialité et la vie privée à couche réseau par analyse du trafic, écoute clandestine et passive surveillance. Ces attaques ont une forte probabilité d'occurrence en raison des mécanismes d'accès à distance et des données échange d'appareils. La couche réseau est très sensible aux Attaque Man-in-the-Middle, qui peut être suivie d'écoute clandestine. Si le matériel de clé des appareils est écouté, le canal de communication sécurisé sera complètement compromis. Le mécanisme d'échange de clés L'IoT doit être suffisamment sécurisé pour empêcher tout intrus d'écoute clandestine, puis vol d'identité [18].

La communication dans l'IoT est différente de celle du Internet car il ne se limite pas à la machine à l'homme. Cependant, la fonction de communication de machine à machine que l'IoT introduit à un problème de sécurité de compatibilité. L'hétérogénéité des composants du réseau le rend difficile d'utiliser les protocoles réseau actuels tels quels, et toujours produire des mécanismes de protection efficaces. Les attaquants peuvent également profiter du fait que tout est connecté dans l'ordre pour obtenir plus d'informations sur les utilisateurs et utiliser les informations pour des futures activités criminelles. Protéger le réseau est important dans l'IoT, mais aussi la protection des objets dans le réseau est tout aussi important. Les objets doivent avoir la capacité de connaître l'état du réseau et la capacité de se protéger de toute attaque contre le réseau. Cette peut être réalisé en ayant de bons protocoles ainsi que des logiciels qui permettent aux objets de répondre à toutes les situations et comportements qui peuvent être considérés comme anormaux ou peuvent affecter leur sécurité [18].

3.5.3 Couche d'application

Étant donné que l'IoT n'a toujours pas de politiques mondiales et normes qui régissent l'interaction et le développement de applications, il existe de nombreux problèmes liés à l'application Sécurité. Différentes applications ont une authentification différente mécanismes, ce qui rend l'intégration de tous très difficile d'assurer la confidentialité des données et l'authentification de l'identité. Les grandes quantités d'appareils connectés qui partagent des données entraîneront une surcharge importante sur les applications qui analysent les données, ce qui peut ont un impact important sur la disponibilité des services. Un autre problème qui doit être pris en compte lors de la conception de l'application dans l'IoT est la façon dont différents utilisateurs interagiront avec eux, la quantité de données qui seront révélées, et qui sera responsable de la gestion de ces applications. Les utilisateurs doivent disposer d'outils pour contrôler les données qu'ils souhaitent divulguer et doivent être conscient de la manière dont les données seront utilisées, par qui et quand [18].

3.6 Mécanismes de sécurité IOT

3.6.1 La protection de la vie privée des utilisateurs et La gestion de confiance

Les utilisateurs des capteurs connectés à l'IoT doivent savoir à quel point leur vie privée est protégée et ils doivent avoir le droit d'autoriser ou de proscrire les manipulations des données reportées par les capteurs qui les entourent. Ainsi, une évaluation préalable de la confiance est plus que nécessaire entre les réseaux de capteurs et les fournisseurs de services de stockage de données de captage sur le cloud et mêmes les entités (les clients) qui demandent de les avoir (les données des capteurs). A ce stade, il est nécessaire de noter que dans le contexte de l'IoT, la gestion de confiance ne tourne pas uniquement entre objets intelligents mais elle se déroule également entre ces objets intelligents et leurs utilisateurs [20].

3.6.2 La gestion des clés

La mise en place des mécanismes pour l'échange des clés entre les stations connectées à Internet, y compris les capteurs dans l'IoT, sur une plateforme reconnue par son insécurité (le réseau Internet) en est un défi majeur du futur pour l'Internet. A cet effet, des solutions de sécurité doivent permettre aux capteurs connectés d'échanger, en toute sécurité, des clés cryptographiques avec les autres dispositifs connectés à Internet. Cependant, un échange de clés avec une faible dissipation énergétiques est vivement recommandé dans l'IoT [20].

3.6.3 La détection d'intrusion

Les solutions cryptographiques et les systèmes de gestion de clés sont très efficaces pour faire face aux attaques externes. Néanmoins, en cas d'existence des attaquants internes (nœuds de compromission) ces solutions deviennent insuffisantes. Cela est dû au fait que les adversaires internes disposent tout comme les nœuds légitimes, des clés cryptographiques ainsi que de tout matériel de sécurité possible. Pour cette raison, il est nécessaire de renforcer la sécurité IoT par des systèmes de détection d'intrusion (IDS) qui offrent une deuxième ligne de défense, en plus des solutions cryptographiques. Un IDS est un système chargé de la détection de l'existence des intrus dans le réseau, à travers un certain nombre d'agents (des nœuds capteurs ordinaires) de surveillance et d'évaluation des comportements et des réputations des nœuds capteurs. Ces agents collectent et diffusent des informations de contrôle visant la détection et l'isolation des nœuds malicieux.

3.6.4 Le contrôle d'accès aux services du IoT

Comme les services des réseaux de capteurs sont souvent très critiques, il est vivement recommandé de définir des règles pour le contrôle d'accès à ces services depuis l'extérieur. Cela signifie que seules les entités autorisées ont le droit d'accéder les ressources des réseaux de capteurs (un capteur, une donnée de captage depuis les capteurs

ou le Cloud, etc.) dans l'IoT. Il y a plusieurs techniques qui peuvent être utilisées pour garantir le contrôle d'accès, en tenant en compte certaines considérations liées aux objectifs derrière l'accès. En limitant l'accès aux données et services des capteurs, certaines problématiques de la protection de la vie privée seront résolues.

Des solutions de contrôle d'accès ont été développées bien avant pour les services de l'Internet classique, comme les techniques de contrôle d'accès à base de rôles ou à base de politiques, mais telles solutions sont très exigeantes en ressources de calcul et en espace mémoire nécessaire pour le stockage des règles de contrôle d'accès. Cela remet en question l'applicabilité de ces solutions pour les réseaux de capteurs intégrés à l'IoT.

3.6.5 Solutions basées sur le Fog Computing

Le Fog Computing a été introduit comme un nouveau paradigme à étendre (et non à remplacer) les ressources de calcul du Cloud computing. Il fournit le stockage, le calcul et mise en réseau / communication à la périphérie du réseau. Architecture informatique de brouillard se compose de nœuds de brouillard déployés à proximité d'appareils IoT et connectés au serveur cloud. L'architecture de brouillard permet de réduire la quantité de données échangées entre les Appareils IoT et infrastructure cloud. L'informatique de brouillard prend en charge la mobilité, l'emplacement prise de conscience, faible latence, hétérogénéité, évolutivité et peut donc être parfaitement adopté dans des applications IoT en temps réel ou sensibles à la latence. Étant donné que les appareils IoT ont limité ressources, les nœuds de brouillard peuvent fournir diverses exigences de sécurité telles que l'authentification, la préservation de la confidentialité et le chiffrement pour sécuriser les environnements IoT [21].

3.6.6 Solutions basées sur la Blockchain

La blockchain est une technologie de rupture qui a révolutionné le monde de la crypto-monnaie. Il s'agit d'un grand livre/base de données distribué qui contient les transactions des nœuds dans un réseau peer-to-peer (P2P). Un ensemble de transactions est regroupé en un seul bloc et validé de manière distribuée à l'aide d'un algorithme de consensus. Ces blocs sont liés pour former une chaîne de blocs ou blockchain comme le montre la figure 3.1.

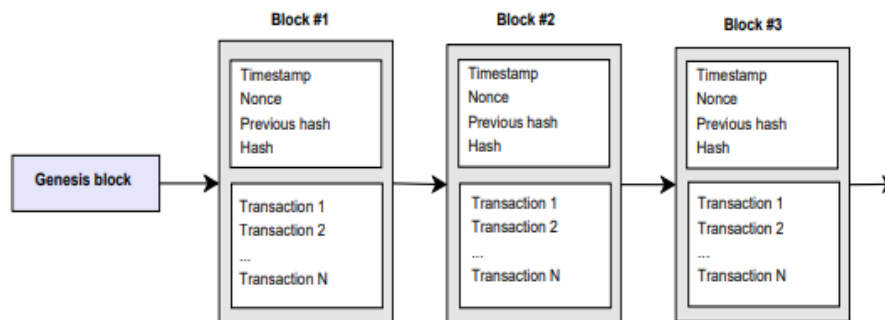


FIGURE 3.1 – Structure de blockchain.

Chaque bloc se compose de deux parties, la première partie représente les transactions validées et la deuxième partie contient l'horodatage du bloc, la valeur nonce, le hachage du bloc et le hachage de bloc précédent. Le processus de consensus est exécuté par certains nœuds du réseau appelés mineurs. Les algorithmes de consensus courants incluent la puissance de travail (PoW), la puissance d'enjeu (PoS), et la tolérance aux pannes byzantine pratique (PBFT). Ces algorithmes peuvent être utilisés pour permettre aux nœuds mineurs de se mettre d'accord sur l'ajout d'un nouveau bloc à la blockchain. Le processus de la validation de la transaction à l'aide de la blockchain est illustré à la figure 3.2 [42].

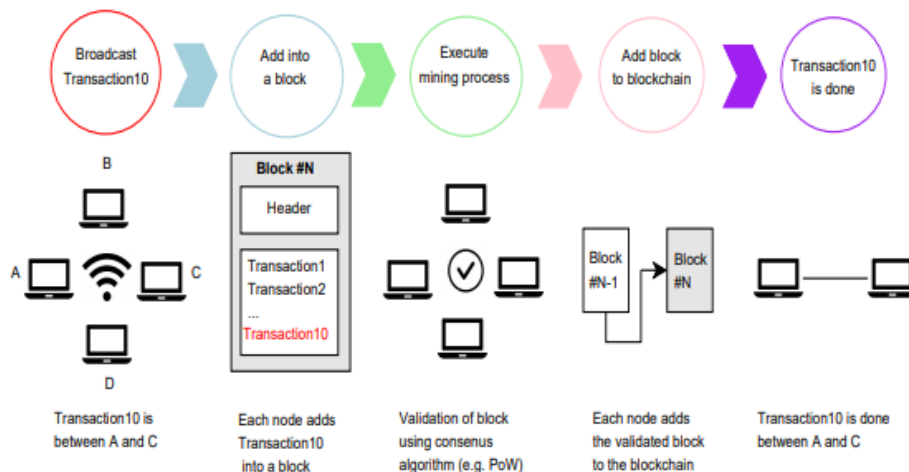


FIGURE 3.2 – Validation de transaction.

Il existe deux principaux types de blockchain, à savoir publique (sans autorisation) et privée (autorisé). Dans la blockchain publique, n'importe quel nœud peut rejoindre le réseau tandis que le privé ne comprend que des nœuds définis. Ethereum et Bitcoin sont les plus populaires blockchains publiques. La sélection du type de blockchain et de l'algorithme de consensus dépend sur la nature et les exigences de l'application IoT. La figure 3.3 montre l'architecture de la blockchain dans l'IoT[42].

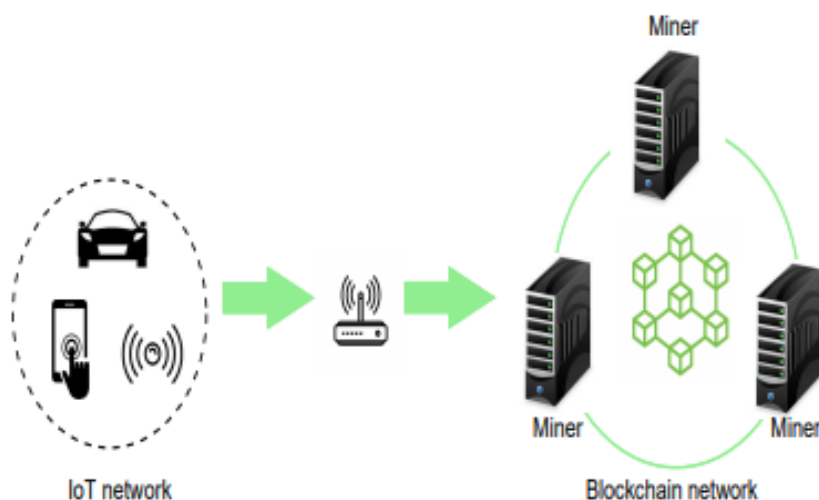


FIGURE 3.3 – Architecture Blockchain pour IoT.

En raison de ses caractéristiques importantes telles que la décentralisation, l'immuabilité, la transparence, la technologie blockchain peut être appliquée dans plusieurs applications IoT pour fournir l'authentification, le contrôle d'accès et la gestion de la confiance[42].

3.6.7 Solutions de chiffrement homomorphiques et interrogeables

Le nombre d'appareils IoT augmente pour permettre la création d'applications. Ces appareils génèrent une quantité massive de données qui doivent être collectées et analysées. Le cloud computing fournit des services de calcul et de stockage pour IoT a collecté des données. Ces données peuvent être très sensibles et doivent donc être protégées contre un accès non autorisé. Pour assurer la préservation de la confidentialité, les données collectées sont cryptées puis stockées dans le cloud public. Le cryptage homomorphique (HE) permet des calculs sur des données cryptées sans révéler les données d'origine. Il existe deux types de chiffement homomorphique de base : partiellement et des méthodes totalement homomorphes [22]. Le cryptage interrogeable (SE) permet une recherche sécurisée sur les données cryptées stockées sur un cloud serveur. Les techniques SE comprennent le SE symétrique, le SE asymétrique et basé sur les attributs SE [23].

3.6.8 Solutions légères basées sur la cryptographie

La cryptographie est un outil efficace pour garantir la confidentialité, l'intégrité et l'authentification. Cependant, la plupart des appareils IoT présentent des caractéristiques difficiles telles que traitement, mémoire et alimentation par batterie. Ainsi, les algorithmes cryptographiques traditionnels ne conviennent pas aux appareils IoT à ressources limitées. Récemment, des primitives cryptographiques légères ont été proposées pour sécuriser les systèmes IoT. Comme le montre la figure 2.7, les algorithmes cryptographiques légers peuvent être classés en quatre classes principales : bloc chiffrements, chiffrements de flux, fonctions de hachage et cryptographie à courbe elliptique (ECC). Dans les chiffrements par blocs, un bloc de texte en clair est chiffré à la fois, tandis que les chiffrements de flux chiffrent / déchiffrent un seul bit ou octet de texte en clair / chiffré. Les fonctions de hachage sont utilisées pour assurer l'intégrité des données en générant un message de longueur fixe à partir d'un message de longueur arbitraire. ECC est une technique cryptographique asymétrique légère qui fournit le même niveau de la sécurité en tant qu'algorithme rivest-shamir-adleman (RSA) avec une taille de clé plus petite [21].

Les techniques cryptographiques légères peuvent être adoptées pour assurer la sécurité des clés exigences, y compris la confidentialité, l'intégrité et l'authentification [21].

