Université de Montréal

Renforcement de la sécurité à travers les réseaux programmables

par

Zakaria Abou El Houda

Département d'informatique et de recherche opérationnelle Université de Montréal
Faculté des arts et des sciences

Thèse présentée en vue de l'obtention du grade de
Philosophiæ Doctor (Ph.D.)
en Informatique

September 15, 2021

Université de Montréal

Faculté des arts et des sciences

Cette thèse intitulée

Renforcement de la sécurité à travers les réseaux programmables

présentée par

Zakaria Abou El Houda

a été évaluée par un jury composé des personnes suivantes :

Louis Salvail
(président-rapporteur)

Abdelhakim Senhaji Hafid
(directeur de recherche)

Lyes Khoukhi
(codirecteur)

Ahmed Serhrouchni
(membre du jury)

Samuel Pierre
(membre du jury)

Marc Lemercier
(membre du jury)

Pascal Lorenz
(examinateur externe)

Stéphane Couture
(représentant du doyen de la FESP)

Résumé

La conception originale d'Internet n'a pas pris en compte les aspects de sécurité du réseau; l'objectif prioritaire était de faciliter le processus de communication. Par conséquent, de nombreux protocoles de l'infrastructure Internet exposent un ensemble de vulnérabilités. Ces dernières peuvent être exploitées par les attaquants afin de mener un ensemble d'attaques. Les attaques par déni de service distribué (Distributed Denial of Service ou DDoS) représentent une grande menace et l'une des attaques les plus dévastatrices causant des dommages collatéraux aux opérateurs de réseau ainsi qu'aux fournisseurs de services Internet.

Les réseaux programmables, dits Software-Defined Networking (SDN), ont émergé comme un nouveau paradigme promettant de résoudre les limitations de l'architecture réseau actuelle en découplant le plan de contrôle du plan de données. D'une part, cette séparation permet un meilleur contrôle du réseau et apporte de nouvelles capacités pour mitiger les attaques par déni de service distribué. D'autre part, cette séparation introduit de nouveaux défis en matière de sécurité du plan de contrôle.

L'enjeu de cette thèse est double. D'une part, étudier et explorer l'apport de SDN à la sécurité afin de concevoir des solutions efficaces qui vont mitiger plusieurs vecteurs d'attaques. D'autre part, protéger SDN contre ces attaques. À travers ce travail de recherche, nous contribuons à la mitigation des attaques par déni de service distribué sur deux niveaux (intra-domaine et inter-domaine), et nous contribuons au renforcement de l'aspect sécurité dans les réseaux programmables.

Mots-clés: SDN (architecture des réseaux d'ordinateurs); Réseaux à grande distance (informatique); Tests d'intrusion (informatique); Apprentissage supervisé (intelligence artificielle); Blockchains.

Abstract

The original design of Internet did not take into consideration security aspects of the network; the priority was to facilitate the process of communication. Therefore, many of the protocols that are part of the Internet infrastructure expose a set of vulnerabilities that can be exploited by attackers to carry out a set of attacks. Distributed Denial-of-Service (DDoS) represents a big threat and one of the most devastating and destructive attacks plaguing network operators and Internet service providers (ISPs) in a stealthy way.

Software defined networks (SDN), an emerging technology, promise to solve the limitations of the conventional network architecture by decoupling the control plane from the data plane. On one hand, the separation of the control plane from the data plane allows for more control over the network and brings new capabilities to deal with DDoS attacks. On the other hand, this separation introduces new challenges regarding the security of the control plane.

This thesis aims to deal with various types of attacks including DDoS attacks while protecting the resources of the control plane. In this thesis, we contribute to the mitigation of both intra-domain and inter-domain DDoS attacks, and to the reinforcement of security aspects in SDN.

Keywords: Software-defined networking (Computer network technology); Wide area networks (Computer networks); Penetration testing (Computer security); Supervised learning (Machine learning); Blockchains.

Table des matières

Liste des tableaux

Liste des figures

Liste des sigles et des abréviations

SDN	Software Defined Networks
DNS	Domain Name System
NTP	Network Time Protocol
DDoS	Distributed Denial of Service
UDP	User Datagram Protocol
API	Application Programming Interface
QOS	Quality Of Service
RFC	Requests For Comments
TCAM	Ternary Content Addressable Memory
AS	Autonomous System

Remerciements

Je tiens à remercier mes directeurs de recherche, le professeur Abdelhakim Senhaji Hafid et le professeur Lyes Khoukhi, pour leur grande disponibilité, leur confiance, et leur consignes précieuses. Leurs directives m'ont permis de mener à bien ce travail. J'aimerais également leur exprimer toute ma gratitude et ma reconnaissance pour leur contributions à ce travail.

Je remercie l'Université de Technologie de Troyes ainsi que l'Université de Montréal de m'avoir accueillie comme étudiant au doctorat. Je remercie également tous mes collègues au Laboratoire des Réseaux de Communications (LRC) et d'ERA. Je remercie tout particulièrement mes amis Hajar, Amir et Achraf pour leur encouragements tout au long de mes recherches doctorales.

Je tiens à présenter mon profond respect et remerciement aux membres du jury pour leur bienveillance à bien vouloir évaluer mon travail.

Enfin, je remercie mes parents, mon frère Younes et sa femme Salima, ma sœur Houda et ma tante Asma pour leur grand support, leur soutien, et leur encouragements.

À mon père Omar Abou El Houda,
À ma mère Latifa Derraji,
À toute ma famille.

Chapitre 1

Introduction

Avant d'entamer la réalisation de cette thèse, il est essentiel de la contextualiser afin d'avoir une vision claire sur le mécanisme de défense à mettre en place, les différents enjeux, ainsi que le paradigme SDN et sa puissance. C'est dans cette optique que s'inscrit ce chapitre qui sera abordé en plusieurs phases. Nous commençons d'abord par présenter le contexte generale de cette thèse. Par la suite, nous étudions les différents vecteurs d'attaques ainsi que les moyens disponibles pour contrer ces attaques. Nous aborderons les motivations et les objectifs de cette thèse. Nous présentons les contributions apportées par cette thèse. Enfin, nous présenterons l'organisation de cette thèse.

1.1. Contexte générale

Internet est un élément critique qui est devenu un réseau universel de communication. Il devient également la cible principale d'attaques. Il présente une sensibilité accrue à leurs impacts. Il ne se passe plus une semaine sans que l'on apprenne que tel organisme ou tel institut a subi de lourdes pertes financières en raison d'une déficience de la sécurité de son système d'informations. Par conséquent, les grands organismes ne peuvent plus ignorer ces risques et se croire à l'abri de telles menaces. Aujourd'hui, les attaques par déni de service distribué sont fréquentes, notamment du fait de la relative simplicité de leur mise en œuvre. Elles sont efficaces contre des cibles non sécurisées. Ces attaques détériorent la qualité de service (QoS) du réseau visé, épuisent les ressources matérielles et peuvent engendrer des pertes financières importantes à cause de l'interruption de service. Pour cette raison, il est nécessaire d'anticiper cette menace, d'établir une modélisation de tous les chemins qui peuvent réussir ces attaques, et de prendre un certain nombre de mesures techniques et organisationnelles afin d'y faire face. Les travaux de cette thèse s'inscrivent dans les thématiques des réseaux logiciels, dits Software-Defined Networking (SDN). SDN rend les réseaux actuels programmables et fournit plusieurs avantages telles que la connaissance globale, la

gestion centralisée et l'abstraction du réseau. SDN est un paradigme novateur qui permet de développer des nouveaux services de sécurité plus efficaces.

Dans cette section, nous introduisons le paradigme SDN. Par la suite, nous aborderons les avantages et les défis de cette approche. Et enfin, nous évoquons une brève présentation du protocole OpenFlow.

1.1.1. Réseaux programmables (Software-Defined Networking)

SDN est un nouveau paradigme qui sépare les fonctions de contrôle (control plane) et les fonctions de transfert de données du réseau (data plane), afin d'avoir une infrastructure physique exempte de tout service réseau. Il y a quatre composantes principales qui définissent l'architecture SDN: (1) la séparation du plan de contrôle du plan de données; (2) la programmabilité du réseau; (3) l'abstraction du réseau physique; et (4) la centralisation des fonctions de contrôle.

SDN se rapporte aux anciennes recherches sur les réseaux de futures. Par exemple, le réseau actif (active Networking (1990)) est un mode de communication permettant aux paquets circulant dans un réseau de modifier dynamiquement l'exploitation du réseau. Le réseau actif se compose: (1) un matériel actif, capable de faire le routage/la commutation; et (2) du code exécuté dans les paquets actifs. La différence principale entre le réseau actif et SDN est que le réseau actif met en place des calculs dans les paquets qui traversent le réseau, tandis que SDN découple le système qui prend les décisions du plan de données. Le réseau actif offre la possibilité d'effectuer des changements en temps réel dans le réseau. Cependant, il souffre de plusieurs problématiques: (1) pas d'application claire; (2) coût élevé du matériel; (3) problème de sécurité (code transporté en clair par les paquets); et (4) problème d'interopérabilité.

1.1.2. Avantages du SDN

Le découplage de plan de contrôle du plan de données permet de déployer des services de contrôle sur une plateforme dont les capacités sont plus grandes de celles des équipements de réseaux classiques (commutateurs, routeurs). Cette nouvelle architecture fournit: (1) la simplicité de l'architecture des réseaux en réduisant le nombre de plans de contrôle; (2) la flexibilité et la programmabilité (ajout de nouveaux services) grâce aux interfaces standards (protocole OpenFlow (OF)); (3) le traitement efficace de données; (4) le déploiement efficace de nouveaux services afin de pouvoir gérer la sécurité, la qualité de service, etc.; et (5) une innovation rapide.

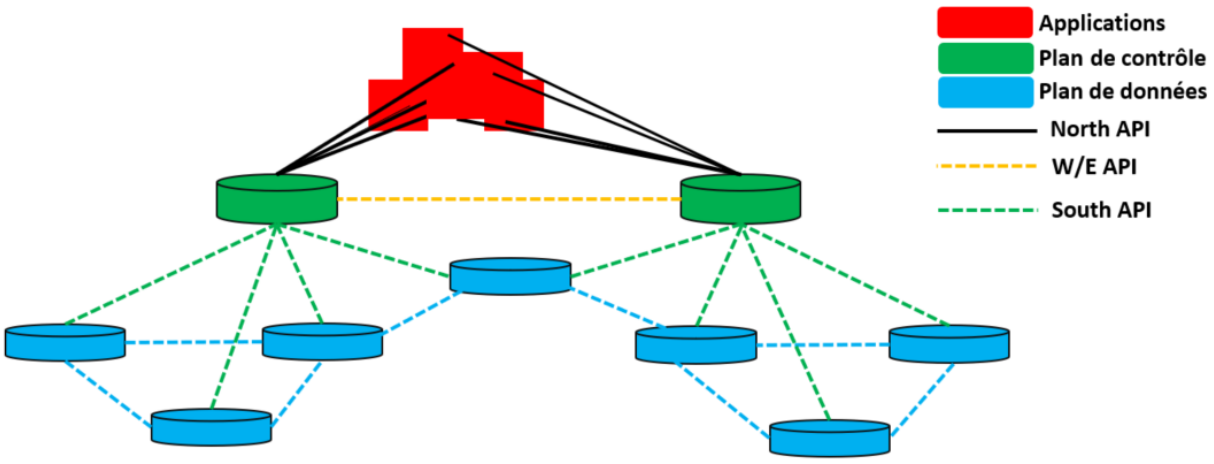


Figure. 1.1. Architecture du SDN

1.1.3. Différence entre SDN et les réseaux traditionnels

En déplaçant le plan de contrôle dans la partie logicielle, SDN permet un accès et une administration dynamique du réseau. SDN se caractérise par: (1) un modèle de réseau basé sur un contrôleur; (2) un contrôle du réseau directement programmable; (3) une gestion centralisée: l'intelligence du réseau est centralisée dans un logiciel appelé contrôleur SDN: qui maintient une vue globale du réseau; et (4) une configuration automatique, SDN permet aux administrateurs réseaux de configurer, administrer, sécuriser et optimiser les réseaux rapidement grâce à des programmes SDN dynamiques et automatisés. La Figure 1.1 illustre l'architecture élaborée par SDN.