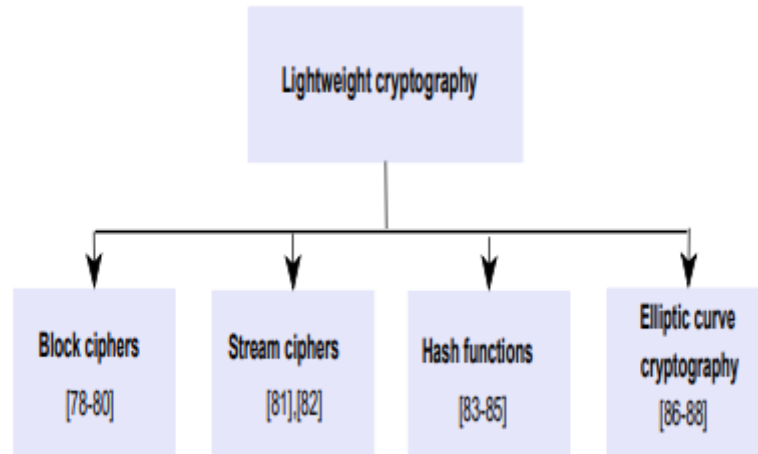


FIGURE 3.4 – Cryptographie légère pour l'IoT.

3.6.9 Software defined networking (SDN)

Avec le développement exponentiel des objets connectés, assurer la gestion de ces dispositifs aux caractéristiques très variées est une tâche difficile pour les administrateurs des réseaux. C'est pourquoi, depuis ces dernières années, les chercheurs dans ce domaine particulier ont concentré leurs travaux sur les nouvelles approches d'architecture réseau, pour mieux organiser et gérer l'IoT. C'est ainsi qu'est née la technologie SDN qui a fait l'objet de notre précédent chapitre. Contrairement aux réseaux classiques, qui ne sont pas conçus pour supporter un passage à l'échelle avec un grand volume de trafic et une forte mobilité, les architectures de type SDN permettent de mieux gérer l'hétérogénéité des réseaux IoT dans le futur. L'émergence des architectures SDN pour IoT est aussi une grande opportunité pour gérer de manière simple la sécurité des objets connectés. Dans cette section, nous présentons un aperçu de certains de ces travaux sur la sécurité des architectures de type SDN pour l'IoT.

Dans [24], les auteurs ont proposé une architecture SDN-IoT avec une implémentation NFV pour faciliter la gestion des objets connectés. Cette architecture se propose, par l'usage du SDN, de faciliter et d'accélérer l'innovation dans la gestion des objets connectés. Pour cela, les auteurs proposent un Framework SDN-IoT.

Bull et al [25] proposent une méthode pour détecter et atténuer le comportement anormal d'un objet connecté à partir d'une passerelle IoT, basée sur le SDN. Les entrées de flux de la passerelle IoT sont préalablement configurées pour permettre d'analyser les flux, afin de détecter tout comportement anormal et associer une action. Trois types d'actions ont été définies à savoir : bloquer, transférer ou appliquer les règles de qualité de service. Le problème de cette solution est la configuration de manière statique des règles sur la passerelle IoT, même si l'usage de plusieurs contrôleurs permet d'assurer la haute disponibilité du réseau.

Dans [26], les auteurs proposent une architecture distribuée de sécurité pour l'IoT avec du SDN. Leur solution permet d'assurer la surveillance et la haute disponibilité du réseau par l'usage de plusieurs contrôleurs SDN synchronisés et repartis par domaine. Ils

proposent également un protocole de routage multi-domaine en environnement SDN pour assurer les échanges entre les contrôleurs des différents domaines. Même si cette solution a l'avantage d'être testée en environnement virtuel, elle ne dispose pas de mécanisme de détection d'intrusion d'où des menaces provenant de l'intérieur du réseau pourraient compromettre la sécurité du réseau.

Dans [27] l'auteur présente une vue d'ensemble de l'état actuel de l'IoT ainsi que les défis de sécurité associés tels que l'identification des objets, la confidentialité et l'intégrité, l'authentification et l'autorisation, et les logiciels malveillants dans IoT. Il propose aussi une architecture de sécurité avec du SDN, basée sur un mécanisme de sécurité où un contrôleur IoT échange avec des agents IoT. Le contrôleur IoT est responsable des décisions à prendre en fonction des informations reçues des agents IoT et puis répercute sur le réseau via le contrôleur SDN. À la réception de la demande de connexion de son agent, le contrôleur IoT établira les règles de transfert en fonction des protocoles de réseau utilisés et communiquera ces règles au contrôleur SDN.

3.7 Conclusion

Nous avons présenté dans ce chapitre un aperçu des travaux de recherche existants sur la sécurité des réseaux IOT. Il est important de noter que les architectures IOT vulnérable à certain type des attaques et des menaces, mais Il est important de noter qu'il existe certains mécanismes efficaces pour sécuriser IOT, parmi ces mécanismes on trouve l'utilisation des réseaux SDN, le concept du SDN a considérablement changé la manière d'administrer et de sécuriser les réseaux.

Chapitre 4

L'architecture proposée

Comme l'indique la figure 4.1, notre solution consiste à un ou plusieurs clusters. Chaque cluster est composé d'un ou de plusieurs équipements réseaux qui sont responsables de l'interconnexion des devises dont des objets connectés. Au sein de chaque cluster, un contrôleur SDN gère le réseau OpenFlow. Chaque contrôleur SDN est couple à un IDS. L'IDS est responsable de la détection des intrusions dans le périmètre réseau de chaque cluster. Autrement dit, un cluster constitue un domaine du SDN dans lequel nous utilisons un réseau OpenFlow avec un contrôleur SDN et un IDS pour gérer le domaine de sécurité, que nous appelons zone de confiance. Pour former une zone de confiance, l'ensemble des équipements et devises de cette zone doit être totalement sécurise. Ce travail de sécurité est assuré par le couple contrôleur et IDS. Le contrôleur SDN sert de firewall intelligent pour la zone de confiance et dispose des règles de sécurité spécifiques aux besoins de sécurité du cluster. Ces règles sont programmées par un administrateur. Elles peuvent être distribuées sur les autres zones de confiance si le besoin de sécurité est le même à travers l'API East-West.

Notre approche permet, non seulement de gérer la sécurité de manière totalement décentralisée à travers une gestion locale de la sécurité par le couple contrôleur SDN/IDS, mais aussi que les contrôleurs échangent les informations sur les menaces détectées dans leurs clusters respectifs. Le contrôleur SDN est l'élément central pour la gestion de la sécurité dans chaque zone de confiance. Il a une vue globale du réseau, gère le trafic et distribue les politiques de sécurité sur les équipements réseaux de son propre cluster.

Avant d'isoler la menace par le contrôleur SDN dans chaque cluster, il faut la détecter. C'est pourquoi nous utilisons un IDS pour résoudre ce problème. Dans la pratique, cela peut être réalise par la mise en place d'un IDS de type snort ou autre.

Nous avons ajouté des mécanismes d'authentification légère entre les différentes devises IoT et la passerelle avec l'algorithme légère Poly 1305. En plus d'un cryptage des données avec l'algorithme AES-128.

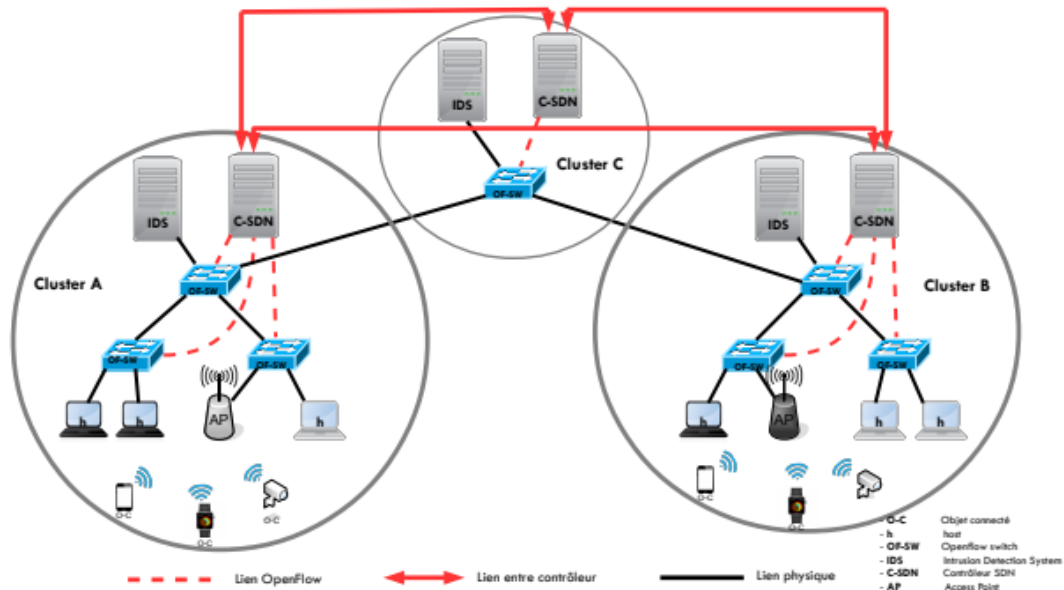


FIGURE 4.1 – Architecture proposée pour sécuriser réseau IoT

4.1 Firewall Intelligent SDN

L'augmentation du nombre d'équipements connectés et le volume de données qu'ils génèrent a considérablement augmenté le besoin de sécurité de ces données. Actuellement, la sécurisation des échanges dans un réseau est en grande partie réalisée par des efforts d'anticipation des menaces et en utilisant les mécanismes traditionnels, tels que firewall, IDS/IPS... Mais ces approches doivent être améliorées pour assurer la sécurité de ces objets connectés car ils sont hétérogènes et mobiles. Autrement dit, les réseaux sont devenus trop complexes et sans frontières.

Plusieurs travaux dans la littérature proposent des solutions de sécurité en utilisant l'architecture SDN. Certains ont développé et implémenté des applications de firewalling dans [33, 34, 35, 36, 37, 38] sur le contrôleur du SDN, d'autres ont installé des règles de sécurité directement sur le commutateur OpenFlow, pour assurer la sécurité dans un réseau.

Les politiques de sécurité sont définies sous forme de règles OpenFlow. Ces règles OpenFlow utilisent les Northbound API pour envoyer les instructions au contrôleur. Ce dernier récupère les informations reçues de l'application firewall, traduit les requêtes abstraites en commandes et les envoie aux équipements réseaux (commutateurs, routeurs...). Les équipements réseaux, qu'ils soient physiques ou virtuels, appliquent la demande des changements reçus du contrôleur pour sécuriser les échanges selon les attentes du firewall. La communication des équipements réseaux avec le contrôleur SDN est faite en utilisant les Southbound API, en l'occurrence OpenFlow.

SDN vise essentiellement à permettre aux administrateurs réseau de gérer et de contrôler l'ensemble du réseau via un contrôleur basé sur un programme logiciel. Cet objectif est atteint grâce à la séparation du plan de données et du plan de contrôle, ce qui simplifie

les services de mise en réseau.

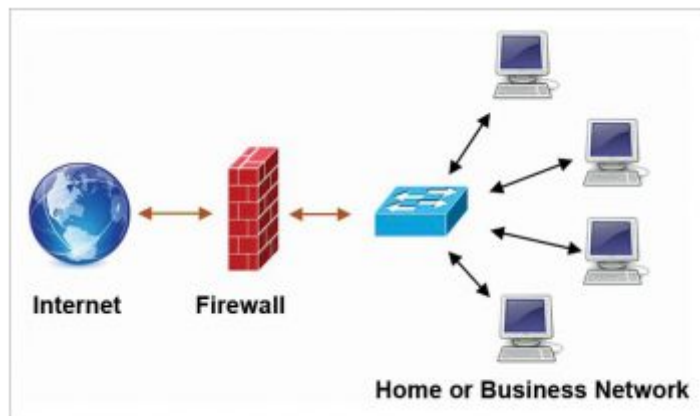


FIGURE 4.2 – Pare-feu traditionnel.

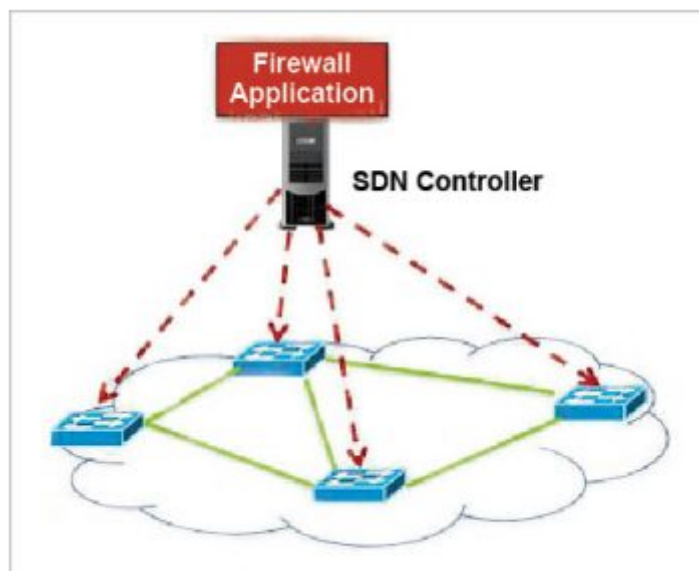


FIGURE 4.3 – Pare-feu basé sur SDN.

Voici quelques différences entre le pare-feu traditionnel et le pare-feu basé sur SDN :

- Le trafic interne n'est pas visible et ne peut pas être filtré par un pare-feu traditionnel.
- Un pare-feu basé sur SDN fonctionne à la fois comme un filtre de paquets et un vérificateur de politique.
- Le premier paquet passe par le contrôleur et est filtré par le pare-feu SDN.
- Les paquets suivants du flux correspondent directement à la politique de flux définie dans le contrôleur.
- La politique de pare-feu est définie de manière centralisée et appliquée au niveau du contrôleur.

4.2 Analyse, détection et génération des alertes (IDS/IPS)

La seconde étape est l'analyse et la détection de menaces puis la génération d'une alerte. Pour réaliser cela, nous avons utilisé un IDS pour écouter l'ensemble du trafic du réseau, analyser et détecter les flux malveillants.

L'IDS analyse les données réseaux et décelé des anomalies ou des patterns d'attaque prédéfinis par l'administrateur réseau. Cette détection porte principalement sur l'analyse des entêtes des paquets des couches réseaux et transports mais également sur le contenu des paquets.

Pour détecter un flux malveillant, l'IDS utilise principalement deux méthodes d'analyse, à savoir la méthode de détection à base de signatures permettant de déceler des patterns connus dans les données analysées, ou la méthode de détection comportementale détectant les déviations d'un comportement vis à vis d'un profil normal. Dans les deux cas, l'IDS compare les données analysées à une référence décrite, soit par une signature, soit par un profil normal.

Une fois les données analysées, l'IDS peut générer une alerte sous forme de fichier log en cas de flux malveillants.

En dernière étape, on va extraire et analyser les fichiers logs puis d'envoyer une instruction au contrôleur SDN pour supprimer dynamiquement les flux malveillants via l'API REST.

En suivant cette procédure, dans chaque zone de confiance, l'IDS analyse le trafic et envoie une alerte en cas de flux malveillants au contrôleur, qui prendra une décision en envoyant une règle de flux au commutateur OpenFlow via le protocole OpenFlow, pour interdire le flux en question. Chaque contrôleur a ses propres politiques de sécurité implémentées en fonction des retours de l'IDS.

La notion de zone confiance nous permet d'avoir des domaines de sécurité réseau et qui ont la possibilité de partager les informations sur les menaces avec d'autres clusters.