La programmation des équipements réseau nécessite sur ces derniers la capacité de recevoir des directives de l'extérieur. Pour cela des interfaces de programmation (API (Application Programming Interface)) sont nécessaires. Il existe de nombreuses API permettant d'agir sur les équipements du plan de données. Nous nous intéressons au protocole OpenFlow qui est le standard du SDN.

1.1.4. OpenFlow

OpenFlow est un protocole standardisé qui est utilisé par SDN; Il permet la programmation d'équipements de commutations par des règles OF (ensemble de règles appliquées). Ces règles sont injectées par le contrôleur dans le plan de données. La standardisation d'OpenFlow est définie par l'Open Networking Foundation (ONF). Les principaux composants d'OpenFlow sont: (1) des commutateurs compatibles avec le protocole OpenFlow; (2) des serveurs capables d'exécuter le processus du contrôleur; et (3) une base de données contenant une vue globale de la topologie du réseau. La composante principale des commutateurs compatibles avec le protocole OpenFlow est la table de flux. La table de flux se compose

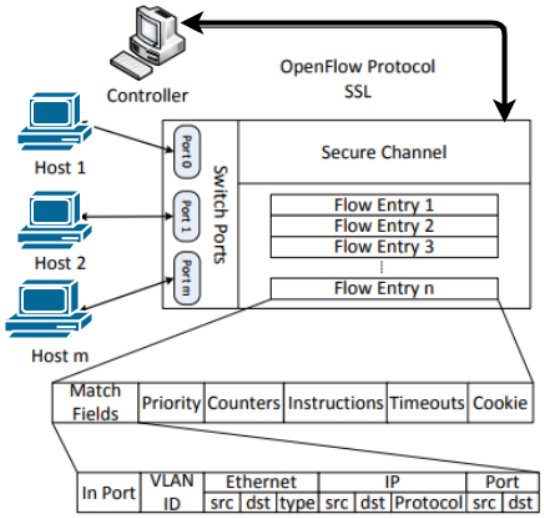


Figure. 1.2. Table de flux d'un commutateur OF

de plusieurs entrées comme montre la Figure 1.2; chaque entrée dans la table est composée des éléments suivants: (1) Match Fields: il s'agit des champs qui correspondent aux paquets et indiquent aux équipements de commutation (OF switch) comment un paquet doit être traité, la correspondance avec le champ d'en-tête d'un paquet IP offre plusieurs possibilités. En effet, il est tout à fait possible de chercher les correspondances avec les en-têtes classiques des paquets IP (adresses Mac, IP source, IP destination, Port source et destination, etc.). Les match Fields sont des champs de l'entête du paquet du niveau L1 à L4 et ils dépendent de la version d'OpenFlow; (2) Instructions: lorsqu'un paquet correspond à une entrée de flux, une fonction est exécutée. Cette dernière correspond à un ensemble d'action à effectuer comme: (a) transmettre un paquet à un port OpenFlow spécifié; (b) modifier les entêtes du paquet, les métas data, etc.; (c) envoyer le paquet au contrôleur; et (d) détruire le paquet; et (3) timeouts: ils représentent la durée de vie (hard-timeout) et le temps d'inactivité avant l'expiration (idle-timeout) des règles d'OpenFlow. Une entrée reste dans les tables de flux du commutateur OF jusqu'à ce que le délai soit terminé.

Le fonctionnement principale d'OpenFlow est décrit comme suit: à la réception d'un paquet, le commutateur commence d'abord par effectuer une recherche dans sa première table. Si un traitement dans son pipeline est défini, il effectue des recherches supplémentaires dans ses autres tables. Dans le cas où le commutateur ne trouve aucune règle, il rejette le paquet. Néanmoins, il est à noter qu'il peut exister une règle par défaut qui consiste à envoyer le paquet qui ne corresponds à aucune règle OF vers le contrôleur SDN, au lieu de le rejeter. Ce fonctionnement est schématisé dans la Figure 1.3. Une composante principale du SDN est son contrôleur; ce dernier constitue son cerveau. Le contrôleur est un élément intelligent capable de configurer les équipements réseaux (les équipements du plan de données) et fournit

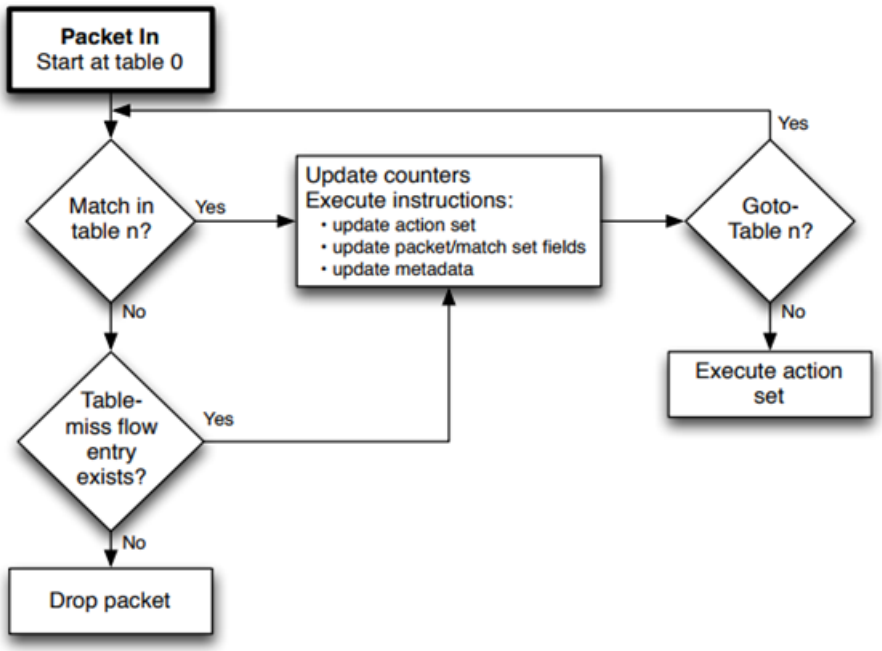


Figure. 1.3. Traitement d'un paquet dans un commutateur OpenFlow

une couche d'abstraction du réseau. Un contrôleur ajoute ou supprime les entrées de flux dans la table de flux.

Le contrôleur SDN permet d'implémenter rapidement un changement sur le réseau à l'aide du protocole OpenFlow. Parmi les contrôleurs existants, on peut trouver: (1) Floodlight [**1**], un contrôleur développé par BigSwitch; (2) OpenDaylight [**2**], un projet open source pour SDN qui a été annoncé publiquement le 8 Avril 2013. C'est un projet de collaboration hébergé par la Fondation Linux. Il contient un contrôleur modulaire et souple; et (3) Ryu [**3**], un exemple d'un contrôleur SDN open source. Il est capable de configurer les équipements réseaux en utilisant OpenFlow et Netconf.

1.2. Vecteurs d'attaques

L'internet est soumis à des fortes variations de trafics, certaines légitimes et d'autres illégitimes. Afin de concevoir une meilleure méthodologie de préventions des attaques, nous devons analyser et prendre en compte ses caractéristiques (les moyens et les méthodes utilisés pour mener et effectuer des attaques). Les principaux vecteurs/types d'attaques sont présentées comme suit: (1) Déni de service (DoS) (voir 1.2.1); (2) Déni de service distribué (DDoS) (voir 1.2.2); (3) attaques par exploitation d'un protocole (voir 1.2.3); (4) attaques logiques (voir 1.2.4); et (5) attaques basées sur la réflexion (voir 1.2.5).

1.2.1. Déni de Service (DoS)

Déni de service (Denial of Service ou DoS) est une attaque réalisée dans le but de rendre indisponible, durant une certaine période, les services ou ressources d'une organisation; ces attaques sont fréquentes, notamment du fait de la relative simplicité de leur mise en œuvre, et de leur efficacité contre une cible non préparée. Ces attaques peuvent engendrer des pertes financières non négligeables par l'interruption de service ou encore indirectement, par l'atteinte portée à l'image de la cible. Le but de ces attaques n'est pas d'altérer ou de supprimer des données mais de nuire au fonctionnement d'un service. La plupart de ces attaques exploitent des failles liées aux protocoles de la pile TCP/IP. Cette thèse se concentre sur les deux vecteurs d'attaques les plus critiques. Parmis ces attaques, nous trouvons DoS. Ce type consiste à submerger des machines légitimes, généralement des serveurs des entreprises, avec un ensemble massif de trafic illégitimes, afin qu'elles ne soient plus accessibles pour répondre aux requêtes des clients légitimes. Le principe est d'envoyer des paquets qui semblent valides, afin de provoquer une saturation ou un état instable des machines victimes. L'attaque les empêche ainsi d'assurer les services réseau qu'elles sont censées offrir. Dans certains cas extrêmes, ce type d'attaque peut conduire au crash des machines cibles; et (2) les attaques par exploitation des vulnérabilités, qui consistent à exploiter une faille d'un système. Le déni de service est un type d'attaque qui peut être coûteux à une organisation puisqu'il interrompt les services disponibles. Sachant qu'à l'heure actuelle, les enjeux des entreprises/organisations sont généralement énormes, cela peut poser de graves problèmes si une telle situation se produit pendant juste quelques secondes.

1.2.2. Déni de service distribué (DDoS)

Le déni de service distribué ou encore "Distributed Denial of Service" est un déni de service qui est provoqué par plusieurs machines. Le but recherché du déni de service distribué est le même que pour DoS, à la différence que plusieurs machines à la fois sont à l'origine de l'attaque, ce qui l'amplifie et rend difficile la localisation des attaquants.

L'attaquant constitue un réseau de machines; ce réseau se compose d'un maître (Master) et de nombreux hôtes distants (esclaves). Pendant le déroulement de l'attaque, l'attaquant se connecte au maître qui envoie un ordre à tous les hôtes distants. Ensuite, ceux-ci vont attaquer la cible suivant une technique choisie par l'attaquant.

Les attaques DDoS se sont démocratisées depuis quelques années. En effet, à leur début, ces attaques nécessitaient de bonnes connaissances de la part des attaquants. Mais aujourd'hui, il existe des outils pour organiser et mettre en place l'attaque. Ainsi le processus de recherche des hôtes secondaires (appelés aussi zombies) a été automatisé. En repérant certaines failles courantes sur les machines présentes sur Internet, l'attaquant prend leur contrôle. Ensuite, Il les utilises pour l'attaque secondaire et essaye également d'effacer ses

traces. Il est essentiel de noter que les victimes dans ce type d'attaques ne sont pas seulement celles qui subissent le déni de service, tous les hôtes secondaires sont également des machines compromises jusqu'au plus haut niveau tout comme l'hôte maître.

Les attaques DDoS peuvent être lancées à partir de réseaux de machines compromises appelés botnets. De nombreux outils accessibles en ligne permettent d'utiliser des botnets. Au cours des dernières années, des services de DDoS en ligne, couramment appelés booter ou stresser (Botnet as-a-service), ont fait leurs apparitions. Ces services proposent des tarifs autorisant leur usage par des particuliers, et permettent à un utilisateur de lancer des attaques contre la cible de son choix. Par ailleurs, certains booter proposent de tester le service gratuitement pendant quelques minutes. La diversité des outils et services permettant de lancer des attaques par déni de service distribué contribue à l'augmentation du nombre de ces attaques.

Ce type d'attaque est très difficile à mitiger. En effet, cette attaque est très dévastatrice car elle provient d'un réseau tout entier. Étant donné que le nombre énorme de machines non sécurisées présentes sur Internet, on peut imaginer l'ampleur d'une telle attaque. Il n'est donc pas évident de s'en protéger étant donné que l'identité des attaquants change souvent, en utilisant les techniques de spoofing. En plus, le temps nécessaire pour mettre en place une protection adéquate est bien souvent supérieure au temps nécessaire pour surcharger la victime.

1.2.3. Attaques par exploitation d'un protocole

Une attaque volumétrique a pour objectif d'épuiser la bande passante réseau disponible afin de rendre un ou plusieurs services inaccessibles. Ce type d'attaque est souvent mené en exploitant les propriétés de certains protocoles (par exemple: DNS [**4**] et NTP [**5**]) afin de maximiser le volume de trafic généré.

Par ailleurs, les attaques volumétriques visent à générer un très grand nombre de paquets afin de saturer les ressources d'une cible (victime). Certains protocoles génèrent des réponses d'une taille très supérieure à celle de la requête (amplification). Le nombre de paquets induits par la réponse peut également être plus important que le nombre de paquets nécessaire à l'envoi de la requête.

La plupart du temps, les attaques volumétriques tirent parti de la réflexion et de l'amplification. Il existe un certain nombre de protocoles pouvant être exploités pour mener ce type d'attaques. Parmi ceux-ci, on peut notamment citer, DNS, NTP (Network Time Protocol), SNMP (Simple Network Management Protocol), ou encore CHARGEN (Character Generator protocole). L'attaque d'amplification DNS inonde la victime en envoyant un ensemble massif de réponses DNS de grandes tailles. Cette attaque tire parti des serveurs DNS récursifs qui acceptent et répondent aux requêtes de résolutions de n'importe qui sur Internet sans

vérifier son identité. Les attaquants exploitent ces serveurs de noms récursifs pour amplifier l'attaque en utilisant l'usurpation de l'adresse IP de la victime. L'effet d'amplification dans cette attaque vient du fait que les petites requêtes peuvent générer des quantités massives de paquets UDP en réponse.

1.2.4. Attaques logiques

Dans les attaques logiques ou logicielles, un petit nombre de paquets malformés exploitent des bogues logiciels spécifiques connus dans le système d'exploitation ou dans une application du système cible. Cela peut potentiellement désactiver la machine de la victime en lui envoyant simplement un ou quelques paquets. Trois types de ces vecteurs d'attaques sont les plus critiques: Remote to Local (R2L), Probe, et User to Root (U2R). R2L est un type d'attaque qui tente d'obtenir un accès local à des objets distants. Probe est un type d'attaque qui essaye d'obtenir des informations sensibles d'un réseau, informations personnelles sur les clients et les informations bancaires. U2R est un type d'attaque où l'attaquant tente d'accéder au système/réseau en tant que super-utilisateur (root); l'attaquant tente d'exploiter les vulnérabilités d'un système pour obtenir les privilèges d'un super-utilisateur (root).

1.2.5. Attaques basées sur la réflexion

Certaines attaques utilisent des machines accessibles sur Internet et répondant à des requêtes émanant d'une source quelconque, il s'agit de réflecteurs. Une attaque par réflexion consiste à envoyer des paquets à des réflecteurs en utilisant l'adresse IP de la victime comme adresse IP source; on parle ainsi de l'usurpation d'identité. Les réponses de ces réflecteurs à la victime induisent la génération d'un trafic non sollicité à destination de cette dernière. Ce trafic peut être suffisamment important pour saturer les liens réseau de la victime, entraînant un déni de service distribué. Les attaques par réflexion impliquent souvent des protocoles reposant sur le protocole de transport UDP. En effet, le protocole UDP laisse à la couche application la tâche d'identification de la source, ce qui permet l'usurpation d'adresse IP. Par ailleurs, UDP ne nécessite pas l'établissement d'une session (contrairement au protocole TCP) préalablement à l'envoi de données. Cette particularité permet à un attaquant d'envoyer une requête à un service utilisant UDP au moyen d'un seul paquet, et de générer une réponse de la part du réflecteur. Les attaques par réflexion ne sont pas uniquement limitées au protocole de transport UDP. Par exemple, il est possible d'envoyer des paquets TCP SYN en usurpant l'adresse IP d'une victime pour générer des paquets SYN-ACK en réponse vers la cible.

1.3. Classification des mécanismes de défenses

Les mécanismes de défense contre les différent types d'attaques sont devenus l'un des défis les plus importants de la sécurité du réseau. Par conséquent, un grand nombre de classifications de défense et de taxonomies ont émergé. Nous présenterons les quatre grandes catégories et politiques. Le but de cette catégorisation est de mettre en évidence les principales caractéristiques de chaque catégorie de défense. (1) prévention de l'attaque: elle vise à arrêter les attaques avant qu'elles puissent atteindre la cible. Le pare-feu est un exemple de ce mécanisme de prévention; (2) la détection d'attaque: elle vise à détecter les anomalies en analysant le trafic entrant dans le réseau; (3) identification de l'origine de l'attaque: elle vise à localiser la source de l'attaque. Cependant, l'identification est difficile; Ceci est dû à deux aspects du protocole IP. D'une part, il est facile d'usurper les adresses IP sources. D'autre part, on ne peut pas connaitre le chemin complet de bout en bout d'un paquet. SDN peut se présenter comme une solution à ces lacunes grâce à sa vue globale du réseau. SDN permet d'organiser un ensemble de commutateurs OpenFlow à travers un contrôleur unique, permettant ainsi la centralisation du plan de contrôle du réseau. Grâce à cette centralisation, il est plus facile de tracer le chemin complet de bout en bout puisque le contrôleur a une vision globale sur le réseau.

1.3.1. Prévention de l'attaque

L'objectif est d'arrêter les attaques avant qu'elles ne causent réellement des dégâts. Ce type essaie de supprimer le trafic illégitime qui peut être reconnu comme malveillant. Le meilleur endroit pour déployer ces mécanismes est dans les routeurs et les hôtes; ceci implique de réparation toutes les vulnérabilités de tous les hôtes qui peuvent être mal utilisés pour une attaque. Certaines approches utiles pour prévenir les attaques DDoS sont: (1) filtrage: ce mécanisme vise à mettre en œuvre des politiques de filtrages du trafic d'entré et de sortie sur tous les routeurs afin de protéger la cible contre les attaques arrivant de l'extérieur et empêcher le réseau lui-même d'être un intrus potentiel et empêcher un attaquant intérieur de mener des attaques vers l'extérieur; (2) pare-feu: avant qu'une attaque ne soit effectuée, un pare-feu pourrait être utile pour filtrer le trafic en fonction du protocole, des ports ou des adresses IP entrantes. Mais, le problème est que les pares-feux ne peuvent pas distinguer entre une attaque et un trafic légitime, et le blocage de tout le trafic pour un port ou protocole spécifique n'est pas une solution mais une génération d'attaque sur lui-même. Seulement dans les attaques dans lesquelles les modèles de signature sont connus, qu'on peut se protéger contre. Cependant, les attaques avec des nouvelles signatures peuvent empêcher l'attaque de ne pas être détecté; et (3) étudier les mécanismes d'attaques et les faiblesses de la conception des protocoles peut s'avérer très utiles pour protéger le réseau telle que DNS amplification attaque.

1.3.2. Détection d'attaque

Une fois le réseau est attaqué, un mécanisme de détection d'attaque doit effectivement mitiger les attaques. En outre, dans une situation d'attaque, le trafic légitime doit passer sans être bien bloqué. De plus, un mécanisme efficace de détection d'attaque doit maintenir l'équilibre entre les faux positifs et les faux négatifs.

1.3.3. Identification de l'origine de l'attaque

Une fois qu'une attaque est détectée, la meilleure solution est de bloquer le trafic d'attaque à sa source, dont l'objectif de localiser les sources d'attaques indépendamment du champ de l'adresse source du paquet qui peut être forgée. Cependant, il est difficile de localiser la vraie source des attaques. Ceci est dû à deux caractéristiques du protocole IP. La première est la facilité avec laquelle les adresses source IP peuvent être forgées. La seconde est la nature du routage IP qui est sans état, où les routeurs ne connaissent normalement que la prochaine étape pour transmettre un paquet, et non le chemin complet de bout en bout pris par chaque paquet.

1.4. Motivations et objectifs

SDN propose une architecture qui découple les fonctions de contrôle et les fonctions de transfert de données du réseau, au contraire de l'architecture de réseau actuelle où les périphériques réseau sont couplés avec un plan de contrôle spécialisé. Cela permet d'une part, de mieux reprogrammer et d'automatiser le réseau et d'offrir de nouvelles capacités pour mitiger les attaques par déni de service distribué. D'autre part, cette séparation introduit de nouveaux défis en matière de sécurité du plan de contrôle.

L'enjeu de cette thèse est double. D'une part, étudier et explorer l'apport du SDN à la sécurité afin de concevoir des solutions efficaces qui vont détecter et mitiger les attaques par déni de service distribué. La thèse explore cette direction de recherche en appliquant les concepts des réseaux programmables permettant d'avoir une vision globale du réseau, des approches dites proactives permettant d'anticiper ces attaques; des algorithmes d'apprentissage automatique (Machine Learning (ML)), des algorithmes d'apprentissage d'ensemble (Ensemble Learning (EL)), des algorithmes d'apprentissage profond (Deep Learning (DL)), et des algorithmes d'apprentissage fédéré (Federate Learning) permettant de détecter efficacement ces attaques; et des mécanismes de prévention permettant de mitiger efficacement ces attaques. En plus de cette mitigation intra-domaine, la thèse propose un système de mitigation inter-domaine permettant à plusieurs domaines SDN de collaborer en toute sécurité et de transférer les informations d'attaque de manière fiable, efficace et sécurisée.