4.3 Sécurisation du lien entre le contrôleur OpenDaylight et l'OVS

le canal de communication entre l'OVS et le contrôleur OpenDaylight n'est pas chiffré par défaut, ce qui veut dire que le cryptage des messages d'échanges OpenFlow entre ces deux éléments du réseau SDN ne s'exécute pas automatiquement. De plus, certains contrôleurs ne supportent même pas le support du TLS pour le chiffrement des communications entre le commutateur SDN et le contrôleur. Un hacker peut exploiter ce manque de sécurité sur le canal OpenFlow pour attaquer le réseau et mener des actions malveillantes. Ce qui est extrêmement dangereux si le hacker obtient un accès au contrôleur qui lui permettrait d'avoir le contrôle de l'ensemble du réseau. Avec une prise en main du contrôleur, le hacker peut supprimer les commutateurs OpenFlow, modifier les règles OpenFlow dans le commutateur, capturer le trafic sensible et surveiller la manière dont le contrôleur gère les paquets OpenFlow. C'est pourquoi, il faut chiffrer en SSL/TLS

les échanges des messages OpenFlow sur le canal entre l'OVS et l'OpenDaylight. Le chiffrement des messages OpenFlow entre l'OVS et l'OpenDaylight est crypté à l'aide d'une connexion SSL/TLS, basée sur le modèle d'infrastructure à clé publique appelée Public Key infrastructure (PKI).

4.4 Authentication IoT (Lightweight Cryptographie)

Lightweight Cryptography (La cryptographie légère) : Est une méthode de cryptage qui présente un faible encombrement et une complexité informatique réduite. La cryptographie légère devient pertinente dans le cas d'Internet of Things, un réseau sans fil à configuration automatique entre objets de différentes classes, pouvant inclure des appareils, des véhicules, des capteurs intelligents et des étiquettes RFID. Les facteurs suivants relatifs à la mise en œuvre sont nécessaires pour la cryptographie légère.

- Taille (taille du circuit, tailles ROM / RAM)
- Puissance
- Consommation d'énergie
- Vitesse de traitement (débit, délai)

4.4.1 Poly1305-AES

Poly1305-AES : est un code d'authentification de messages à clé secrète qui convient à une grande variété d'applications. Poly1305-AES calcule un authentifiant de 16 octets d'un message de n'importe quelle longueur, en utilisant un nonce (numéro de message unique) de 16 octets et une clé secrète de 32 octets. Les attaquants ne peuvent pas modifier ou falsifier les messages si l'expéditeur du message transmet un authentificateur avec chaque message et que le destinataire du message vérifie chaque authentificateur [29].

4.4.2 CMAC

Le National Institute of Standards and Technology (NIST) a récemment spécifié le code d'authentification de message basé sur le chiffrement (CMAC). CMAC [NISTCMAC] est une fonction de hachage à clé basée sur un chiffrement par bloc de clé symétrique, tel que Advanced Encryption Standard [NIST-AES]. CMAC est équivalent au One-Key CBC MAC1 (OMAC1) soumis par Iwata et Kurosawa [OMAC1a, OMAC1b]. OMAC1 est une amélioration du mode eXtended Cipher Block Chaining (XCBC) soumis par Black et Rogaway [XCBCa, XCBCb], qui est lui-même une amélioration de l'authentification de base des messages d'enchaînement de blocs de chiffrement Code (CBC-MAC). XCBC résout efficacement les lacunes de sécurité de CBC-MAC, et OMAC1 réduit efficacement la taille de clé de XCBC [30].

4.4.3 HMAC

L'algorithme HMAC signifie code d'authentification de message haché ou basé sur le hachage. C'est le résultat d'un travail de développement d'un MAC dérivé de fonctions de

hachage cryptographique. HMAC est un excellent résistant aux attaques de cryptanalyse car il utilise le concept de hachage deux fois. HMAC comprend deux avantages de hachage et de MAC, et est donc plus sécurisé que tout autre code d'authentification. La RFC 2104 a publié HMAC, et HMAC a été rendu obligatoire à mettre en œuvre dans la sécurité IP. La norme FIPS 198 NIST a également publié HMAC[31].

4.4.4 Comparaisons

Dans cette étude, on va utiliser comme matériel de travail une machine personnelle (PC Portable) avec les caractéristiques suivantes :

- RAM : 1.9Go
- CPU : 2.1 GHz
- Operating System : Ubuntu 20.04 LTS

Generation du MAC

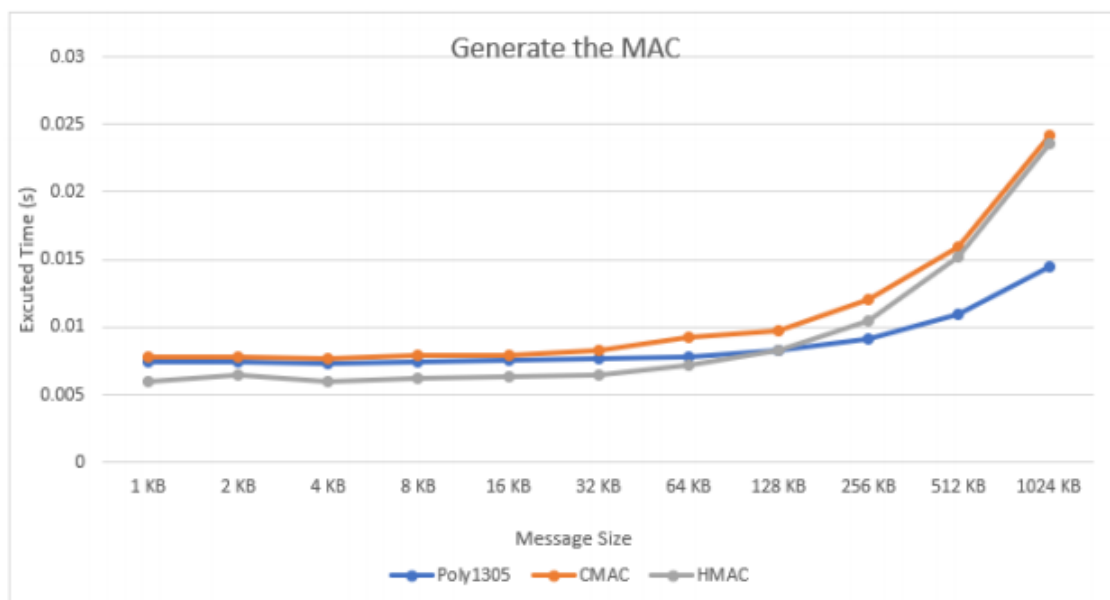


FIGURE 4.4 – Temps d'exécution pendant la génération mac.

Validation du MAC

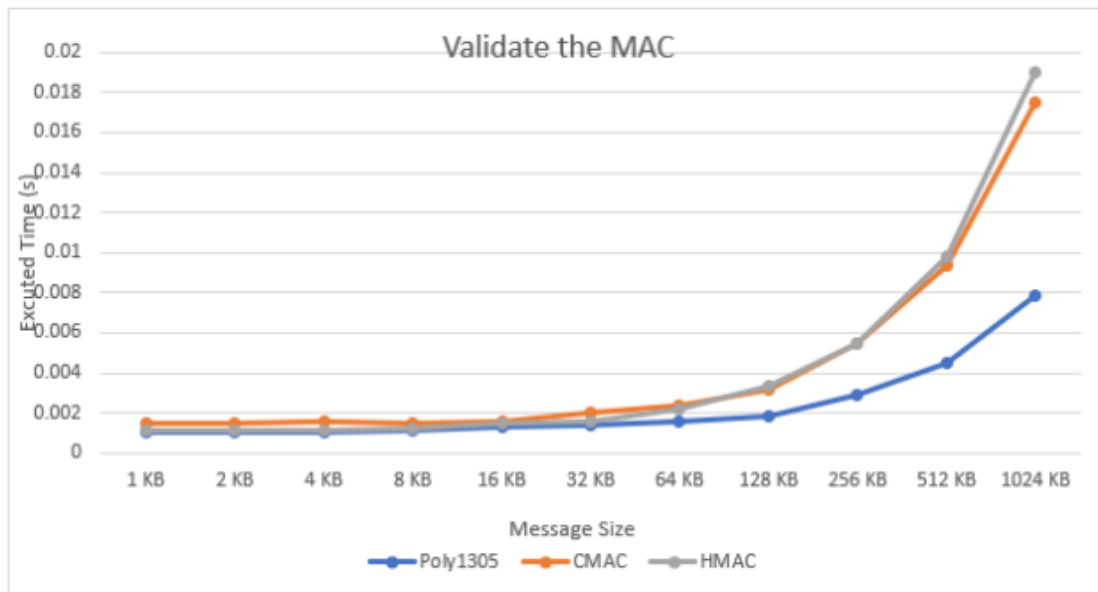


FIGURE 4.5 – Temps d'exécution pendant la validation du mac.

Conclusion

Dans cette partie, on a choisi trois algorithmes de MAC parmi les algorithmes de MAC les plus connues et qui sont Poly1305-AES, CMAC et HMAC, Nous avons étudié ces trois algorithmes théoriquement. Et après, nous avons fait une comparaison entre ces 3 algorithmes en point de vue temps d'exécution en fonction de la taille du message traité. Nous avons constaté finalement que Poly1305-AES est le bon choix pour le reste de notre projet.

4.5 Chiffrement IoT (Lightweight Cryptographie)

4.5.1 Advanced Encryption Standard

Advanced Encryption Standard (En abréviation AES) : Est un algorithme de chiffrement symétrique par blocs, gagné en octobre 2000 le concours AES lancé par le NIST en 1997, cet algorithme est approuvé par NSA, et aussi il est l'algorithme de chiffrement le plus utilisé actuellement. Il chiffre les données avec des blocs de 128bits, avec une clé de 128, 192, ou bien 256bits, et cet algorithme fonctionne sur 16 tours. Algorithme AES est utilisé non seulement pour la sécurité, mais aussi pour sa grande vitesse. La mise en œuvre matérielle et logicielle est encore plus rapide. Il est utilisé dans de nombreux protocoles tels que Secure Sockets Layer (SSL)/Transport Layer Security (TLS) et se trouve dans la plupart des applications modernes et les appareils qui ont besoin d'une fonctionnalité de cryptage. Il est soigneusement testé pour de nombreuses applications de sécurité notamment dans le domaine d'IOT. Ce qui fait qu'il est un candidat idéal pour cette étude [32].

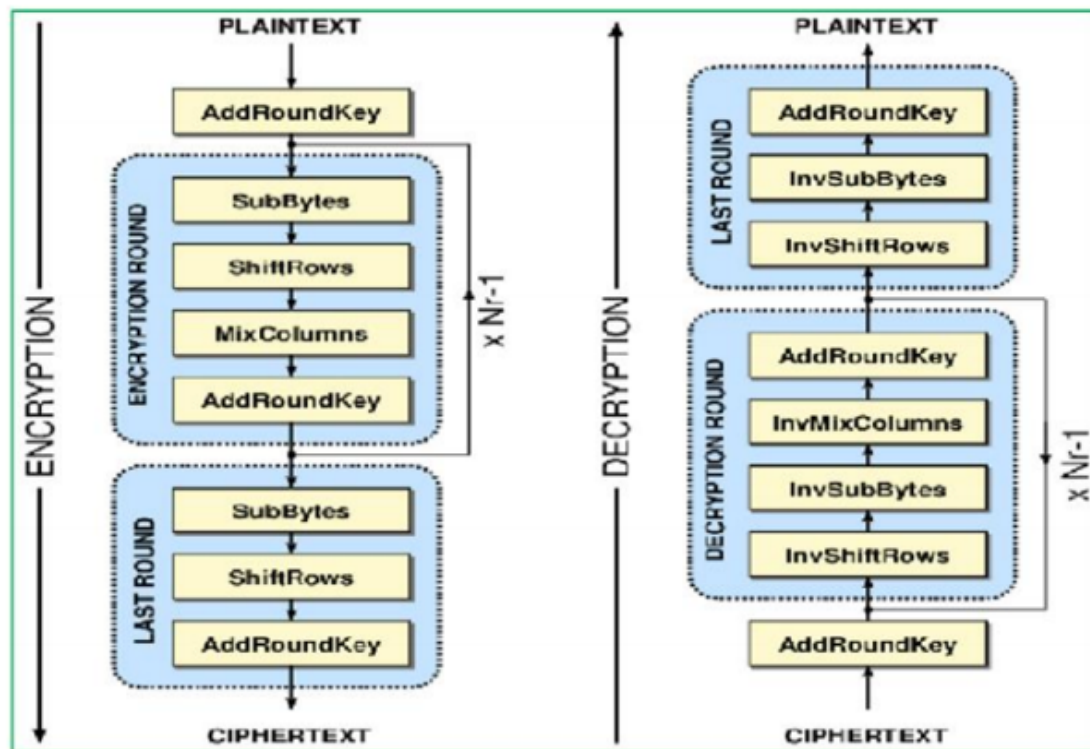


FIGURE 4.6 – Fonctionnement d'AES

4.5.2 Data Encryption Standard

Data Encryption Standard (En abrégiation DES) : Est un algorithme de chiffrement symétrique par blocs, chiffre les données les en divisés en des blocs de 64bits, utilisant des clés de 56bits, avec 16 tours. Le premier standard DES est publié par FIPS le 15 janvier 1977 sous le nom FIPS PUB 46. La dernière version avant l'obsolescence date du 25 octobre 1999.

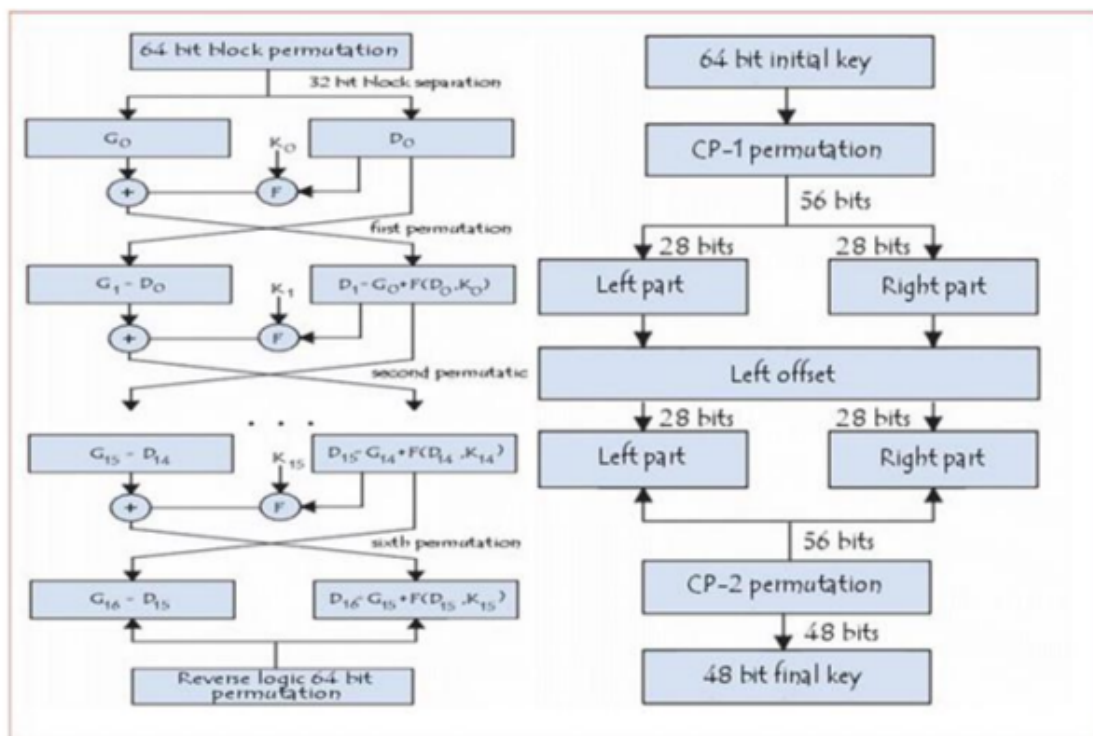


FIGURE 4.7 – Fonctionnement de DES.

4.5.3 Blowfish

Blowfish : Est un algorithme de chiffrement par blocs, crée par Bruce Schneier en 1993. Cet algorithme chiffre et déchiffre les messages sur 64 bits, et utilise une clé de longueur variable entre 32 et 448bits. Blowfish est basé sur une idée principale qui dit « la bonne sécurité contre les attaques de cryptanalyse peut être obtenue en utilisant de très grandes clés pseudoaléatoire ». Blowfish dans sa version complète de 16 tours à ce jour est très fiable, et la recherche exhaustive (Brute Force) reste la seule moyenne pour l'attaquer.

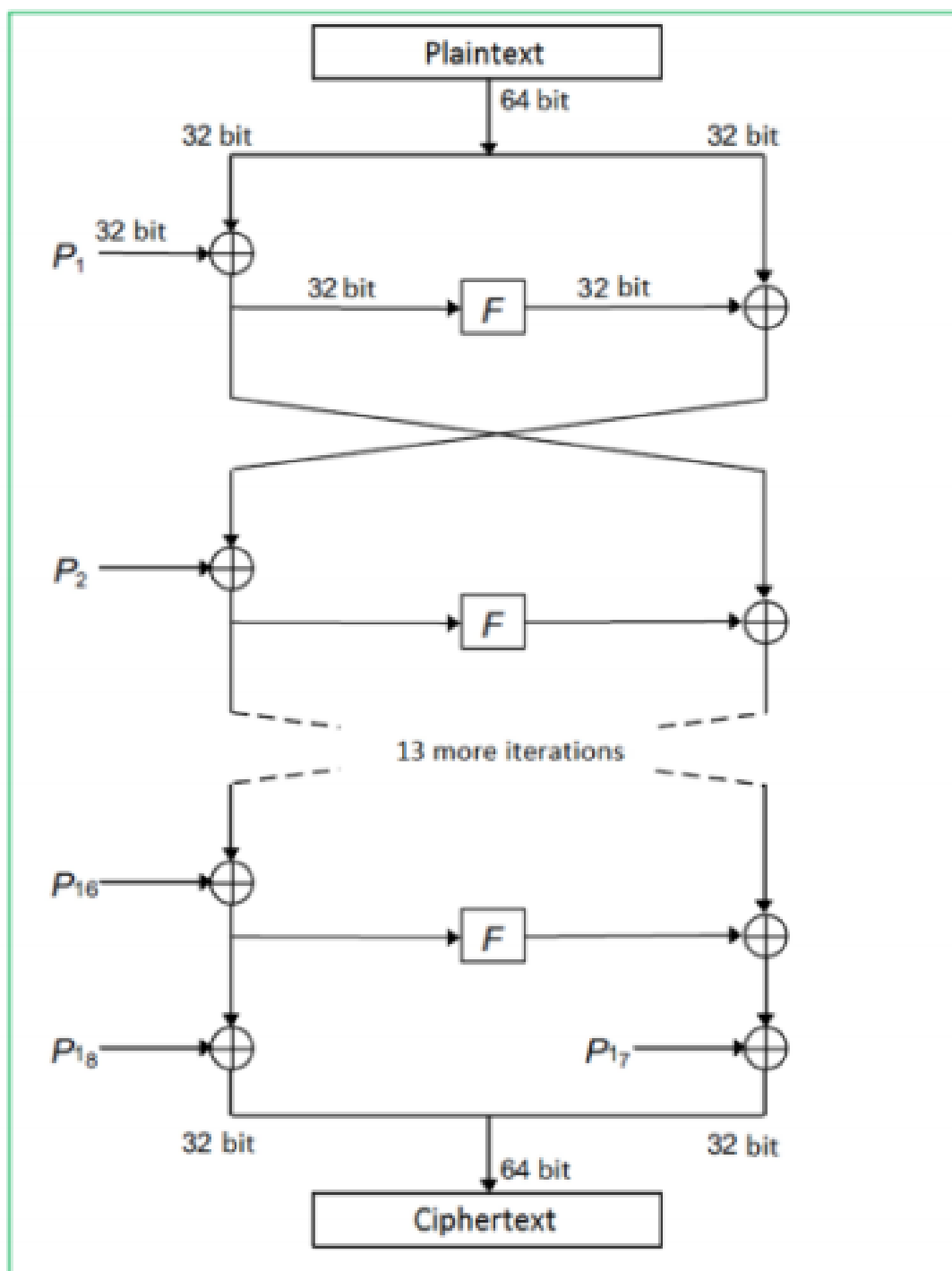


FIGURE 4.8 – Fonctionnement de Blowfish.

4.5.4 Comparaisons

Chiffrement

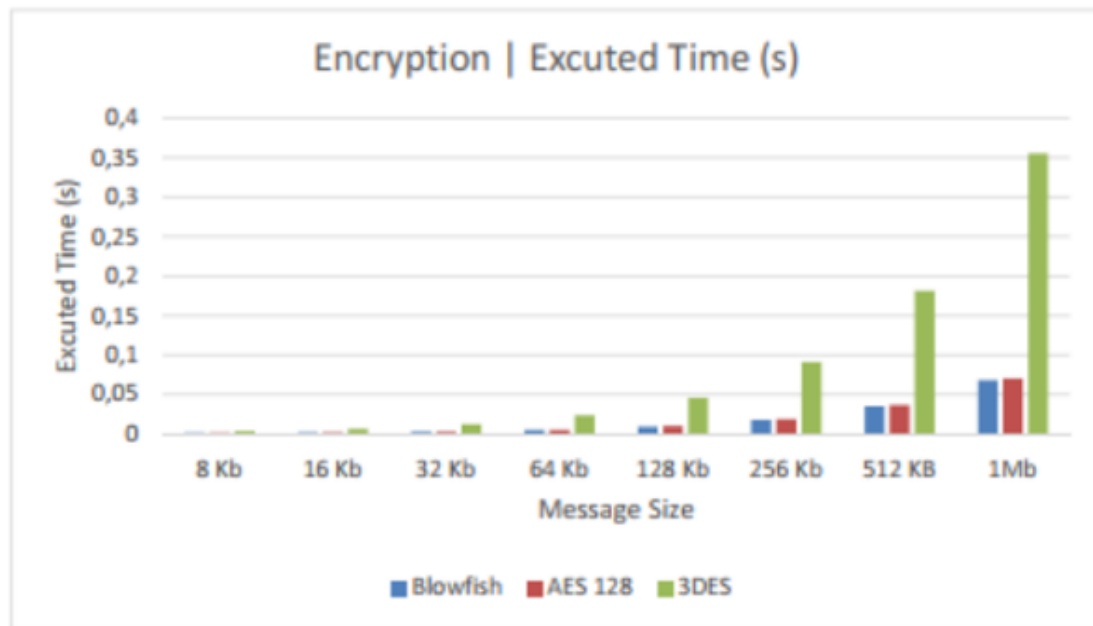


FIGURE 4.9 – Temps d'exécution pendant le chiffrement.

Déchiffrement.

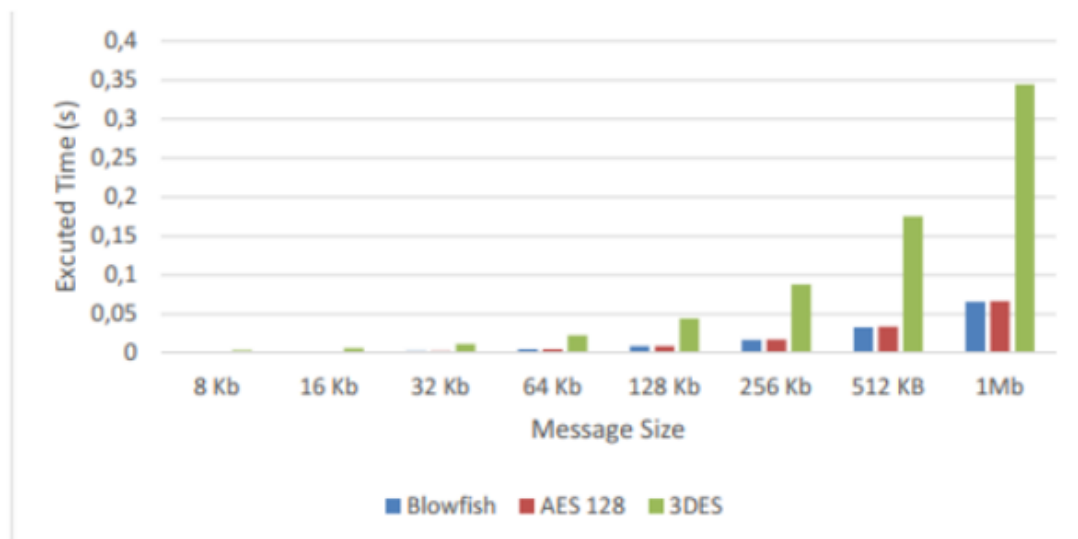


FIGURE 4.10 – Temps d'exécution pendant le déchiffrement.

Conclusion

Dans cette partie, on a choisi trois algorithmes de chiffrement parmi les algorithmes de chiffrement les plus connues et qui sont AES, DES et Blowfish, nous avons étudié ces trois algorithmes théoriquement. Et après, nous avons fait une comparaison entre sur ces

3 algorithmes en point de vue temps d'exécution en fonction de la taille du message traité. Et on a constaté finalement qu'AES est le bon choix pour le reste de notre projet, mais avec des bons résultats pour le Blowfish qui peut être considéré comme un deuxième choix. Par contre nous avons conclu que le DES donne des mauvais résultats par rapport celles du Blowfish et AES.

4.6 Conclusion

Dans ce chapitre nous avons décrit notre approche pour sécuriser les réseaux IoT avec le SDN. La proposition par Cisco du protocole OpFlex, un protocole complémentaire à OpenFlow, a permis d'améliorer de manière significative la distribution des politiques de sécurité sur un commutateur OpenFlow. Cette gestion déclarative des politiques de sécurité dans un réseau permet de rendre le réseau plus résilient et évolutif. Ce qui fait que, même en cas de panne d'un contrôleur SDN, le réseau reste en service en attendant la maintenance car les équipements réseaux conservent leur intelligence, contrairement à l'idée initiale du SDN où les équipements réseaux ne disposent pas d'intelligence. nous avons fait aussi une comparaison entre trois algorithmes de chiffrement et trois algorithmes d'authentification, nous avons basé sur le temps de génération et validation de MAC et le temps de chiffrement et déchiffrement, et nous avons utilisé des messages de petite taille conviennent aux équipements IoT, nous avons par la suite constaté que AES et POLY1305 est le bon choix pour le reste de notre projet.

Chapitre 5

Implémentation

5.1 Introduction

Dans ce chapitre, nous présentons notre maquette d'expérimentation qui nous a permis de tester notre approche de sécurisation des échanges dans un réseau, basée sur le concept du SDN, présentée dans le précédent chapitre, avec une description des outils utilisés dans nos différentes phases de mise en œuvre. Concrètement, nous avons implémenté un réseau géré par un contrôleur de type OpenDaylight, un IDS snort, un Par-feu sur le contrôleur et la mise en place de l'authentification et de chiffrements entre les équipements IOT et la communication entre le OVSwitch et OpenDaylight.

5.2 Les outils utilisés

Pour mettre en œuvre notre expérimentation en environnement virtuel, nous avons fait appel à plusieurs outils à savoir : un contrôleur OpenDaylight, un Switch OVS, un IDS de type Snort, un Nmap, Wireshark, système Ubuntu. Nous avons utilisé des outils de gestion de connexions à distance des machines virtuelles open source multi-onglets et multi-protocoles, et Virtual Network Computing (VNC) pour accéder plus facilement à notre réseau et à nos machines à distance via un OpenVPN permettant d'assurer la sécurité de la connexion.

5.2.1 Le contrôleur OpenDaylight

Le projet OpenDaylight (ODL) est une plate-forme collaborative et open source pour accélérer l'adoption et l'innovation de la mise en réseau logiciel (SDN) et de la visualisation des fonctions réseaux (NFV). ODL est un logiciel basé sur Java et pris en charge par l'industrie, géré par le consortium Linux Foundation avec près de 50 entreprises membres, dont Brocade, Cisco, Citrix, Dell, Ericsson, HP, IBM, Juniper, Microsoft et Red Hat. La mission d'ODL est de créer une communauté collaborative qui partage et contribue au succès et à l'adoption du SDN [1].

- **Architecture OpenDaylight :**

L'architecture de l'OpenDaylight ressemble comme pour la plupart des architectures SDN à une architecture classique sur trois couches qui sont à savoir : La couche application, la couche contrôleur et enfin la couche infrastructure réseau. La figure représente une architecture simplifiée d'OpenDaylight.

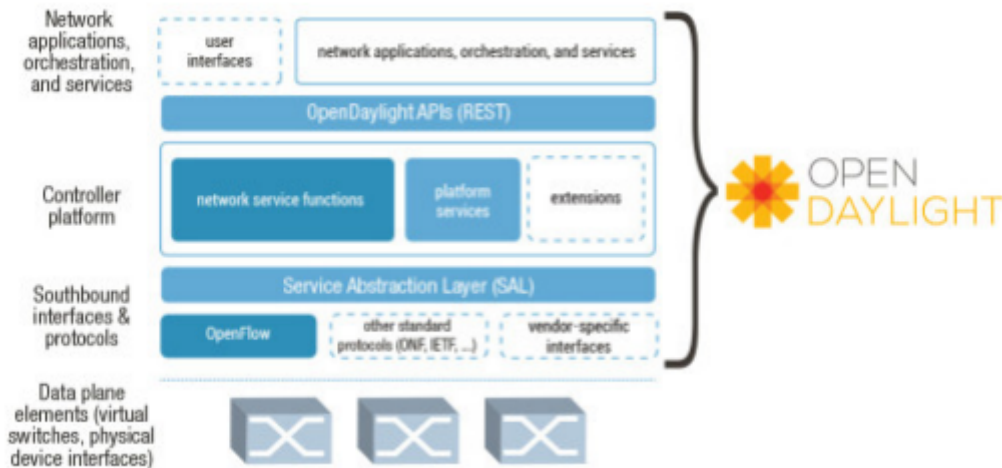


FIGURE 5.1 – Architecture simplifier d'OpenDaylight.