D'autre part, la séparation du plan de contrôle du plan de donnés introduit de nouveaux défis en matière de sécurité du plan de contrôle. La thèse explore cette direction de recherche en proposant des nouvelles méthodes de collectes et de gestion de données permettant de protéger les ressources du plan de contrôle. La thèse conçoit et implémente ces solutions sur une infrastructure SDN; nous utilisons la programmabilité apportée par SDN pour renforcer les politiques de sécurité. Nous proposons des mécanismes pour évaluer ces aspects de sécurité, de performance et de mitigation aux attaques par déni de service distribué et aussi pour déployer ces politiques dans un contexte de réseaux programmables. Dans cette thèse, nous allons aborder quatres problématiques:

Le premier problème est l'identification, la quantification des vulnérabilités du protocole DNS et la mitigation des attaques d'amplification en utilisant ce protocole (*i.e.*, DNS amplification attaque); cette attaque est considérée comme la plus grande attaque DDoS de l'histoire, dépassant 1 Tbit/s [**6**]. Les attaques d'amplification DNS sont parmi les attaques DDoS les plus complexes et répandues aujourd'hui. En vogue, ces attaques deviennent de plus en plus puissantes et dévastatrices et causent des dégâts catastrophiques pour les organismes, les instituts, les opérateurs de réseaux, et les fournisseurs d'accès Internet. DNS est vulnérable aux attaques de déni de service distribuées parce que la majorité des échanges dans ce protocole sont basés sur le protocole UDP. Ces attaques sont difficiles à vaincre car les attaquants usurpent l'adresse IP de la victime (cible d'une attaque par amplification DNS) et l'inondent avec des réponses DNS valables provenant de serveurs DNS légitimes. Dans cette attaque, l'attaquant émet une requête DNS avec une adresse IP forgée (l'adresse IP de la victime) vers plusieurs résolveurs DNS ouverts (serveurs DNS accessibles au public); ces derniers se chargent de faire la résolution récursive et de répondre à ces requêtes par des réponses amplifiées envoyées à la victime. Avec de nombreuses fausses requêtes DNS envoyées par de nombreuses machines contrôlées par l'attaquant (botnet), et plusieurs résolveurs DNS qui répondent simultanément, le réseau de la victime se trouve submergé par le nombre des réponses DNS. Les solutions existantes dans la littérature proposent de resserrer la sécurité des serveurs DNS en bloquant les serveurs DNS de relais récursifs. Ils existent actuellement plus de 7.5 millions de serveurs de relais ouverts sur Internet [**7**]. Il est très difficile voire impossible de les identifier et de tous les bloquer. En outre, si ces serveurs sont bloqués, ils ne seront plus utilisés pour faire la résolution des requêtes DNS légitimes. En conclusion, ces propositions sont impossibles à mettre en place dans un cas réel. Récemment, SDN a émergé comme un paradigme novateur qui apporte de nombreux avantages en découplant le plan de contrôle du plan de données; la gestion centralisée du SDN permet de développer de nouveaux mécanismes de sécurité afin de contrer les attaques d'amplification par DNS. Plusieurs solutions dans la littérature, en utilisant SDN, ont été proposées pour contrer ces attaques. Ces solutions proposent d'utiliser le plan de contrôles (le contrôleur SDN) pour

analyser le trafic DNS. Ceci est fait comme suit: (1) le contrôleur SDN ordonne les équipements du plan de données de lui transférer le trafic DNS pendant un certain intervalle de temps; puis (2) le contrôleur SDN analyse et alerte s'il y une attaque d'amplification DNS. Ces solutions souffrent de plusieurs faiblesses. Tout d'abord, ce mécanisme réduit les performances du contrôleur puisqu'il reçoit tout le trafic suspecté d'être illicite et l'analyse; cette surcharge peut induire un DDoS sur le plan de contrôle. De ce fait, l'attaque n'est pas mitigée mais plutôt convertie en vecteur d'attaque contaminant plusieurs parties du réseau (plan de contrôle). En plus, ces solutions mettent l'hypothèse sur un seul cas d'attaque (attaquant externe ou bien attaquant interne/ serveur DNS externe ou bien interne). Cependant, il n'y aucune solution dans la littérature qui permet de résoudre la problématique en considérant tous les cas possibles d'attaques et en protégeant les ressources du plan de contrôle. Alors, cette thèse a comme première contribution de détecter et prévenir les attaques d'amplification DNS en reposant sur la puissance du SDN et son apport sur le contrôle global du réseau tout en protégeant les ressources du plan de contrôle et en considérant tous les cas possibles d'attaques.

Le deuxième problème est lié à l'identification et la quantification des caractéristiques communes des attaques de déni de service distribués (DDoS); cela permet l'amélioration de notre mécanisme de mitigation intra-domain afin de considérer n'importe quel type d'attaque DDoS et pas seulement l'attaque d'amplification DNS. Notre objective est de généraliser notre mécanisme pour qu'il détecte n'importe quel type d'attaque par déni de service distribué qui essaye d'inonder le réseau de la victime avec un ensemble massif de trafic illégitime. En plus de cette mitigation intra-domaine, nous abordons un mécanisme de mitigation inter-domaine permettant à plusieurs domaines SDN de collaborer en toute sécurité et de transférer les informations d'attaque de manière fiable, efficace et sécurisée. Les différentes solutions existantes dans la littérature sont confrontées à des obstacles en raison de leur faible flexibilité, de leur manque de ressources et de leur coût élevé; ces solutions de collaboration sont centralisées. Les solutions centralisées, par nature, constituent un goulot d'étranglement. Les nouvelles technologies émergentes, telles que la blockchain et les contrats intelligents, offrent de nouvelles opportunités permettant à plusieurs domaines de collaborer en toute sécurité et de transférer les informations d'attaque de manière flexible, fiable, efficace et décentralisée. Aucune solution dans la littérature n'a proposé de mitiger les attaques par déni de service distribué sur deux niveaux (intra-domaine et inter-domaine) en combinant SDN et la blockchain. L'utilisation du SDN et la blockchain permet d'assurer la sécurité, la flexibilité et l'efficacité d'une collaboration inter-domaine de manière décentralisée et automatisée. Les principaux avantages de l'utilisation de ces technologies sont: (1) le déploiement d'une infrastructure décentralisée déjà existante (Ethereum [**8**]) afin de collaborer et mitiger l'attaque. En effet, avec l'utilisation d'Ethereum, on peut éviter la complexité du développement de nouveaux protocoles et/ou la modification de protocoles

existants. DOTS est l'une des solutions traditionnelles qui ont été proposées par l'IETF (Internet Engineering Task Force); cependant, il n'a pas été envisagé pour un déploiement à grande échelle en raison du coût élevé de son déploiement ainsi que la complexité de sa mise en œuvre; (2) blockchain élimine la nécessité d'un tiers de confiance centralisé qui permet de maintenir la collaboration entre les domaines; les problèmes liés à l'utilisation de ce dernier sont nombreux (fiabilité et disponibilité des données de collaboration et les coûts élevés afin de maintenir et sécuriser ces données); ceci est assuré via les contrats intelligents; et (3) dans les systèmes traditionnels de communication inter-domaine, une infrastructure à clé publique (PKI) est nécessaire afin d'assurer l'intégrité des données partagées. Les systèmes basés sur PKI sont coûteux à installer et à maintenir. Blockchain permet une communication inter-domaine sécurisée tout en maintenant le système transparent, à moindre coût et flexible.

Le troisième problème est lié à l'identification, la quantification et la mitigation d'autres vecteurs d'attaques afin de permettre à notre mécanisme intra-domaine élaborées dans les 2 premières problématiques de mitiger non seulement les attaques DDoS mais aussi d'autres vecteurs d'attaques. La différence principale entre ces vecteurs d'attaques et les attaques par dénis de service distribuées (DDoS) élaborées dans les 2 premières problématiques est que les attaques par DDoS tentent d'inonder le réseau de la victime avec un trafic amplifié afin d'arrêter complètement son service alors que ces nouveaux vecteurs d'attaques tentent d'infiltrer discrètement le système sans être détectés. Plus précisément, nous nous intéressons dans cette contribution aux attaques qui essayent de: (1) voler et manipuler leurs données sensibles; et (2) prendre le contrôle à distance et les utiliser pour créer des outils d'amplification pour les cyberattaques (botnet as-a-service) pour gagner de l'argent. La croissance rapide du nombre des objets connectés non sécurisés, avec un nombre estimé à 75 milliards d'appareils d'ici la fin de 2025 [**9**], peut améliorer et faciliter la capacité des attaques à grande échelle. Dans ce troisième problème, nous allons extraire les données d'attaques depuis des ensembles de données publiques. Parmis ces vecteurs d'attaques, nous trouvons: Remote to Local (R2L), Probe, et User to Root (U2R). R2L est un type d'attaque qui tente d'obtenir un accès local à des objets distants. Probe, est un type d'attaque qui essaye d'obtenir des informations sensibles d'un réseau (informations personnelles sur les clients et les informations bancaires). U2R est un type d'attaque où l'attaquant tente d'accéder au système/réseau en tant que super-utilisateur (root). L'attaquant tente d'exploiter les vulnérabilités d'un système pour obtenir les privilèges d'un super-utilisateur (root). Nous étudions d'abord les caractéristiques de chacune de ces attaques. Par la suite, nous utilisons des algorithmes d'apprentissage automatique (Machine Learning (ML)) et des algorithmes d'apprentissage d'ensemble (Ensemble Learning (EL)) pour détecter efficacement ces attaques. Nous menons des expérimentations approfondies pour évaluer la solution proposée. Et enfin, nous comparons les résultats obtenus avec ceux de l'état de l'art.

Le quatrième problème est l'extension de notre mitigation inter-domaine élaboré dans la deuxième problématique en considérant des techniques d'apprentissage fédéré. L'absence d'ensembles de données d'entraînement contenant des caractéristiques (patterns) de nouvelles attaques rend les systèmes de détection d'attaques basés sur l'apprentissage automatique et/ou l'apprentissage profond inefficaces. La nature privée de ces ensembles de données et l'émergence de l'apprentissage automatique contradictoire/Réseaux antagonistes génératifs (Adversarial machine learning/Generative adversarial networks [**10**]) rendent difficile le partage les données d'attaques entre les organisations. L'apprentissage fédéré (FL) résout cette limitation en utilisant un entraînement distribué sans que les particiapnts, appelés collaborateurs, ne partagent leurs données. Cela permettra à notre mécanisme inter-domaine d'apprendre d'autres vecteurs d'attaques parmi un grand nombre de collaborateurs tout en préservant la confidentialité de chaque collaborateur. Plus précisément, nous proposons une nouvelle architecture distribuée qui permet à plusieurs domaines SDN de collaborer de manière sécurisée afin de contrer les attaques de type "zero-day" tout en préservant la confidentialité de chaque domaine SDN. Par la suite, nous utilisons un nouveau mécanisme de calcul multipartite sécurisé (Secure multi-party computation(SMPC)) pour agréger de manière sécurisée les mises à jour des modèles de chaque participant. Nous intégrons les contrats intelligents d'Ethereum pour maintenir la collaboration d'une manière totalement décentralisée, fiable, flexible et efficace. Nous menons des expérimentations approfondies pour évaluer la solution proposée en utilisant des données d'attaques réelles. Et enfin, nous comparons les résultats obtenus avec ceux de l'état de l'art.