Si l'on devait donner une définition simplifiée pour chacune des trois couches on dirait que :

La couche application :

Possède l'application client qui permet de programmer le réseau physique. L'interface application client possède les fonctionnalités que le client peut opérer sur son réseau. Les fonctionnalités qui sont donc visualisables sur l'interface graphique du client et donc cliquables vont alors déclencher l'exécution d'APIs qui vont interagir avec la couche du contrôleur. On parle alors d'APIs Nord (NB APIs). ODL utilise deux APIs nord qui sont : REST API et OSGi APIs.

La couche contrôleur :

Cette couche est celle du contrôleur. Elle communique avec la couche application via les APIs cité plus haut ainsi que la couche infrastructure réseau (qui contient tout le matériel réseau). Cette couche permet de récupérer les APIs de la couche application et de les exécuter afin de comprendre le comportement qu'elle souhaite exécuter sur le matériel réseau. Une fois le comportement extrait, cette couche se charge de transférer cette information à la couche infrastructure réseau pour que le comportement puisse prendre effet sur le métier ciblé par l'application. La communication entre le contrôleur et la couche infrastructure se fait via des APIs nommés API Sub (SB APIs).

La couche infrastructure réseau :

La couche infrastructure réseau est composée de divers équipements de réseaux qui forment le réseau sous-jacent pour acheminer le trafic du réseau. Il peut s'agir d'un ensemble de commutateurs et de routeurs de réseaux dans le centre de données. Cette couche serait la couche physique sur laquelle la virtualisation du réseau serait établie par l'intermédiaire de la couche de contrôle (où les contrôleurs s'associeraient et gèreraient le réseau physique sous-jacent).

On note donc, que la communication entre la couche infrastructure et la couche contrôle se fait via des API SUDs. Cette communication est établie grâce au SAL de la couche de contrôle qui communique via des APIs SUD avec les équipements réseaux (par exemple via le standard OpenFlow). Le SAL est donc le cœur du contrôleur ODL [1].

5.2.2 Open vSwitch (OVS) :

L'OVS est une implémentation logicielle open source d'un switch Ethernet avec la particularité d'être multicouche et distribuée. Elle est conçue pour fonctionner comme un commutateur dans les environnements de machines virtuelles. Son objectif est de fournir une fonction de routage niveau 2 et 3 du modèle OSI pour les environnements virtualisés supportant différents protocoles et standards. L'OVS supporte le protocole OpenFlow et fait intégralement partie du SDN. Dans nos travaux, elle nous a permis de faire communiquer des machines virtuelles Alpine Linux entre elles [1].

5.2.3 Système d'exploitation :**Ubuntu**

Ubuntu est un système d'exploitation GNU/Linux basé sur Debian. Il est développé, commercialisé et maintenu pour les ordinateurs individuels (desktop), les serveurs (Server) et les objets connectés (Core) par la société Canonical. Ubuntu est disponible en deux versions, la version stable standard qui évolue tous les six mois, et une version LTS, pour Long Term Support (« Support long terme ») qui évolue tous les deux ans [1].

Lubuntu

Lubuntu est un projet de distribution GNU/Linux visant à produire une version dérivée d'Ubuntu, plus légère, plus économe en ressources matérielles, et moins consommatrice en énergie. Pour y parvenir, Lubuntu utilise l'environnement de bureau LXDE. Depuis la version 18.10, Lubuntu embarque l'environnement de bureau LXQt.

Initialement cette distribution existait principalement sous la forme du métapaquet lubuntu desktop utilisable depuis une distribution Ubuntu (ou dérivée). À partir de la version 10.04 (2010), une image ISO du live CD installable est disponible au téléchargement comme pour Ubuntu. Description Le 11 mai 2011, l'annonce de l'intégration de Lubuntu dans l'écosystème Ubuntu est faite lors d'une session de l'UDS. Cette intégration est devenue effective dès la sortie de la version 11.10 (2011) [1].

Kali linux

Kali Linux est une distribution GNU/Linux sortie le 13 mars 2013, basée sur Debian. La distribution a pris la succession de BackTrack. L'objectif de Kali Linux est de fournir une distribution regroupant l'ensemble des outils nécessaires aux tests de sécurité d'un système d'information, notamment le test d'intrusion. Depuis la version 2016.2, Kali Linux est disponible pré-installée avec de nombreux environnements de bureau. On retrouve : GNOME, KDE, LXDE, MATE, Enlightenment et Xfce, à choisir lors du téléchargement [1].

5.2.4 Surveillance de trafic et gestion de flux malveillants :

Wireshark

Wireshark est un analyseur de paquets libre et gratuit. Il est utilisé dans le dépannage et l'analyse de réseaux informatiques, le développement de protocoles, l'éducation et la rétro-ingénierie. Wireshark utilise la bibliothèque logicielle Qt pour l'implémentation de son interface utilisateur et pcap pour la capture des paquets ; il fonctionne sur de nombreux environnements compatibles UNIX comme GNU/Linux, FreeBSD, NetBSD, OpenBSD ou Mac OSX, mais également sur Microsoft Windows. Il existe aussi entre autres une version en ligne de commande nommé TShark. Ces programmes sont distribués gratuitement sous la licence GNU General Public License [1].

Nmap

Nmap est un scanner de ports libre créé par Fyodor et distribué par Insecure.org. Il est conçu pour détecter les ports ouverts, identifier les services hébergés et obtenir des informations sur le système d'exploitation d'un ordinateur distant. Ce logiciel est devenu une référence pour les administrateurs réseaux car l'audit des résultats de Nmap fournit des indications sur la sécurité d'un réseau. Il est disponible sous Windows, Mac OS X, Linux, BSD et Solaris. Le code source de Nmap est disponible sous la licence GNU GPL [1].

GNS3

Graphical Network Simulator (abrégié en GNS3) est un émulateur de logiciel réseau publié pour la première fois en 2008. Il permet la combinaison de dispositifs virtuels et réels, utilisés pour simuler des réseaux complexes. Il utilise le logiciel d'émulation Dynamips pour simuler Cisco IOS .

GNS3 est utilisé par de nombreuses grandes entreprises, dont Exxon , Walmart , ATT et la NASA , et est également populaire pour la préparation des examens de certification professionnelle en réseau. En 2015, le logiciel a été téléchargé 11 millions de fois.

Snort

Snort est un logiciel de détection et de prévention d'intrusion réseau open source. Il permet d'analyser le trafic réseau de type IP en temps réel et de détecter une grande variété d'attaques (scan de port, ...), grâce à sa capacité d'analyses de protocole et de recherche de correspondances de contenu. Actuellement maintenu par Sourcière (racheté par Cisco en 2013), Snort a été à l'origine développé par Martin Roesch en 1998. Très populaire et exploité par des millions d'utilisateurs de par le monde, Snort est considéré de facto comme un standard de fait. Snort est compatible avec plusieurs systèmes d'exploitation tels que Linux, Mac OS, FreeBSD, Open BSD, UNIX et Windows et il est supporté par une grande communauté open source [1].

Architecture du Snort :

- Un noyau de base : (PacketDecoder) au démarrage, ce noyau charge un ensemble de règles, les compile, les optimise et les classe. Durant l'exécution, le rôle principal du noyau est la capture des paquets. Une série de pré-processeurs : (Détection Engine) ces derniers améliorent les possibilités de SNORT en matière d'analyse et de reconstitution du trafic capturé. Ils reçoivent les paquets directement capturés et décodés, les retravaillent éventuellement puis les fournissent au moteur de recherche de signatures pour les comparer avec la base des signatures.

- Une série de « Detection plugins » : Ces analyses se composent principalement de comparaison entre les différents champs des headers des protocoles (IP, ICMP, TCP et UDP) par rapport à des valeurs précises.

- Une série de « output plugins » : permet de traiter cette intrusion de plusieurs manières : envoie vers un fichier log, envoie d'un message d'alerte vers un serveur syslog, stocker cette intrusion dans une base de données SQL [2].

Snort peut être configuré pour fonctionner principalement en trois modes opérationnels : Le mode NIDS : dans ce mode, Snort agit comme un détecteur d'intrusion réseau en analysant le trafic réseau. Ensuite, il compare ce trafic à des règles déjà définies par l'administrateur réseau et établit des actions à exécuter ;

- Le mode sniffer : c'est le mode qui permet à Snort de lire les paquets circulant sur le réseau et les afficher en continu sur un écran ou les enregistrer dans un fichier spécifique ;

- Le mode packet logger : dans ce mode Snort journalise le trafic réseau dans des répertoires sur le disque. Dans nos travaux, le mode NIDS est le plus approprié puisque nous allons utiliser Snort pour surveiller plusieurs interfaces réseaux. Ce qui nous permettra d'analyser le trafic sur l'ensemble des éléments du réseau, selon des règles spécifiques préalablement définies. Dans Snort, les règles sont définies comme des chaînes de texte brut composées de différentes parties (par exemple : action, protocole, adresse IP source et destination, ports source et de destination et options de messages et de règles pour les administrateurs). Cette flexibilité est l'une des principales raisons pour lesquelles nous avons choisi Snort, car il nous donne la possibilité d'écrire facilement des règles pour tout protocole si nous comprenons correctement son architecture et les vulnérabilités.

5.3 Implémentation

Comme l'indique la figure ci-dessous, notre maquette d'implémentation est réalisée entièrement en environnement virtuel avec des outils open source.

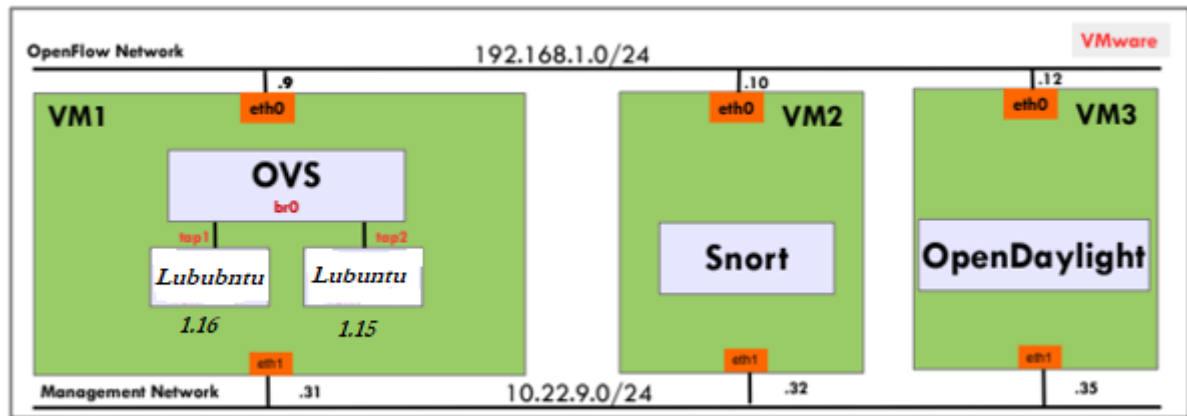


FIGURE 5.2 – Maquette de mise en œuvre de sécurisation d'un réseau avec du SDN.

5.3.1 Installation d'OpenDaylight

Le projet OpenDaylight (ODL) est une plate-forme collaborative et open source pour accélérer l'adoption et l'innovation de la mise en réseau logiciel (SDN) et de la visualisation des fonctions réseaux (NFV). ODL est un logiciel basé sur Java et pris en charge par l'industrie, géré par le consortium Linux Fondation avec près de 50 entreprises membres, dont Brocade, Cisco, Citrix, Dell, Ericsson, HP, IBM, Juniper, Microsoft et Red Hat. La mission d'ODL est de créer une communauté collaborative qui partage et contribue au succès et à l'adoption du SDN.

Pour expérimenter notre solution, nous avons créé une machine virtuelle de 2 CPU et 2GB de RAM avec un système d'exploitation Ubuntu 20.04 sur la plateforme VMware. Ensuite OpenDaylight nécessite un environnement d'exécution Java (JRE) pour s'exécuter. OpenDaylight peut tirer parti d'un JRE autonome sur le JRE regroupé dans un kit de développement logiciel Java .

La commande suivante permet d'installer JAVA 8 JRE.

```
lssam@ubuntu:~$ sudo apt-get -y install openjdk-8-jre
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

FIGURE 5.3 – Installation de JAVA 8 JRE.

La commande update-alternatives pour définir Java par défaut sur JAVA 8. update-alternatives présente une liste des versions Java installées et nous permet de sélectionner la version par défaut souhaitée. Si update-alternatives fournit une liste de versions, sélectionnez JAVA 8 dans la liste.

```

issam@ubuntu:~$ sudo update-alternatives --config java
There is only one alternative in link group java (providing /usr/bin/java): /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java
Nothing to configure.
issam@ubuntu:~$

```

FIGURE 5.4 – La sélection du version JAVA par défaut

Récupérons le chemin complet de votre exécutable JAVA . Si nous avons perdu la trace, nous pouvons exécuter la commande suivante :

```

issam@ubuntu:~$ ls -l /etc/alternatives/java
lrwxrwxrwx 1 root root 46 Jun 10 02:35 /etc/alternatives/java -> /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java
issam@ubuntu:~$

```

FIGURE 5.5 – Le chemin complet de votre exécutable JAVA.

Pour définir (et conserver) la valeur de JAVA HOME, modifions notre fichier de ressources BASH.

```

issam@ubuntu:~$ sudo echo 'export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre' >> ~/.bashrc
issam@ubuntu:~$ source ~/.bashrc
issam@ubuntu:~$

```

FIGURE 5.6 – La valeur du JAVA HOME.

```

issam@ubuntu:~$ echo $JAVA_HOME
/usr/lib/jvm/java-8-openjdk-amd64/jre
issam@ubuntu:~$

```

FIGURE 5.7 – Vérification de valeur du JAVA HOME.

Nous avons installé sur cette machine un contrôleur SDN de type OpenDaylight version NEXUS.

Nous avons téléchargé OpenDaylight version NEXUS, avec des privilèges administrateur.

```

issam@ubuntu:~$ curl -XGET -O https://nexus.opendaylight.org/content/repositories/opendaylight.release/org/opendaylight/integration/karaf/0.12.2/karaf-0.12.2.zip
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  T
ime      Time          Current             Dload  Upload   Total   S
pent     Left   Speed
  0     0     0     0     0     0     0     0  --:--:--  --:
  0     0     0     0     0     0     0     0  --:--:--  --:
  0     0     0     0     0     0     0     0  --:--:--  0:
  0 260M     0 7864   0     0    2572     0 29:28:08  0:
  0 260M     0 143k   0     0   40592     0  1:52:02  0:

```

FIGURE 5.8 – Téléchargement OpenDaylight version NEXUS.

OpenDaylight peut être initialisé en exécutant le script `karaf.sh` à partir de la ligne de commande, comme Indique la figure 5.9.

[illegible]

FIGURE 5.9 – Ecran de lancement OpenDaylight.

Toutes les distributions OpenDaylight sont livrées sans aucune fonctionnalité activée par défaut. En revanche, pour installer une fonctionnalité il faut lancer la commande `feature :install`.

nous avons ajouté les fonctionnalités telles que `odl-l2switch` `switch`, `odl-dluxe-all` et `odl-restconf` pour supporter les commutateurs de niveau 2/3, l'interface web et communiquer avec les applications via l'API REST. Il est aussi important d'activer la version OpenFlow 1.3 en ajoutant l'option `-of13` sur le fichier du script de lancement, car la version 1.0 de OpenFlow est par défaut implémentée sur le contrôleur OpenDaylight. OpenDaylight fournit plusieurs types de fonctionnalités à utiliser selon le besoin.

Pour prendre une décision d'acheminement de niveau 2/3 du modèle OSI, le contrôleur OpenDaylight connaît la topologie du réseau, ainsi que les équipements qui sont connectés avec leurs identifiants (adresses IP et adresses MAC). En utilisant OpenFlow 1.3, le contrôleur OpenDaylight configure un commutateur OVS, gère et met à jour le réseau OpenFlow.

Le lien <http://192.168.1.12:8181/index.html> permet d'accéder à OpenDaylight via une interface de navigation Web. Les accès par défaut (identifiant et mot de passe) de l'interface graphique du contrôleur sont admin/admin. Ces identifiants peuvent être personnalisés à

partir de cette même interface. La figure 5.10 ci-dessous montre l'environnement graphique d'OpenDaylight version NEXUS.

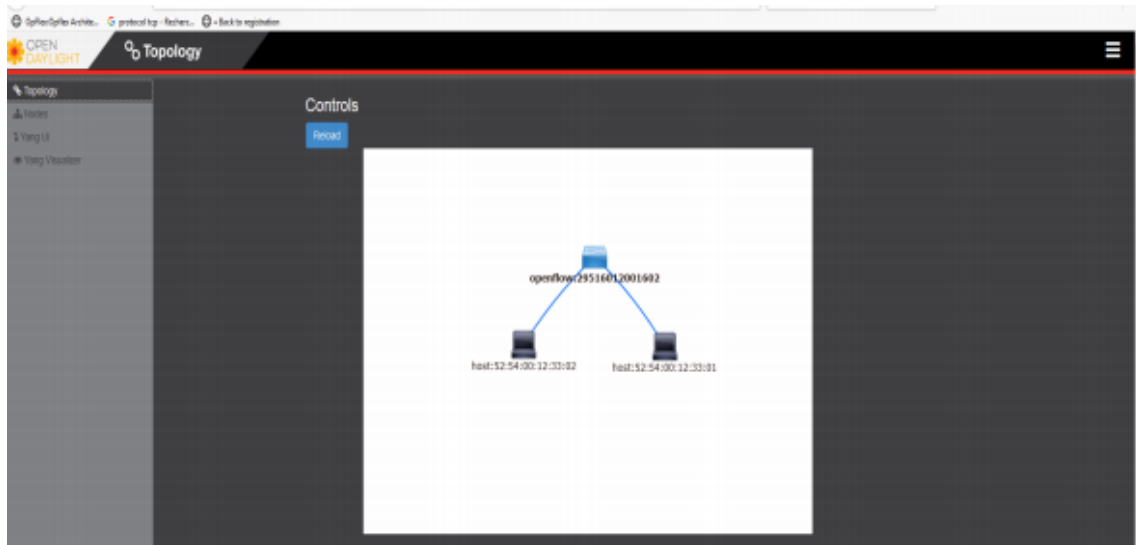


FIGURE 5.10 – Interface graphique d'OpenDaylight NEXUS.

Après cette étape d'installation du logiciel contrôleur, nous allons, dans le paragraphe suivant, simuler le réseau OpenFlow à gérer par OpenDaylight.

5.3.2 Installation d'un commutateur virtuel OpenFlow (OVS)

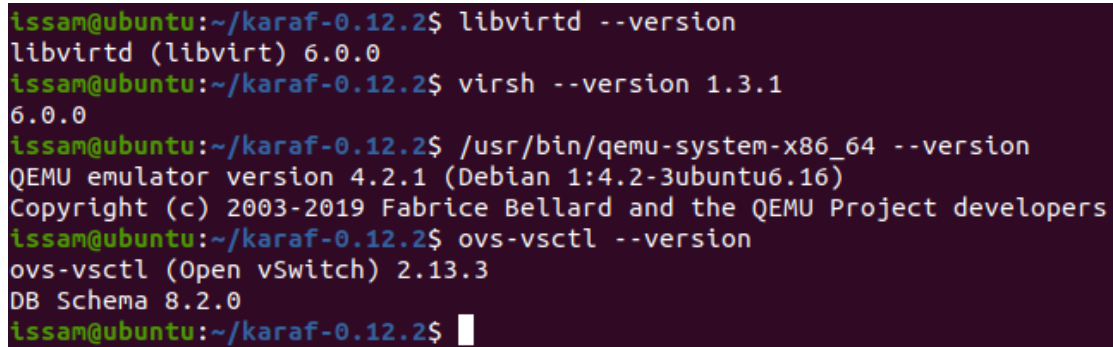
Nous avons créé une deuxième machine virtuelle avec un système d'exploitation Ubuntu 20.04, 2 CPU virtuel et 2GB de RAM sur une plateforme VMware. Sur cette machine, nous avons installé un commutateur virtuel compatible OpenFlow 1.3 (OVS version 2.6.0).

```
issam@ubuntu:~/karaf-0.12.2$ sudo apt-get install qemu-kvm libvirt-bin ubuntu-v
m-builder bridge-utils
[sudo] password for issam:
Reading package lists... Done
Building dependency tree
```

FIGURE 5.11 – Installation de KVM.

```
issam@ubuntu:~/karaf-0.12.2$ sudo apt-get install openvswitch-switch
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

FIGURE 5.12 – Installation de OVS.



```
lssam@ubuntu:~/karaf-0.12.2$ libvirtd --version
libvirtd (libvirt) 6.0.0
lssam@ubuntu:~/karaf-0.12.2$ virsh --version 1.3.1
6.0.0
lssam@ubuntu:~/karaf-0.12.2$ /usr/bin/qemu-system-x86_64 --version
QEMU emulator version 4.2.1 (Debian 1:4.2-3ubuntu6.16)
Copyright (c) 2003-2019 Fabrice Bellard and the QEMU Project developers
lssam@ubuntu:~/karaf-0.12.2$ ovs-vsctl --version
ovs-vsctl (Open vSwitch) 2.13.3
DB Schema 8.2.0
lssam@ubuntu:~/karaf-0.12.2$
```

FIGURE 5.13 – Vérification et comparaison des versions des packages.

L'OVS est une implémentation logicielle open source d'un commutateur Ethernet avec la particularité d'être multicouche et distribué. Il est conçu pour fonctionner comme un commutateur de niveau 2/3 du modèle OSI dans les environnements des machines virtuelles supportant différents protocoles et standards, dont le protocole OpenFlow. Dans nos travaux, il nous a permis de faire communiquer les machines virtuelles clientes entre elles.

Nous avons ensuite créé des machines virtuelles avec le système d'exploitation Lubuntu (une distribution légère de Linux) 256 MO RAM . Puis on va interconnecter les machines virtuelles Lubuntu entre elles et créer la liaison entre le commutateur OpenFlow et le contrôleur OpenDaylight, pour permettre à ce dernier de contrôler le réseau via le protocole OpenFlow.

5.3.3 Implémentation d'un pare-feu basé sur SDN

Un pare-feu est utilisé comme barrière pour protéger les ordinateurs en réseau en bloquant le trafic réseau malveillant généré par les virus et les vers.

J'ai implémenté un pare-feu de couche 2 qui s'exécute parallèlement au module basé sur l'adresse physique. L'application pare-feu est fournie avec une liste de paires d'adresses MAC, c'est-à-dire une liste de contrôle d'accès (ACL). Lorsqu'une connexion est établie entre le contrôleur et le commutateur, l'application installe des entrées de règle de flux statique dans la table OpenFlow pour désactiver toute communication entre chaque paire MAC.

Exemples de quelques règles de pare-feu

Sur le commutateur spécifié, bloquez tout le trafic provenant de l'hôte 192.168.1.33 :

```
1 | 00-00-02-00-00-00-00-00 BLOCK srcip=192.168.1.33|
```

Au niveau du commutateur spécifié, bloquez tout le trafic provenant de l'hôte 192.168.1.33, si le TOS du paquet est marqué 32 et qu'il est destiné à 192.168.1.16 :

```
2 | 00-00-03-00-00-00-00-00 BLOCK srcip=192.168.1.33 dstip=192.168.1.16 tos=32
```

Au niveau du commutateur spécifié, redirigez le trafic destiné à 192.168.1.16 et, à la place, envoyez-le à 192.168.1.17 :

```
3 | 00-00-01-00-00-00-00-00 REDIRECT dstip=192.168.1.16 TO 192.168.1.17|
```

Au niveau du commutateur spécifié, marquez le champ TOS de tous les paquets envoyés par 192.168.1.33 avec la valeur 40 :

```
4 | 00-00-04-00-00-00-00-00 MARK srcip=192.168.1.33| TOS 40
```

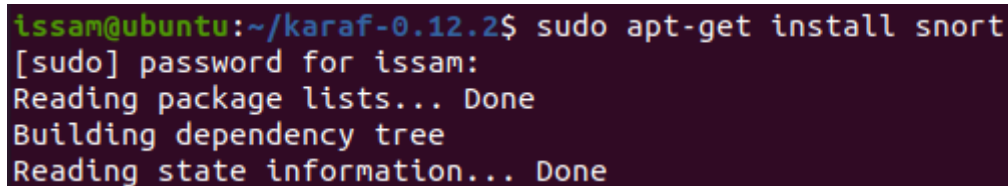
5.3.4 Mise en place de Snort

Afin de détecter les menaces, nous avons utilisé un IDS de type Snort. C'est un logiciel de détection d'intrusion réseau open source qui permet d'analyser le trafic réseau de type IP en temps réel et de détecter une grande variété d'attaques (scan de port...), grâce à sa capacité d'analyse de protocole et de recherche de correspondance de contenu.

Dans nos travaux, nous avons utilisé Snort en mode NIDS, approprié pour surveiller plusieurs interfaces réseau. Dans ce mode, Snort agit comme un détecteur d'intrusion réseau en analysant le trafic réseau et compare ce trafic à des règles déjà définies par l'administrateur réseau, pour établir des actions à exécuter. Les règles sont définies comme des chaînes de texte brut composée de différentes parties (par exemple : action, protocole, adresse IP source et destination, ports source et de destination, et options de messages et de règles pour les administrateurs).

Pour intégrer Snort dans le réseau, nous avons créé une troisième machine virtuelle avec un système d'exploitation Ubuntu 20.04, 2 CPU virtuel et 2 GB de RAM sur laquelle nous avons installé le logiciel Snort version 2.9 .

Nous allons installer Snort, pour cela tapez la commande suivante :



```
issam@ubuntu:~/karaf-0.12.2$ sudo apt-get install snort
[sudo] password for issam:
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

FIGURE 5.14 – Installation de SNORT.

Nous devons ensuite renseigner l'interface sur laquelle l'outil écouterait le réseau, nous sélectionnons l'interface reliée à notre réseau local.

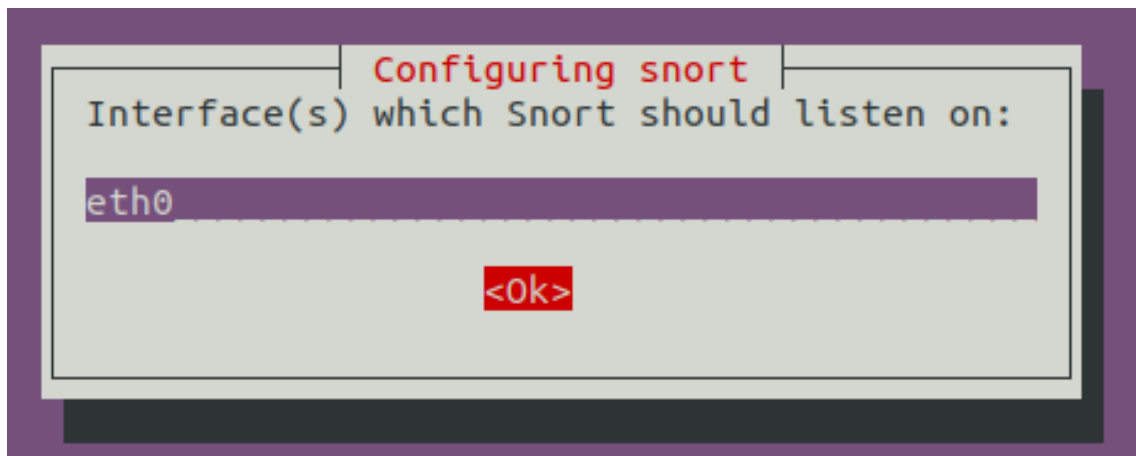


FIGURE 5.15 – L'interface sur laquelle l'outil écouterait le réseau.

Pour utiliser les règles présentes sur le site de Snort, nous avons téléchargé le fichier correspondant à la version installée. Puis nous avons Décompressé le fichier.

```
lssam@ubuntu:~/Desktop$ sudo tar zxvf community-rules.tar.gz
community-rules/
community-rules/community.rules
community-rules/VRT-License.txt
community-rules/LICENSE
community-rules/AUTHORS
community-rules/snort.conf
community-rules/sid-msg.map
lssam@ubuntu:~/Desktop$
```

FIGURE 5.16 – Décompression des règles présentes sur le site de Snort.

Nous allons dans le dossier décompressé et nous déplaçons ensuite le fichier "community.rules" dans le répertoire contenant les règles de Snort :

```
lssam@ubuntu:~/Desktop/community-rules$ ls
AUTHORS  community.rules  LICENSE  sid-msg.map  snort.conf  VRT-License.txt
lssam@ubuntu:~/Desktop/community-rules$ sudo mv community.rules /etc/snort/rules
lssam@ubuntu:~/Desktop/community-rules$
```

FIGURE 5.17 – Déplacer le fichier "community.rules".

Nous allons maintenant indiquer à l'outil qu'il doit prendre en compte le fichier « community.rules » pour générer des alertes. Nous n'avions pas eu à faire cette opération pour la création de l'alerte du précédent point, car celle-ci était déjà présente dans le fichier de configuration par défaut. Nous utilisons la commande suivante :

Ajoutons la ligne suivante au fichier

```
GNU nano 4.8 /etc/snort/snort.conf
# include $SO_RULE_PATH/web-iis.rules
# include $SO_RULE_PATH/web-misc.rules

# Event thresholding or suppression commands. See threshold.conf
include threshold.conf

include $RULE_PATH/community.rules
```

FIGURE 5.18 – Fichier /etc/snort/snort.conf

Nous allons vérifier que notre alerte fonctionne bien. Nous allons Lancer l'outil Snort avec la commande suivante :

```
lssam@ubuntu:~/karaf-0.12.2$ sudo snort -A console -i ens33 -u snort -c /etc/snort/snort.conf
Running in IDS mode
```

FIGURE 5.19 – Fonctionnement des alertes

Nous allons Lancer un ping a partir de n'importe quelle machine de notre réseau, comme expliqué précédemment Snort est un sniffer réseau, il va aspirer l'ensemble du trafic de notre réseau. nous devrions voir l'alerte que nous venons de créer apparaître :

```
07/09-04:01:56.048105  [**] [1:1917:6] SCAN UPnP service discover attempt [**]  
[Classification: Detection of a Network Scan] [Priority: 3] {UDP} 169.254.112.9  
2:56305 -> 239.255.255.250:1900  
07/09-04:01:59.049139  [**] [1:1917:6] SCAN UPnP service discover attempt [**]  
[Classification: Detection of a Network Scan] [Priority: 3] {UDP} 169.254.112.9  
2:56305 -> 239.255.255.250:1900
```

FIGURE 5.20 – Fonctionnement des alertes

Les informations sur les alertes générées par Snort sont enregistrées sous forme de fichier logs dans un répertoire sur le serveur hébergeant Snort.