1.5. Solutions et contributions

Dans cette thèse, nous étudions les attaques les plus critiques et les plus dévastatrices, nous analysons leurs vulnérabilités, et nous proposons des solutions pour mitiger ces attaques dans un environnement SDN. Notre solution est le premier travail qui: (1) permet de résoudre la problématique en considérant tous les cas possibles d'attaques; (2) mitige ces attaques sur deux niveaux (intra-domaine et inter-domaine) en combinant des algorithmes d'apprentissage automatique, des algorithmes d'apprentissage d'ensemble, des algorithmes d'apprentissage profond, des algorithmes d'apprentissage fédéré, SDN, et la blockchain. De plus, nous concevons, implémentons et évaluons notre solution dans un environnement SDN et blockchain afin de prouver sa flexibilité et son efficacité à détecter et mitiger ces attaques ainsi que son rapport coût-efficacité. Nous nous appuyons sur nos travaux de recherche dans cette thèse pour aborder les quatres problèmes mentionnés ci-dessus. Les solutions apportées par notre thèse mènent aux contributions suivantes:

Pour résoudre le problème 1, nous proposons une solution qui bénéficie de l'apport du SDN afin de contrer les attaques d'amplification DNS tout en protégeant les ressources du

plan de contrôle et en considérant tous les cas possibles d'attaques. Nous proposons un mécanisme proactif pour filtrer les requêtes et les réponses illégitimes de l'ensemble du trafic DNS. Cela permet de créer le chemin qui ne laisse passer que les réponses légitimes et sans remonter tout le trafic au contrôleur; ceci permet de se prémunir des problèmes des attaues d'amplification DNS sur le plan de contrôle. Nous proposons un nouveau mécanisme de collecte des statistiques des flux de données afin d'extraire le vecteur représentant l'attaque. Nous proposons un mécanisme de détection en temps réel pour détecter automatiquement cette attaque en utilisant des algorithmes d'apprentissage automatique. Nous avons évalué les performances de notre première contribution en termes d'efficacité et de précision. Les résultats des simulations montrent que notre solution permet d'atténuer efficacement les attaques d'amplification DNS avec une précision élevée et moins de charges sur le plan de contrôle. Dans notre première contribution, nous proposons une mitigation intra-domaine permettant de protéger la victime de tous les cas possibles d'attaques. Cependant, une mitigation intra-domaine seule ne peut pas mitiger les attaques DDoS sur les chemins de la victime parce que les attaquants et la victime ne sont pas nécessairement dans le même système autonome (AS). Ainsi, Il est donc nécessaire de mettre en place une collaboration inter-domaine efficace afin de: (a) réduire efficacement la charge de transfert du trafic d'attaque amplifié, à travers plusieurs domaines; et (b) bloquer l'attaque près de sa source.

Pour résoudre le problème 2, nous étudions d'abord les caractéristiques communes des attaques de déni de service distribuées (DDoS) et retenons celles qui permettent d'avoir un taux de précision plus élevé. Nous concevons un système basé sur la blockchain qui utilise les contrats intelligents pour réaliser une collaboration décentralisée, sécurisée, flexible et à moindre coût, et cela entre plusieurs SDN domaines; il prend en charge deux niveaux de mitigation: intra-domaine et inter-domaine. Nous proposons un mécanisme intra-domaine de détection et de mitigation en temps réel afin de détecter automatiquement les attaques DDoS en se reposant sur des algorithmes d'apprentissage automatique. Nous proposons un mécanisme de collaboration inter-domaine efficace, sécurisé, de faible coût, facile à déployer et décentralisé. Il permet à plusieurs domaines basés sur SDN de collaborer en toute sécurité et de transférer les informations d'attaque d'une manière décentralisée reposant sur la blockchain et les contrats intelligents. Ainsi, en combinant notre approche d'atténuation intra-domaine basée sur SDN et le mécanisme de collaboration inter-domaine, nous sommes non seulement en mesure de mitiger efficacement les attaques DDoS dans le domaine de la victime, mais également de partager les informations d'attaque de manière entièrement décentralisée permettant en une prise de décision efficace, par plusieurs domaines, pour atténuer les attaques DDoS. Nous avons implémenté notre solution dans le réseau de test officiel d'Ethereum. Nous avons évalué les performances de notre deuxième contribution en termes de flexibilité, d'efficacité, de sécurité et de coût d'exécution. Les résultats des simulations

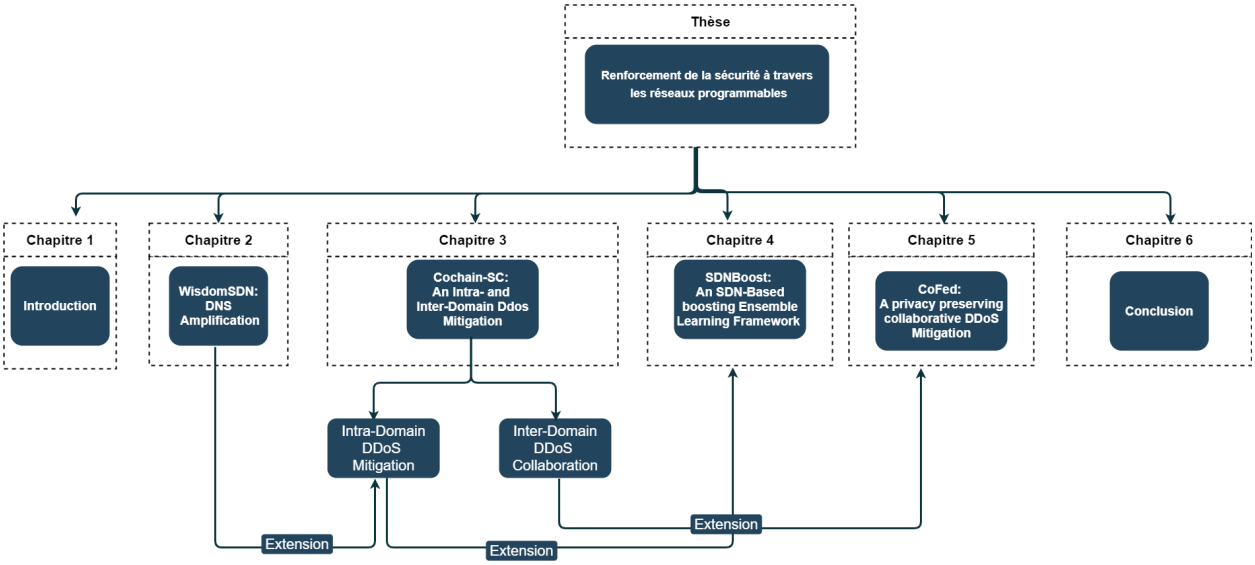


Figure. 1.4. Organisation de la thèse

montrent que notre solution permet d'atténuer efficacement les attaques DDoS avec une précision élevée, un faible taux de faux positifs, et un coût réduit de déploiement.

Pour résoudre le problème 3, nous étendons la mitigation intra-domaine afin de permettre à notre mécanisme intra-domaine de mitiger d'autres vecteurs d'attaques (par exemple: R2L, Probe, U2R, etc.). Nous améliorons les performances de détection en utilisant des algorithmes d'apprentissage d'ensemble (Ensemble Learning (EL)). L'apprentissage d'ensemble (EL) est considéré comme un développement récent dans le domaine d'apprentissage automatique (Machine Learning (ML)) qui vise à combiner plusieurs ML modèles afin de construire des ML modèles plus flexibles (moins de biais) et moins sensibles aux données (moins de variance). Nous intégrons des mécanismes de prévention permettant de mitiger efficacement ces attaques. Nous avons évalué les performances de notre troisième contribution en utilisant des données d'attaques réelles provenant de deux ensembles de données publiques, à savoir le NSL-KDD [**11**] et l'UNSW-NB15 [**12**]. Les résultats des simulations montrent que notre solution permet de mitiger efficacement les attaques avec une précision élevée et un faible taux de faux positifs surpassant les contributions de l'état de l'art avec une plus grande précision de 12%.

Pour résoudre le problème 4, nous étendons notre mitigation inter-domaine en considérant des techniques d'apprentissage fédéré. Nous proposons une nouvelle architecture distribuée qui permet à plusieurs domaines SDN de collaborer de manière sécurisée. Par la suite, nous utilisons un nouveau mécanisme de calcul multipartite sécurisé (Secure multi-party computation(SMPC)) pour agréger de manière sécurisée les mises à jour des modèles de chaque participant. Nous intégrons les contrats intelligents d'Ethereum pour maintenir la

collaboration d'une manière totalement décentralisée, fiable, flexible, et efficace. Nous menons des expériences approfondies pour évaluer la solution proposée en utilisant des données d'attaques réelles, à savoir le NSL-KDD [**11**], et en considérant plusieurs scénarios de test. Les résultats des simulations montrent que notre solution permet de mitiger efficacement les attaques tout en préservant la vie privée des collaborateurs.

1.6. Organisation de la thèse

Cette thèse est organisée comme suit (voir Figure 1.4). Le 2ème chapitre présente notre solution adoptée pour contrer les attaques par amplification DNS: "Bringing Intelligence to Software DefinedNetworks: Mitigating DDoS Attacks". Le 3ème chapitre présente notre mécanisme intra-domaine de détection et de mitigation en temps réel ainsi que notre mécanisme de collaboration inter-domaine: "Cochain-SC: An Intra and Inter-DomainDdos Mitigation Scheme Based on Blockchain Using SDN and Smart Contract". Le 4ème chapitre présente notre contribution améliorons les performances de détection intra-domain en utilisant des algorithmes d'apprentissage d'ensemble: "SDNBoost: An SDN-based BoostingEnsemble Learning Framework for Advanced Network Attack Mitigation". Le 5ème chapitre présente notre contribution améliorons les performances de collaboration inter-domain en utilisant des algorithmes d'apprentissage fédéré: "CoFed: A privacy preserving collaborativeDDoS Mitigation framework based on Federated Learning using SDN and blockchain". Finalement le chapitre 6 conclut la thèse et discute nos prochains travaux.