```
issam@ubuntu:/var/log/snort$ sudo snort -r /var/log/snort/snort.log.1625828448  
Running in packet dump mode  
  
--== Initializing Snort ==--  
Initializing Output Plugins!  
pcap DAQ configured to read-file.  
Acquiring network traffic from "/var/log/snort/snort.log.1625828448".
```

FIGURE 5.21 – Exemple d'un fichier logs.

Maintenant on va extraire et analyser les informations sur les logs générés par Snort et d'envoyer une règle de sécurité au contrôleur OpenDaylight, pour désinstaller un flux malveillant via l'API REST en temps réel. L'API REST permet l'échange d'informations d'une manière simple entre un client et un serveur basé sur le protocole HTTP.

5.3.5 La sécurité du lien entre le contrôleur OpenDaylight et l'OVS

En utilisant chiffrement OpenSSL, nous avons généré un keyStore, un fichier contenant les clés privées et publiques du contrôleur. Ensuite, le fichier de clés est importé dans un fichier de clés au format JKS, adapté pour être configuré sur le fichier de configuration OpenFlow du contrôleur OpenDaylight comme l'indique la figure 5.22.

```
<module>
  <type xmlns:prefix="urn:opendaylight:params:xml:ns:yang:openflow:switch:connection:provider:impl">prefix:openflow-switch-connection-provider-impl</type>
  <name>openflow-switch-connection-provider-default-impl</name>
  <port>6633</port>
  <switch-idle-timeout>15000</switch-idle-timeout>
  <transport-protocol>TLS</transport-protocol>
  <tls>
    <keystore>configuration/ssl/ctf.jks</keystore>
    <keystore-type>JKS</keystore-type>
    <keystore-path-type>PATH</keystore-path-type>
    <keystore-password>opendaylight</keystore-password>
    <truststore>configuration/ssl/truststore.jks</truststore>
    <truststore-type>JKS</truststore-type>
    <truststore-path-type>PATH</truststore-path-type>
    <truststore-password>opendaylight</truststore-password>
    <certificate-password>opendaylight</certificate-password>
    <cipher-suites>TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384</cipher-suites>
    <cipher-suites>TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384</cipher-suites>
    <cipher-suites>TLS_DHE_DSS_WITH_AES_256_GCM_SHA384</cipher-suites>
    <cipher-suites>TLS_DHE_RSA_WITH_AES_256_GCM_SHA384</cipher-suites>
  </tls>
</module>

<module>
  <type xmlns:prefix="urn:opendaylight:params:xml:ns:yang:openflow:switch:connection:provider:impl">prefix:openflow-switch-connection-provider-impl</type>
  <name>openflow-switch-connection-provider-legacy-impl</name>
  <port>6653</port>
  <switch-idle-timeout>15000</switch-idle-timeout>
  <transport-protocol>TLS</transport-protocol>
  <tls>
    <keystore>configuration/ssl/ctf.jks</keystore>
    <keystore-type>JKS</keystore-type>
    <keystore-path-type>PATH</keystore-path-type>
    <keystore-password>opendaylight</keystore-password>
    <truststore>configuration/ssl/truststore.jks</truststore>
    <truststore-type>JKS</truststore-type>
    <truststore-path-type>PATH</truststore-path-type>
    <truststore-password>opendaylight</truststore-password>
    <certificate-password>opendaylight</certificate-password>
    <cipher-suites>TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384</cipher-suites>
    <cipher-suites>TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384</cipher-suites>
    <cipher-suites>TLS_DHE_DSS_WITH_AES_256_GCM_SHA384</cipher-suites>
    <cipher-suites>TLS_DHE_RSA_WITH_AES_256_GCM_SHA384</cipher-suites>
  </tls>
</module>
```

FIGURE 5.22 – Fichier de configuration OpenFlow pour supporter TLS.

Pour créer la clé publique, la clé privée et le certificat pour l'OVS, la commande `ovs-pki req + signe sc` permet de créer les fichiers de l'autorité de certification (CA).

L'utilisation de la commande `ovs-vsctl set-ssl` pour pointer sur le répertoire des clés publiques et privées générées précédemment, permet de configurer l'OVS avec cryptage SSL. En outre, il est nécessaire d'ajouter le certificat de contrôleur `cacert.pem` qui autorise la connexion au contrôleur ODL en vérifiant la signature de ce certificat (CA). Ainsi, le chiffrement en SSL/TLS de la connexion entre OVS et le contrôleur OpenDaylight fonctionne dans les deux sens. D'où la sécurité du canal entre les deux éléments du réseau SDN. En revanche, les communications SSL/TLS sont susceptibles d'être interceptées et modifiées via une attaque de type man-in-the-middle en cas de fuite de la clé privée. C'est pourquoi, il est important de conserver toutes les clés privées dans un espace sécurisé, car en disposant de la clé privée correspondante, une personne malintentionnée et ayant les compétences techniques peut déchiffrer un message crypté avec la clé privée.

5.3.6 L'authentification et le chiffrement par poly1305 et AES-128

Comme l'indique la figure ci-dessous, un fichier généré par capteur cardiofréquence et Arduino, il est maintenant sur la machine Lubuntu. Après l'authentification entre la machine Lubuntu et la passerelle avec l'algorithme poly 1305, on va chiffrer ce message avec l'algorithme légère AES 128 pour garantir la confidentialité.



FIGURE 5.23 – Fichier généré par capteur cardiofréquencemètre.

L'authentification POLY1305

L'authentification pour un système informatique est un processus permettant au système de s'assurer de la légitimité de la demande d'accès faite par une entité (être humain ou un autre système...) afin d'autoriser l'accès de cette entité à des ressources du système (systèmes, réseaux, applications...) conformément au paramétrage du contrôle d'accès. L'authentification permet donc, pour le système, de valider la légitimité de l'accès de l'entité, ensuite le système attribue à cette entité les données d'identité pour cette session (ces attributs sont détenus par le système ou peuvent être fournis par l'entité lors du processus d'authentification). C'est à partir des éléments issus de ces deux processus que l'accès aux ressources du système pourra être paramétré (contrôle d'accès).

La première étape : la génération du MAC

```
from Crypto.Hash import Poly1305
from Crypto.Cipher import AES
from binascii import unhexlify

file = open ("Heart_Rate_Sensor.txt","r")
contents = file.read()
contents = bytes(contents,"ascii")

secret = b 'thirtytwo very very secret bytes'
mac = poly1305.new(key=secret,cipher=AES)
mac.update(contents)
print("Nonce: ", mac.nonce.hex())
print("MAC: ", mac.hexdigest())
file.close()
```

FIGURE 5.24 – Le script python pour la génération d'une balise MAC .

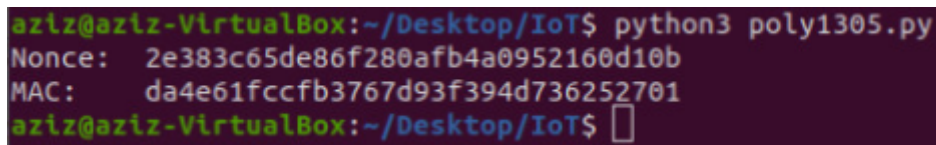
- key (bytes / bytearray / memoryview) - La clé de 32 octets pour l'objet Poly1305.

- cipher (module from Crypto.Cipher) - L'algorithme de chiffrement à utiliser pour dériver la paire de clés Poly1305 (r,s). Cela ne peut être que Crypto.Cipher.AES ou Crypto.Cipher.ChaCha20.

- nonce (bytes / bytearray / memoryview) - Facultatif. Valeur non répétable à utiliser pour le MAC de ce message. Il doit faire 16 octets pour AES et 8 ou 12 octets pour ChaCha20. S'il n'est pas passé, un nonce aléatoire est créé ; vous le trouverez dans l'attribut nonce du nouvel objet.

- data (bytes / bytearray / memoryview) - Facultatif. Le tout premier morceau du message à authentifier. Cela équivaut à un appel précoce à update().

La génération d'une balise MAC avec nonce par l'exécution de script sur la machine Ubuntu.



```
aziz@aziz-VirtualBox:~/Desktop/IoT$ python3 poly1305.py
Nonce:  2e383c65de86f280afb4a0952160d10b
MAC:    da4e61fccfb3767d93f394d736252701
aziz@aziz-VirtualBox:~/Desktop/IoT$
```

FIGURE 5.25 – La génération d'une balise MAC avec nonce.

La deuxième étape la validation de Mac :

```
from Crypto.Hash import Poly1305
from Crypto.Cipher import AES
from binascii import unhexlify

file = open ("Heart_Rate_Sensor.txt","r")
contents = file.read()
contents = bytes(contents,"ascii")

secret = b 'thirtytwo very very secret bytes'
mac = poly1305.new(key=secret,cipher=AES)
mac.update(contents)

nonce = unhexlify (b '2jh34B3v9ffd4gff8s7zz5rrt6y4u7j3
                    )
MAC = b 'dv2h5g56ze346hbnv534tydf34v4yl2h'

mac = poly1305.new(key=secret,nonce=nonce,cipher=AES,data=contents)

try:
    mac.hexverify(Mac)
    print ("the message is authentic")
except ValueError:
    print ("the message pr the key is wrong")
file.close()
```

FIGURE 5.26 – Le script python pour la validation de MAC.

Comme l'indique la figure ci-dessous, le MAC généré précédemment il est envoyé à la passerelle, maintenant c'est l'étape de vérifier ce MAC.



FIGURE 5.27 – Le fichier Mac générer.

```
aziz@aziz-VirtualBox:~/Desktop/Gateway$ python3 poly1305.py
The message is authentic
aziz@aziz-VirtualBox:~/Desktop/Gateway$
```

FIGURE 5.28 – La validation de Mac.

Comme l'indique la figure, le MAC est authentique donc l'authentification se fait de manière normale.

Le chiffrement AES-128

Le chiffrement (ou cryptage) est un procédé de cryptographie grâce auquel on souhaite rendre la compréhension d'un document impossible à toute personne qui n'a pas la clé de (dé)chiffrement. Ce principe est généralement lié au principe d'accès conditionnel.

Après l'authentification. On va passer au cryptage de fichier HeartRateSensor.txt avec AES-128, nous utilisons la commande openssl, le fichier de sortie : EncryptedMessage.dat

```
issam@ubuntu:~$ sudo openssl enc -aes-128-cbc -in Heart_Rate_Sensor.txt -out encrypted_message.dat
enter aes-128-cbc encryption password:
Verifying - enter aes-128-cbc encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
issam@ubuntu:~$
```

FIGURE 5.29 – Le cryptage de fichier avec la commande openssl (AES-128).

```
issam@ubuntu:~$ sudo openssl enc -aes-128-cbc -in Heart_Rate_Sensor.txt -out encrypted_message.dat
enter aes-128-cbc encryption password:
Verifying - enter aes-128-cbc encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
issam@ubuntu:~$
```

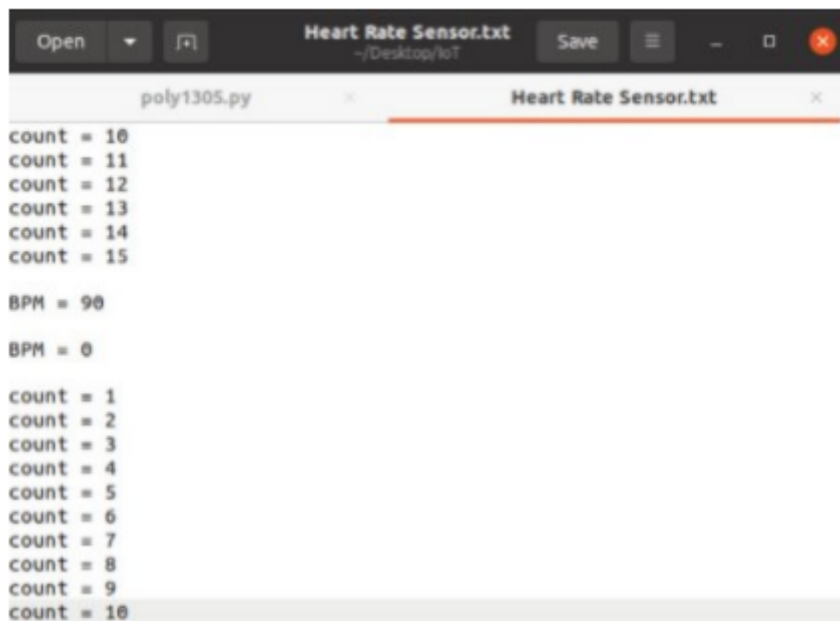
FIGURE 5.30 – Le cryptage de fichier avec la commande openssl (AES-128).

le décryptage de fichier avec la commande openssl (AES-128) :

```
issam@ubuntu:~$ sudo openssl enc -aes-128-cbc -in encrypted_message.dat -out Heart_Rate_Sensor.txt
enter aes-128-cbc encryption password:
Verifying - enter aes-128-cbc encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
issam@ubuntu:~$
```


FIGURE 5.31 – Le décryptage de fichier avec la commande openssl (AES-128)

Le contenu de fichier après le décryptage :