1.7. Liste des publications

Dans cette section, nous énumérons les articles produits dans le cadre de cette thèse:

1.7.1. Revues internationaux avec comité de lecture

[1] Z. Abou El Houda, A. Senhaji Hafid and L. Khoukhi, **"SDNBoost: An SDN-based Boosting Ensemble Learning Framework for Advanced Network Attack Mitigation,"** IEEE Transactions on Cognitive Communications and Networking, 2021. Submitted.

[2] Z. Abou El Houda, A. Senhaji Hafid and L. Khoukhi, **"CoFed: A privacy preserving collaborative DDoS Mitigation framework based on Federated Learning using SDN and blockchain,"** IEEE Transactions on Network Science and Engineering, 2021. Submitted.

[3] Z. Abou El Houda, L. Khoukhi and A. Senhaji Hafid, **"Bringing Intelligence to Software Defined Networks: Mitigating DDoS Attacks,"** IEEE Transactions on Network and Service Management, vol. 17, no. 4, pp. 2523-2535, Dec. 2020, doi: 10.1109/TNSM.2020.3014870.

[4] Z. Abou El Houda, A. S. Hafid and L. Khoukhi, **"Cochain-SC: An Intra- and Inter-Domain Ddos Mitigation Scheme Based on Blockchain Using SDN and Smart Contract,"** IEEE Access, vol. 7, pp. 98893-98907, 2019, doi: 10.1109/ACCESS.2019.2930715.

1.7.2. Congrès internationaux avec comité de lecture

[1] Z. Abou El Houda, A. Senhaji Hafid and L. Khoukhi, **"A Novel Machine Learning Framework for Advanced Attack Detection using SDN,"** IEEE Global Communications Conference (GLOBECOM), 2021. Accepted.

[2] Z. A. E. Houda, A. Hafid and L. Khoukhi, **"Blockchain-based Reverse Auction for V2V charging in smart grid environment,"** ICC 2021 - 2021 IEEE International Conference on Communications (ICC), Montréal, Canada, 2021.

[3] Z. A. E. Houda, A. Hafid and L. Khoukhi, **"Blockchain Meets AMI: Towards Secure Advanced Metering Infrastructures,"** ICC 2020 - 2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 2020, pp. 1-6.

[4] Z. A. E. Houda, A. Hafid and L. Khoukhi, **"BrainChain - A Machine learning Approach for protecting Blockchain applications using SDN,"** ICC 2020 - 2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 2020, pp. 1-6.

[5] Z. A. El Houda, A. Hafid and L. Khoukhi, **"Co-IoT: A Collaborative DDoS Mitigation Scheme in IoT Environment Based on Blockchain Using SDN,"** 2019 IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 2019, pp. 1-6.

[6] Z. A. El Houda, L. Khoukhi and A. Hafid, **"ChainSecure - A Scalable and Proactive Solution for Protecting Blockchain Applications Using SDN,"** 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 2018, pp. 1-6.

1.7.3. Chapitres de livres

[1] Z. Abou El Houda, A. Senhaji Hafid and L. Khoukhi, **"A novel unsupervised learning method for intrusion detection in SDN,"** Computational Intelligence in Recent Communication Networks, EAI/Springer, 2021. Accepted.

Chapitre 2

Bringing Intelligence to Software Defined Networks: Mitigating DDoS Attacks

2.1. Abstract

As one of the most devastating types of Distributed Denial of Service (DDoS) attacks, Domain Name System (DNS) amplification attack represents a big threat and one of the main Internet security problems to nowadays networks. Many protocols that form the Internet infrastructure expose a set of vulnerabilities that can be exploited by attackers to carry out a set of attacks. DNS, one of the most critical elements of the Internet, is among these protocols. It is vulnerable to DDoS attacks mainly because most of the exchanges in this protocol use User Datagram Protocol (UDP). These attacks are difficult to defeat because attackers spoof the IP address of the victim and flood him with valid DNS responses coming from legitimate DNS servers. In this chapter, we propose an efficient and scalable solution, called WisdomSDN, to effectively mitigate DNS amplification attack in the context of software defined networks (SDN). WisdomSDN covers both detection and mitigation of illegitimate DNS requests and responses. WisdomSDN consists of: (1) a novel proactive and stateful scheme (PAS) to perform one-to-one mapping between DNS requests and DNS responses; it operates proactively by sending only legitimate responses, excluding amplified illegitimate DNS responses; (2) a machine learning DDoS detection module to detect, in real-time, illegitimate DNS requests. This module consists of: (a) Flow statistics collection scheme (FSC) to gather the features of flows in an efficient and scalable way using sFlow protocol; (b) Entropy calculation scheme (ECS) to measure randomness of network traffic; and (c) Bayes Network based Filtering scheme (BNF) to classify, based on entropy values, illegitimate DNS requests; and (3) DNS Mitigation scheme (DM) to effectively mitigate illegitimate DNS requests. The experimental results show that, compared to state-of-the-art, WisdomSDN can effectively detect/mitigate DNS amplification attack quickly with a high detection rate, less false positive rates, and a low overhead making it a

promising solution to mitigate DNS amplification attack in a SDN environment.

Keywords: DDoS; SDN; Entropy; Bayes Classifier.

Status: This journal paper is published. Z. Abou El Houda, L. Khoukhi and A. Senhaji Hafid, **"Bringing Intelligence to Software Defined Networks: Mitigating DDoS Attacks,"** IEEE Transactions on Network and Service Management, vol. 17, no. 4, pp. 2523-2535, Dec. 2020, doi: 10.1109/TNSM.2020.3014870.

2.2. Introduction

DNS amplification attack is a popular form of DDoS attacks that relies on the use of Open Resolver (publically accessible DNS servers) to overwhelm the victim (*i.e.*, target of DNS amplification attack) with amplified DNS traffic. This attack is based on a recursive function of DNS servers. Usually, the DNS server accepts and responds to resolution requests from anyone without verifying its identity. Thus, attackers can exploit recursive functions to amplify the attack by spoofing the victim's IP address. The spoofed queries (*i.e.*, DNS requests) sent by the attacker are of type *ANY*; they include all known information about a DNS zone in a single request. The amplification impact of this attack comes from the fact that small queries can generate massive amounts of UDP packets in response. This category of attack can be divided in two types: (a) amplification with repeated DNS requests that have the same content; and (b) amplification with varied DNS requests that have different contents. The query can be of type *ANY* that requests all records for a particular domain or different domains. The size of the response may be large to produce a high level of amplification. The amplification ratio, of a factor up to 4670 [**13**], is calculated as the ratio between the response size and the request size. According to a recent study, there are about 7.5 million external DNS servers in the Internet; more than 75% of these servers allow recursive name service to the public [**7**]. This can cause significant collateral damage on the victim, if attackers use many recursive servers to amplify and generate the attack. As example, on the 2 of October 2016, a huge attack was conducted against the servers of Dyn, a company that controls many Internets DNS servers. As a consequence, many popular Internet services, *e.g.*, Amazon, Twitter, GitHub [**14**], PayPal and others became unavailable for several hours [**6**]. This attack [**6**] is considered as the largest ever DDoS attack, exceeding a rate of 1 Tbit/s. Such incidents harm Internet service providers (ISPs) and cost millions of dollars of lost revenues for enterprises. DNS amplification attack consists of: (1) an attacker (*e.g.*, bot master); (2) a large number of compromised devices (called zombies); and (3) reflectors (*i.e.*, Open Resolvers). Each zombie is ordered by the bot master to send a large number of DNS requests, in which the source IP address is replaced with

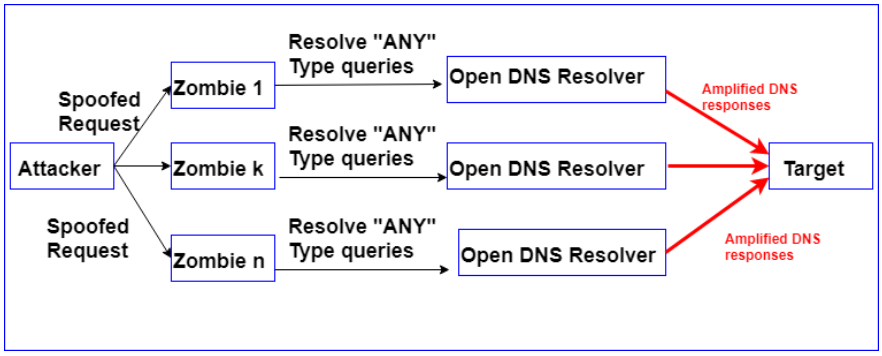


Figure. 2.1. DNS amplification attack.

the victim's IP address (*i.e.*, spoofed), to Open Resolvers. Upon receipt of these illegitimate DNS requests, Open Resolvers make a recursive resolution and flood the victim with large number of amplified DNS responses (see Fig. 2.1).

In this chapter, we propose an efficient, stateful, proactive and scalable solution in the context of SDN, called WisdomSDN. Recently, SDN has attracted tremendous attention from industry and academia as an emerging networking paradigm that facilitates network management and provides new approaches to manage and deploy networks dynamically [**15**–**18**]. SDN separates data and control planes; this separation allows for more control over the network and brings a new way to deal with various forms of DDoS attacks [**19**–**23**]. While SDN can protect the network from DDoS attacks [**24**–**29**], it can be a victim of these attacks [**30**]. To address this problem, WisdomSDN promises to leverage the advantages of SDN to protect the victim from DNS amplification attack while maintaining the SDN secure *i.e.*, protecting the resources of data plane (*i.e.*, Ternary Content Addressable Memory (TCAM) of OF (OpenFlow) switches), the resources of control plane (*i.e.*, the SDN controller resources) and the workload of OF channel. To this aim, WisdomSDN makes use of a novel proactive and stateful mapping scheme (PAS), based on in-OF switch (*e.g.*, OpenvSwitch [**31**]) processing capabilities, to mitigate DNS amplification attack and protect SDN controller. PAS adopts a proactive and stateful paradigm to perform one-to-one mapping between DNS requests and DNS responses; this can effectively offload the SDN controller resources and the OF channel. Each OF switch filters DNS traffic according to the header fields (*i.e.*, MAC address, IP address and UDP port). PAS checks the legitimacy of each DNS response by comparing its MAC address, IP address and UDP Port with the corresponding DNS request and drops automatically illegitimate DNS responses. Thus, PAS allows: (1) OF switches to be smart enough to react proactively and quickly to mitigate illegitimate DNS responses, and not wait for a reactive rule from the SDN controller; and (2) to effectively offload the control plane (*i.e.*, SDN controller). However, if each OF switch maintains the one-to-one mapping of all DNS requests that it receives it will be overwhelmed. To address this issue and protect TCAM of OF switches that is limited in size, WisdomSDN makes use of a robust machine

learning DDoS detection module that aims to detect, in real-time, illegitimate DNS requests. This module consists of: (1) FSC, to gather the features of flows in an efficient and scalable way using sFlow protocol; (2) ECS, to measure the disorder/randomness of network traffic; and (3) BNF, to automatically detect illegitimate DNS requests. To evaluate the effectiveness of WisdomSDN, we conducted a set of experiments in Mininet [**32**]. We launch the attack on a simulated SDN network environment in the context of 2 scenarios: (1) without WisdomSDN; and (2) with WisdomSDN. The results show that without WisdomSDN, the SDN controller and victim are flooded with illegitimate DNS traffic (DNS requests and DNS responses). However, with WisdomSDN, OF switches can effectively detect and mitigate illegitimate DNS requests and DNS responses. Also, we evaluated BNF through Receiver Operating Characteristic (ROC) curves and we compared it to the most prominent state-of-art schemes. The results show that BNF can effectively detect the attack with a high detection rate and low false positive rates. To the best of our knowledge, WisdomSDN is the first contribution that mitigates this attack considering all possible attacks setup while maintaining the SDN secure.