```
Open  [icon]  Heart Rate Sensor.txt  Save  [icon]  [icon]  [icon]
poly1305.py  x  Heart Rate Sensor.txt  x

count = 10
count = 11
count = 12
count = 13
count = 14
count = 15

BPM = 90

BPM = 0

count = 1
count = 2
count = 3
count = 4
count = 5
count = 6
count = 7
count = 8
count = 9
count = 10
```

FIGURE 5.32 – Le contenu de fichier après le décryptage.

5.4 Quelques exemples d'attaques simulées

Pour simuler une attaque et voir si Snort la détecte ou pas, nous avons installé sur une des machines virtuelles clientes l'outil Nmap. Ensuite, nous avons successivement lancé une attaque de déni de service, de scan de port et une usurpation d'adresse IP de la machine avec la commande Nmap spécifique au cas par cas.

5.4.1 Déni de services

L'objectif ici est de détecter et de bloquer les tentatives de saturation d'une machine cible par des attaques de type DoS avec le protocole ICMP. Nous avons procédé à l'envoi des requêtes ICMP à la deuxième machine de notre réseau, afin de voir si Snort a réagi en détectant les flux non désirés. Avec cet exemple, nous avons constaté que suite à cette requête ICMP, notre IDS Snort a détecté et enregistré un fichier log sur le répertoire spécifique du serveur Sort.

Ce type de tentative d'attaque peut rendre le contrôleur ou une machine indisponible pour ses utilisateurs. Il interrompt ou suspend temporairement ou indéfiniment les services du réseau. Avec la solution proposée, il est possible de bloquer la communication des nœuds malveillants de façon automatisée.

5.4.2 Scan de ports

Il s'agit dans ce cas de détecter les tentatives de scan de ports sur les protocoles TCP et UDP et de bloquer ces requêtes provenant d'une même source avec l'outil Nmap.

Nmap est un logiciel de scan de ports open source conçu pour détecter les ports ouverts et, plus généralement, obtenir des informations sur le système d'exploitation d'un ordinateur distant. Pour connaître les ports ouverts sur une machine, Nmap envoie un paquet sur tous les ports de la machine cible et analyse les réponses.

Pour simuler le scan de ports, nous avons installé l'outil Nmap sur une des machines host kali Linux de notre réseau. Ensuite, nous avons lancé un balayage de ports sur une des machines du réseau avec la commande spécifique (`nmap -p "*" Adresse Ip machine cible`) et de la même manière, Snort a détecté cette tentative d'attaque et a enregistré le log correspondant.

5.4.3 Usurpation d'adresse IP ou MAC

Cette usurpation est faite lorsqu'un attaquant malveillant essaye d'usurper une adresse MAC ou IP légitime afin d'envoyer des paquets vers le réseau, en utilisant une adresse de confiance. La réplique de l'adresse MAC/IP force les systèmes à croire que la source est digne de confiance. De la même manière que le scan de ports, nous avons expérimenté avec Nmap, et à travers la commande spécifique (`nmap spoof-mac adresseMAC de la machine cible ou AdresseIp machine cible`), l'usurpation d'adresse IP et d'adresse MAC et nous avons constaté que Snort a détecté la menace et a enregistré le log associé.

Nous avons remarqué que Snort a détecté l'ensemble des attaques et a enregistré les fichiers correspondants dans le répertoire des logs. Cette procédure peut être étendue à d'autres types de menaces plus complexes et intelligentes.

5.5 Conclusion

L'implémentation de notre approche a été, pour nous, l'occasion d'acquérir une expérience pratique avec le SDN et a été très utile pour le développement de notre solution de sécurisation des échanges dans un réseau IOT. Au cours de cette étude, nous nous sommes familiarisés avec les techniques d'installation et de la mise en service des solutions réseaux utilisant OpenFlow. et aussi sur l'amélioration de l'intégration des règles de sécurité de manière automatisée.

Conclusion et Perspectives

L'Internet des objets représente une idéologie globale qui supprime les frontières entre le monde physique et le monde virtuel, et qui couvre la plupart des systèmes informatiques et technologies d'information. En effet, l'IoT regroupe plusieurs domaines tel que les réseaux de capteurs sans fils, les villes intelligentes, les véhicules connectés, les systèmes de contrôle commande, etc. De ce fait, un système IoT est un système hétérogène utilisant différentes technologies, déployées sur des architectures et des plateformes différentes et implémentées sur des matériaux informatiques (hardware) très variés.

Généralement, il utilise des technologies de communications sans fil afin de connecter des objets intelligents et autonomes. Ces objets ont la capacité de rassembler, analyser, traiter, générer et échanger des informations afin de fournir des services avancés.

Cependant, les problèmes de sécurité informatique ralentissent considérablement l'évolution et le déploiement rapide de cette technologie de pointe. De plus, cette dernière, ne peut pas utiliser les protocoles de sécurité classiques (ex. TLS [45]) et les solutions existante car la plupart ne permettent pas d'assurer des bonnes performances ou bien ils ne sont pas adaptées aux capacités des objets qui sont généralement très limités terme de stockage de calcul et en énergie.

Le SDN est un paradigme qui fournit des innovations majeures en termes d'architecture et de gestion des réseaux, par la séparation entre le plan de contrôle et le plan de données dans les équipements réseaux, tels que des commutateurs et des routeurs. L'introduction d'un contrôleur externe permet de programmer les flux grâce aux protocoles OpenFlow et OpFlex.

En utilisant cette nouvelle approche de gestion des réseaux, nous avons proposé une solution de sécurisation des échanges dans un réseau en fonction des événements détectés. C'est dans ce contexte que nous avons mis en œuvre un réseau SDN, même si certains aspects techniques de ce paradigme est encore au stade de développement.

Nous avons aussi proposé une architecture de sécurité avec du SDN étendu aux objets connectés. En divisant un réseau en cluster et en utilisant plusieurs contrôleurs, il est possible non seulement d'équilibrer la charge des contrôleurs et d'éviter que le contrôleur ne devienne un point critique, mais aussi de gérer la sécurité de manière collaborative grâce aux East WestBound API.

Les résultats des expérimentations de mise en œuvre du SDN, et les différentes possibilités de définition et d'implémentation des règles de sécurité, ont prouvé l'efficacité du SDN par rapport aux mécanismes de gestion classique des réseaux, d'où l'importance du

SDN dans l'amélioration de l'administration et de la sécurité des réseaux IoT.

Une question que nous nous posons pour le long terme serait de savoir quels outils utiliser pour simuler des équipements IoT non IP , afin d'expérimenter un réseau IoT hybride IP et Non IP. Et comment en profitant les avantages offerts par le cloud computing pour évoluer les fonctionnalités assurées par les contrôleurs SDN ?

Liste des acronymes

ACL (Liste de contrôle d'accès)
AES (Advanced Encryption Standard)
CA (Certification Authority)
DES (Data Encryption Standard)
DoS (Denial of Service)
DDoS (Distributed Denial of Service)
ECC (Cryptographie à courbe elliptique)
GPS (Global Positioning System)
HE (Cryptage homomorphique)
IDS (Intrusion Detection System)
IoE (Internet of Everythings)
IoT (Internet of Things)
IPSO (Internet Protocol for Smart Objects)
IPS (Intrusion Prevention System)
IP (Internet Protocol)
IBSG (Internet Business Solutions Group)
IETF (Internet Engineering Task Force)
IIoT (Internet des objets industriel)
LG (Life's Good)
LPWAN (Low Power Wide Area Network)
M2M (Machine-to-Machine)
NIST (National Institute of Standards and Technology)
NIC (National Intelligence Council)
NFC (Near Field Communication)
ONF (Open Networking Foundation)
OVSDB (Open vSwitch Database)
OVS (Open vSwitch)
ODL (OpenDayLight)
P2P (Peer-to-peer)
PoW (Puissance de travail)
PoS (Puissance d'enjeu)
PBFT (Tolérance aux pannes byzantine pratique)
PIB (Produit intérieur brut)

PKI (Public Key infrastructure)
QoS (Quality of Service)
Qemu (Quick Emulator)
RFID (Radio Frequency IDentification)
RSA (Rivest-shamir-adleman)
RAM (Random Access Memory)
RFID (Radio Frequency IDentification)
SSL (Secure Sockets Layer)
SDN (Software-Defined Network)
STI (Systèmes de transport intelligents)
SE (Cryptage interrogeable)
TLS (Transport Layer Security)
V2V (Vehicle to Vehicle)
V2I (Vehicle to Infrastructure)
V2P (Vehicle to Pedestrian)
V2G (Vehicle to Grid)
VNC (Virtual Network Computing)
WSN (Wirless Sensor Networks)

Références

- [1] Postscapes, IoT Home Guide, Postscapes.com [Consulte : 3-01- ' 2017]. [Online] <http://www.postscapes.com/internet-of-things-award/connected-home-products/>.
- [2] J. Greenough. How the 'Internet of Things' will impact consumers, businesses, and governments in 2016 and beyond. Business Insider. [Consulte : 06-12-2016]. [Online]. <http://uk.businessinsider.com/how-the-internet-of-things-market-will-grow-2014-10>.
- [3] Announcing the Advanced Encryption Standard (AES). Technical report, November 2001. 91.
- [4] Fast Ethereum RPC client for testing and development. Online. Test RPC. <https://github.com/ethereumjs/testrpc>, 2015. 109, 128.
- [5] Securing the Internet of Things (IoT) : A Security Taxonomy for IoT 2018 17th. Syed Rizvi, Joseph Pfeffer III, Andrew Kurtz, Mohammad Rizvi. Department of Information Sciences and Technology, Pennsylvania State University, Altoona-PA, USA.
- [6] Le SDN pour les nuls. Jérôme Durand. Cisco Systems .11 rue Camille Desmoulins 92130 Issy-les-Moulineaux.
- [7] S. Wang, D. Li, and S. Xia, "The problems and solutions of network update in sdn : A survey," in 2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), April 2015, pp. 474–479.
- [8] F. Hu, Q. Hao, and K. Bao, "A survey on software-defined network and openflow : From concept to implementation," IEEE Communications Surveys Tutorials, vol. 16, no. 4, pp. 2181–2206, 2014.
- [9] C. Martinez, R. Ferro, and W. Ruiz, "Next generation networks under the sdn and openflow protocol architecture," in 2015 Workshop on Engineering Applications - International Congress on Engineering (WEA), Oct 2015, pp. 1–7.
- [10] D. Kreutz, F. M. V. Ramos, P. E. VerAssimo, C. E. Rothenberg, S. Azodolmolky, and ~ S. Uhlig, "Software-defined networking : A comprehensive survey," Proceedings of the IEEE, vol. 103, no. 1, pp. 14–76, Jan 2015.
- [11] Software Defined Networking (SDN) : Etat de L'art. Ihssane Choukri, Mohammed Ouzzif, Khalid Bouragba. Laboratoire RITM, ESTC, Université Hassan II, Casablanca, Maroc.
- [12] F. Benamrane, M. Ben mamoun, et R. Benaini, « Performances of OpenFlow-Based Software-Defined Networks : An overview », J. Netw., vol. 10, no 6, juin 2015
- [13] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang. " A survey on mobile edge networks : Convergence of computing, caching and communications ". IEEE

Access, 5 :6757–6779, 2017.

[14] H. Noura. "Adaptation of Cryptographic Algorithms According to the Applications Requirements and Limitations : Design, Analyze and Lessons Learned ". HDR dissertation, UNIVERSITY of PIERRE MARIE CURIE -Paris VI, 2016.

[15] M. Farooq, M. Waseem, A. Khairi, and S. Mazhar, "A Critical Analysis on the Security Concerns of Internet of Things (IoT)," *Perception*, vol. 111, 2015.

[16] R. Roman, P. Najera, and J. Lopez, "Securing the internet of things," *Computer*, vol. 44, 51-58, 2011.

[17] R. Roman, J. Zhou, and J. Lopez, "On the features and challenges of security and privacy in distributed internet of things," *Computer Networks*, vol. 57, 2266-2279, 2013.

[18] Rwan Mahmoud, Tasneem Yousuf, Fadi Aloul, Imran Zuolkernan, " Internet of Things (IoT) Security : Current Status, Challenges and Prospective Measures" Department of Computer Science et Engineering. American University of Sharjah, UAE.

[19] Syed Rizvi, Joseph Pfeffer III, Andrew Kurtz, Mohammad Rizvi, "Securing the Internet of Things (IoT) : A Security Taxonomy for IoT",¹ Department of Information Sciences and Technology, Pennsylvania State University, Altoona-PA, USA, 2018 17th IEEE International Conference on Trust, Security and Privacy In Computing And Communications/ 12th.

[20] Somia SAHRAOUI, "Mécanismes de sécurité pour l'intégration des RCSFs à l'IoT (Internet of Things) », Université de Batna 2 Faculté des Mathématiques et d'Informatique.

[21] Yasmine HARBI, " Security in Internet of Things ",PEOPLES' DEMOCRATIC REPUBLIC of ALGERIA ,Ministry of Higher Education and Scientific Research Ferhat Abbas University Setif 1 Faculty of Sciences Department of Computer Science.

[22] Goiuri Peralta, Raul G Cid-Fuentes, Josu Bilbao, and Pedro M Crespo. Homomorphic encryption and network coding in iot architectures : Advantages and future challenges. *Electronics*, 8(8) :827, 2019.

[23] Umasankararao Varri, Syamkumar Pasupuleti, and KV Kadambari. A scoping review of searchable encryption schemes in cloud computing : taxonomy, methods, and recent developments. *The Journal of Supercomputing*, 76(4) :3013–3042, 2020

[24] M. Ojo, D. Adami, and S. Giordano, "A sdn-iot architecture with nfv implementation," in 2016 IEEE Globecom Workshops (GC Wkshps), Dec 2016, pp. 1–6.

[25] P. Bull, R. Austin, E. Popov, M. Sharma, and R. Watson, "Flow based security for iot devices using an sdn gateway," in 2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud), Aug 2016, pp. 157–163

[26] C. Gonzalez, O. Flauzac, F. Nolot, and A. Jara, "A novel distributed sdn-secured architecture for the iot," in International Conference on Distributed Computing in Sensor Systems (DCOSS) Washington, DC, May 2016, pp. 244–249.

[27] C. Vandana, "Security improvement in iot based on software defined networking," in International Journal of Science, Engineering and Technology Research (IJSETR), vol. 5, 2016.

[28] Andis Arins, "Firewall as a service in SDN OpenFlow network," Faculty of Com-

puting, University of Latvia Riga, Latvia .

[29] Daniel J. Bernstein, “The Poly1305-AES message-authentication code” Department of Mathematics, Statistics, and Computer Science (M/C 249). The University of Illinois at Chicago Chicago, IL 60607-7045. djb@cr.yp.to

[30] <https://www.ietf.org/rfc/rfc4493.txt>

[31] <https://www.geeksforgeeks.org/hmac-algorithm-in-computer-network/>

[32] <https://searchsecurity.techtarget.com/definition/Advanced-Encryption-Standard/>

[33] N. Zope, S. Pawar, and Z. Saquib, “Firewall and load balancing as an application of sdn,” in 2016 Conference on Advances in Signal Processing (CASP), June 2016, pp. 354–359.

[34] T. V. Tran and H. Ahn, “A network topology-aware selectively distributed firewall control in sdn,” in 2015 International Conference on Information and Communication Technology Convergence (ICTC), Oct 2015, pp. 89–94.

[35] J. G. V. Pena and W. E. Yu, “Development of a distributed firewall using software defined networking technology,” in 2014 4th IEEE International Conference on Information Science and Technology, April 2014, pp. 449–452.

[36] T. Javid, T. Riaz, and A. Rasheed, “A layer2 firewall for software defined network,” in 2014 Conference on Information Assurance and Cyber Security (CIACS), June 2014, pp. 39–42.

[37] K. Kaur, K. Kumar, J. Singh, and N. S. Ghumman, “Programmable firewall using software defined networking,” in 2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom), March 2015, pp. 2125–2129.

[38] M. Suh, S. H. Park, B. Lee, and S. Yang, “Building firewall over the software-defined-network controller,” in 16th International Conference on Advanced Communication Technology, Feb 2014, pp. 744–748.

[39] Ahmad Khalil, " Gestion autonome de la qualité de service et de la sécurité dans un environnement Internet des objets " ;04/12/2019 UNIVERSITE BOURGOGNE FRANCHE-COMTE .

[40] Djamel Eddine Kouicem, "Sécurité de l'Internet des objets pour les systèmes de systèmes", 21 novembre 2019 Université de Technologie de Compiègne.

[41] Salim Mahamat Charfadine, "Gestion dynamique et évolutive de règles de sécurité pour l'Internet des Objets", 2 juillet 2019 UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE ÉCOLE DOCTORALE SCIENCES DU NUMÉRIQUE ET DE L'INGÉNIEUR .

[42] Jérôme Durand, "Le SDN pour les nuls", Cisco Systems ,11 rue Camille Desmoulins ,92130 Issy-les-Moulineaux

[42] Yasmine HARBI, "Security in Internet of Things", Ferhat Abbas University Setif 1.