The main contributions of this chapter can be summarized as follows:

- We propose a novel proactive and stateful scheme (PAS) to perform one-to-one mapping between DNS requests and DNS responses.
- We propose a flow statistics collection scheme (FSC) to gather the features of flows in an efficient way using sFlow protocol.
- We introduce an entropy calculation scheme (ECS) to measure the disorder/randomness of network traffic.
- We propose a Network based Filtering scheme (BNF) to classify, based on entropy values, illegitimate DNS requests.
- We propose a DNS Mitigation scheme (DM) to effectively mitigate illegitimate DNS requests.
- We evaluate the performance of WisdomSDN in terms of scalability, effectiveness and efficiency. The experiments results show that WisdomSDN can effectively mitigate DNS amplification attack with a high detection rate, low false positive rates and a minor overhead.

The rest of this chapter is organized as follows. Section 2.3 presents related works. Section 2.4 presents an overview of WisdomSDN. Section 2.5 introduces our system design. Section 2.6 presents PAS, our proactive and stateful scheme. Section 2.7 presents our machine learning DDoS detection module that consists of: FSC, ECS, and BNF. Section 2.8 describes DM. Section 2.9 evaluates WisdomSDN in terms of efficiency, scalability, detection rate and presents the simulation results. Finally, section 2.10 concludes the chapter and presents our future works.

2.3. Related Work

Several schemes have been proposed in the literature to mitigate DNS amplification attack. In the following: (1) we classify countermeasures into two groups; and (2) for each group, we present some of the most prominent works as well as their limitations.

2.3.1. Countermeasures in Legacy Networks

In [**33**], Huistra proposed a scheme to detect malicious DNS traffic by detecting all IP addresses that cause the attack, based on collected dataset from NetFlow. The scheme consists of two phases: (1) detection of suspicious IP addresses based on the quantity of requests generated by this IP address and stored in the flow-record; and (2) detection of any IP address that receives suspicious DNS responses based on the huge amount of received responses. However, the execution of the two phases, in this scheme [**33**], result in large response times. In [**34**], Rozekrans et al. proposed a defense mechanism, called response rate limiting (RRL), to limit the amount of generated responses by dropping the ones that exceed a predefined threshold. This is performed by storing the requestor IP address when DNS server generates a response for a DNS request. When the number of responses exceeds the threshold, the server drops requests for this IP address. However, RRL only examines DNS responses and ignores the amount of incoming DNS requests. In [**35**], Sun et al. proposed a low-cost hardware based scheme to mitigate DNS amplification attack. The solution works well except that it is hardware-based, which makes it hard to update and extend. In [**36**], Guo et al. proposed a scheme that deploys filters at the border of networks in order to block incoming source IP addresses that are not belong to their networks. However, this scheme has "neighborhood policy" that requires all ISPs to participate in order to provide the complete list of IP addresses that do not sent from the network; moreover, the effectiveness of this scheme depends on its global deployment across the Internet. In [**37**], Kambourakis et al. proposed a scheme that stores all incoming DNS requests and DNS responses. Once an illegitimate DNS response is detected, a counter is incremented until it reaches a threshold. When the threshold is reached, an alert is generated and the attack is assumed to have happened. This scheme did not scale for large scale networks because it needs to store all DNS requests and responses.

2.3.2. SDN-Based Countermeasures

Our previous works [**38**, **39**] did show their effectiveness in protecting permissioned blockchain from DNS amplification attack. In this chapter, we extend these works to protect any type of applications and not consider only blockchain applications. Moreover, we combine a proactive and stateful scheme with a machine learning algorithm (*i.e.*, BNF) in order

to: (1) distinguish between legitimate and illegitimate responses and systematically eliminate the amplified illegitimate DNS responses; and (2) decrease false positive rates while maintaining a high detection rate. In [**40**], Rodrigo et al. proposed a flow-based intrusion detection scheme using OF protocol to gather network traffic features. This scheme [**40**] focuses only on the attack in data plane without any analysis of the overhead to the control plane. Moreover, the performance analysis did not include the overall system performance. In [**41**], Mehdi et al. proposed an anomaly detection scheme in the context of SDN using OF protocol. However, this scheme [**41**] was designed only for small-scale setup; in large-scale environment, a high rate traffic from data plane to control plane may overload the SDN controller. In [**42**], Wang et al. proposed an entropy scheme based on OF switches; it focuses only on detection, but it cannot find the victim or the illegitimate hosts. In [**43**], Lim et al. proposed a DDoS attacks mitigation scheme for botnet-based attacks that runs on SDN controller. This scheme [**43**] requires a large amount of communications between the SDN controller and OF switches in order to protect the victim; moreover, it not only generates DDoS attacks against the SDN controller but also requires high latency to cooperate with the SDN controller. In [**44**], Zaalouk et al. proposed a scheme based on SDN to mitigate DNS amplification attack; it uses sFlow protocol to monitor DNS traffic. When the attack is detected, the orchestrator commands one of the SDN controllers to forward suspicious traffic to it in order to analyze the size of DNS response packets and to compute their average size. If the average of responses size exceeds the value of a threshold, it proceeds to the second phase of detection. In this phase, the orchestrator calculates the entropy of destination IP address. If the entropy value is low, then it is assumed that there is a DNS amplification attack. In this case, it applies a set of rules to limit the responses rate. However, this scheme [**44**] did not distinguish between legitimate and illegitimate responses since all DNS responses are sent to SDN controller and may cause DDoS attack against control plane as well as orchestrator. In [**45**], K. Giotis et al. proposed the use of sFlow protocol with OF protocol in order to detect DDoS attacks reducing the communication overhead between data plane and control plane. This scheme [**45**] works well; however, it has high false positive rates.

To address the weaknesses of these existing schemes [**33**–**45**], we propose WisdomSDN, an efficient, stateful, proactive, and scalable solution to detect and mitigate DNS amplification attack. In WisdomSDN, we use: (1) PAS, to exclude the amplified DNS responses; (2) a machine learning DDoS detection module, to detect, in real-time, illegitimate DNS requests; and (3) DM, to effectively mitigate illegitimate DNS requests. WisdomSDN uses FSC to separate flow monitoring from the forwarding logic; this makes it much more scalable compared to existing native OF schemes [**40**–**43**]. Using BNF, WisdomSDN is much accurate in comparison with the ones using sampling technology [**38**, **44**, **45**].

2.4. WisdomSDN: An Overview

In this section, we present an overview of WisdomSDN. More specifically, we explain how WisdomSDN can combine detection and mitigation of DNS amplification attack, allowing for a robust detection and an effective mitigation of this attack.

The majority of DNS requests use UDP as a transport protocol without providing any mechanism to verify the source IP address of DNS requests. Therefore, the network can be flooded with illegitimate DNS requests and amplified DNS responses. WisdomSDN is inspired from the techniques of Moving Target Defense [**46**]. These techniques allow to contain the attack in the space of the real source of the request and avoid propagating DDoS attacks to the potential victim.

Fig. 2.2 shows the flow diagram of WisdomSDN. When an OF switch receives a new packet, it first checks the type of incoming packet. If the packet is DNS request; then, WisdomSDN checks whether the maximum number of requests threshold per ingress port is reached. If the response is yes, it triggers, using sFlow protocol, an event to sFlow collector (*i.e.*, sFlow-RT [**47**]) in order to collect network traffic features using FSC. Afterwards, ECS extracts network traffic features from collected information and calculates entropy values. Based on this calculation, BNF detects automatically illegitimate flows. If the flow is classified as illegitimate, then a mitigation action is performed using DM scheme. Otherwise, OF switch: (1) learns the authorized response from ingress port of incoming DNS request; and (2) installs the rule that allows only the DNS response that matches the corresponding DNS request with a short timeout (*i.e.*, idle timeout and hard timeout) in order to avoid the storage complexity to maintain the one-to-one mapping. Thus, any amplified illegitimate DNS response will not be sent to neither the victim nor the SDN controller; this excludes amplified illegitimate DNS responses and offload the SDN controller and the OF channel. Upon receipt of DNS response, OF switch checks whether there is any corresponding DNS request; if it is the case, it forwards the DNS response following the ingress port of incoming DNS request; otherwise, it drops the DNS response.

2.5. System Design

2.5.1. Design Overview

When designing WisdomSDN, we did consider the following objectives. First, WisdomSDN should give a full protection from DNS amplification attack. Unlike existing schemes [**33**–**45**] that try to analyze the network state; then, detect the attack. WisdomSDN aims to act proactively, by maintaining one-to-one mapping between DNS requests and DNS responses, in order to avoid sending illegitimate DNS traffic to victim (*i.e.*, target of the attack); this is ensured via PAS. To protect TCAM of OF switches which is limited in

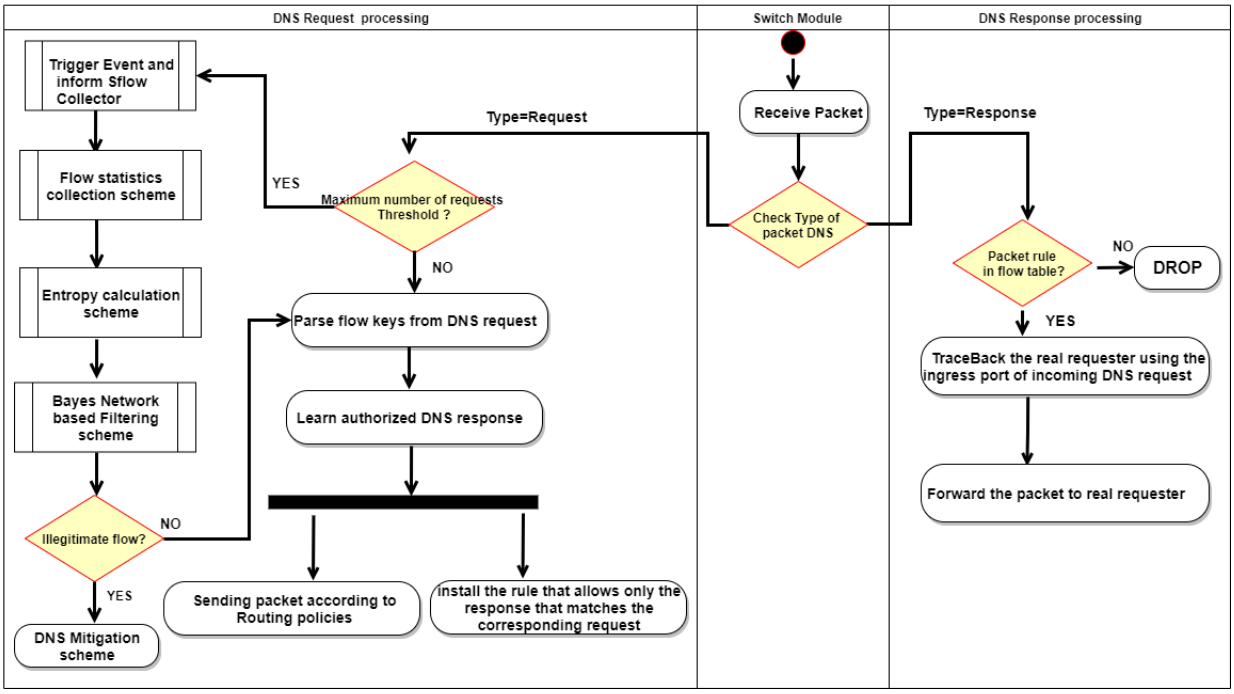


Figure. 2.2. Flow diagram of WisdomSDN.

size, WisdomSDN makes use of a robust machine learning DDoS detection module that aims to detect, in real-time, illegitimate DNS requests. Finally, the attack should be effectively mitigated, using DM, and the whole system has to be as scalable as possible.

2.5.2. System Architecture

Fig. 2.3 shows the architecture of WisdomSDN. WisdomSDN has three phases: (1) PAS, a novel proactive and stateful scheme to perform one-to-one mapping between DNS requests and DNS responses; it operates proactively by sending only legitimate responses, excluding amplified illegitimate DNS responses. PAS is implemented in data plane (*i.e.*, OF switch); (2) a machine learning DDoS detection module that aims to detect, in real-time, illegitimate DNS requests. This module is implemented on the top of the SDN controller (*i.e.*, application plane) and consists of: (a) FSC, a novel Flow statistics collection scheme to monitor, using sFlow protocol, network traffic features in an efficient and scalable way; it defines the monitoring metrics (*e.g.*, the attributes of flows aggregation and thresholds); (b) ECS, an entropy calculation scheme to measure disorder/randomness of network traffic; and (c) BNF, a real-time detection scheme, to classify, based on entropy values, illegitimate DNS requests; (3) DM, a DNS mitigation scheme to effectively mitigate illegitimate DNS requests enabling the network to recover quickly in short time. DM installs new OF rules into OF switches under attack in order to monitor the speed of illegitimate DNS requests. WisdomSDN separates flow monitoring from the forwarding logic; this makes it much scalable

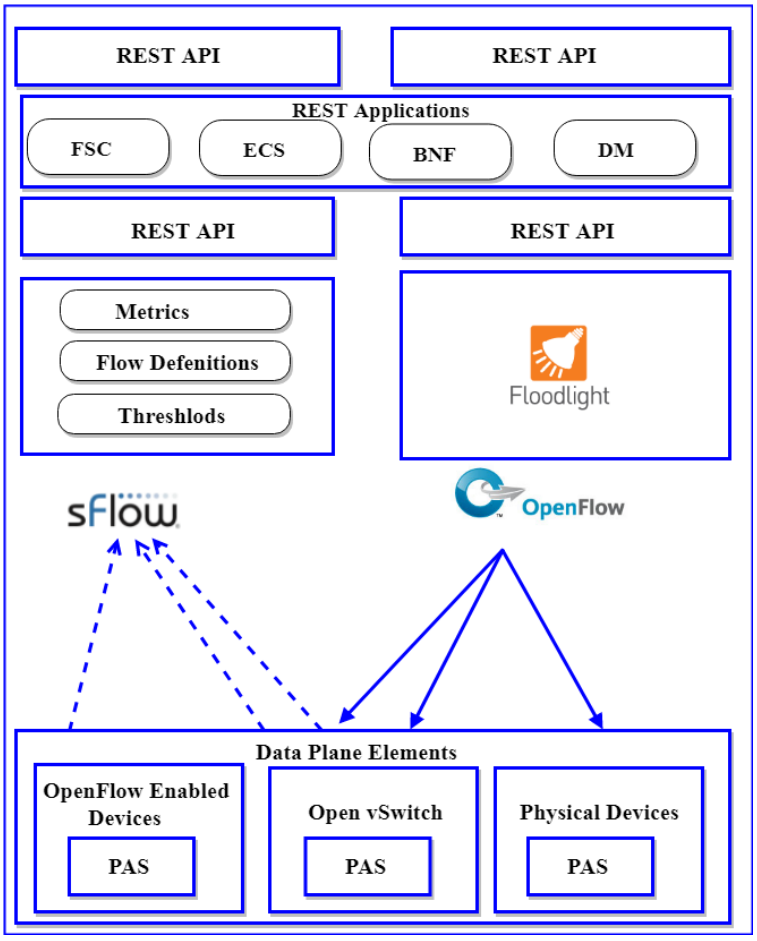


Figure. 2.3. System Architecture.

and more efficient than OF native schemes. The REST API [**48**] is used in the process of detection/mitigation to offer the interoperability in order to manage any SDN controller (*e.g.*, OpenDaylight [**49**], Floodlight [**50**]).

2.6. Proactive And Stateful Scheme (PAS)

PAS is a novel proactive and stateful scheme that is inspired by the techniques of one-to-one mapping between DNS requests and DNS responses [**35**]. First, the SDN controller pushes the OF rules (see Algorithm 1) to OF switches in order to process proactively all DNS traffic (*i.e.*, DNS requests and DNS responses). In PAS, a DNS response is considered as legitimate if it has the same reversed fields' values (*i.e.*, MAC addresses, IP addresses, and UDP ports) of a pre-sent DNS request; otherwise, this DNS response will be considered as illegitimate and systematically eliminated. This allows OF switches to be smart enough to react quickly, to avoid any attempt of external DNS amplification attack that aims to flood the victim's network with amplified DNS traffic, and not wait for a reactive rule from the SDN controller. However, when an attacker is within the victim's network (see Figs.

2.4(c), and 2.4(d)), he can easily spoof the source IP address of the victim in order to direct the amplified DNS responses to that victim. To alleviate this issue, each DNS response received, by each OF switch, is forwarded to the original port (*i.e.*, ingress port) from which the corresponding DNS request came. Then, if the attacker spoofed the source IP address in the prior DNS request, he will receive the returned amplified DNS traffic (see Figs. 2.5(b), 2.5(c), and 2.5(d)). Otherwise, legitimate sources will receive legitimate DNS responses. Thus, PAS totally ensures the protection of the victim from any external or internal DNS amplification attack. PAS uses short timeout (*i.e.*, idle timeout and hard timeout) in order to avoid the storage complexity to maintain the one-to-one mapping. Algorithm 1 illustrates steps executed by each OF switch upon receipt of a packet. Fig. 2.5(a) shows that with WisdomSDN and since there is no DNS request that is sent from a host in the network, the default OF rule (Proto=UDP, port_src=53, prior=0, action=DROP) is triggered in order to drop illegitimate DNS responses; unlike without WisdomSDN, where the victim is flooded with illegitimate amplified DNS responses (see Fig. 2.4(a)). Figs. 2.5(b), 2.5(c) and 2.5(d) show that PAS traces the real path of the DNS requests provenance, using the ingres port in the OF switch, and sends the corresponding responses using the same path as the requests. Thus, the attacker will receive the returned amplified DNS responses, and the victim will not receive any illegitimate DNS traffic; unlike without WisdomSDN, where the victim is flooded with illegitimate amplified DNS responses (see Figs. 2.4(a), 2.4(b), 2.4(c) and 2.4(d)). To protect TCAM of OF switches, which can be the target for attackers, from the huge amount of DNS requests, we propose a machine learning DDoS detection module that consists of FSC, ECS and BNF. In what follows, we investigate how this module can effectively detect illegitimate DNS requests.

2.7. Machine Learning DDoS Detection Module

This module aims to protect TCAM of OF switches while maintaining the SDN secure; it consists of the following: (1) FSC to gather network traffic features in an efficient and scalable way using sFlow protocol; (2) ECS to extract network traffic features; and (3) BNF to, based on ECS calculation, detect network anomalies.

2.7.1. Flow Statistics Collection scheme (FSC)

In SDN environment, two commonly approaches are used to collect network traffic features (*e.g.*, count number of received packets). The first approach is based on OF protocol while the second one is based on flow sampling (*e.g.*, sFlow). In OF based approach, collection of network traffic features can be initiated when the control plane (*i.e.*, SDN controller) sends a state request *(ofp_flow_stats_request)* to data plane devices (*i.e.*, OF switches); these latter respond with one or more reply messages *(ofp_flow_stats_reply)* by sending

network traffic features. To this aim, each OF switch needs to maintain a large number of flow entries. However, this can exhaust TCAM in OF switches. Moreover, a large size of reply messages sent by OF switches to SDN controller can exhaust the bandwidth between data plane and control plane, congest the OF channel and generate a DDoS attack on control plane. Thus, OF based approach is not efficient to detect high rate attacks (*i.e.*, DNS amplification attack).

Algorithm 1: (PAS): Proactive And Stateful Scheme

```
   Input  : DNS packet
   Output: Action to carry out
 1 for each DNS packet do
 2 |   Check Type of packet DNS;
 3 |   if this.dns_packet.type==Request then
 4 |   |   Expect_solution=new Packet() ;
 5 |   |   Expect_solution.eth_src ←this.dns_packet.eth_dst
   |   |   Expect_solution.eth_dst←this.dns_packet.eth_src
   |   |   Expect_solution.ip_src←this.dns_packet.ip_dst
   |   |   Expect_solution.ip_dst←this.dns_packet.ip_src
   |   |   Expect_solution.udp_dst←this.dns_packet.udp_src
   |   |   Install rule that allows only this Expect_solution
   |   |   This.dns_packet::Forward_Routing_Policies()
 6 |   else
 7 |   |   if this.dns_packet.type==Response then
 8 |   |   |   if Match(this.dns_packet)::In_Flow_Table then
 9 |   |   |   |   This.dns_packet::Action(Output:IN_PORT);
10 |   |   |   else
11 |   |   |   |   This.dns_packet::ACTION(DROP);
12 |   |   |   end
13 |   |   else
14 |   |   |   return;
15 |   |   end
16 |   end
17 end
```

To address the issues of OF based approach, we decided to monitor DNS flows based on flow sampling approach using sFlow protocol. This is more scalable and efficient; moreover, it does neither overload the OF channel nor consume bandwidth between control plane and data plane. FSC performs flow aggregation which makes it more adequate to detect high rate DNS amplification attack. In FSC, at each monitoring interval ΔT, the sFlow collector (*i.e.*, sFlow-RT) receives network traffic features from sFlow agents embedded in data plane (*i.e.*, OF switches). Then, ECS computes the entropy values of network traffic. Table 2.1 shows the list of notations used to describe ECS. In this work, we define a

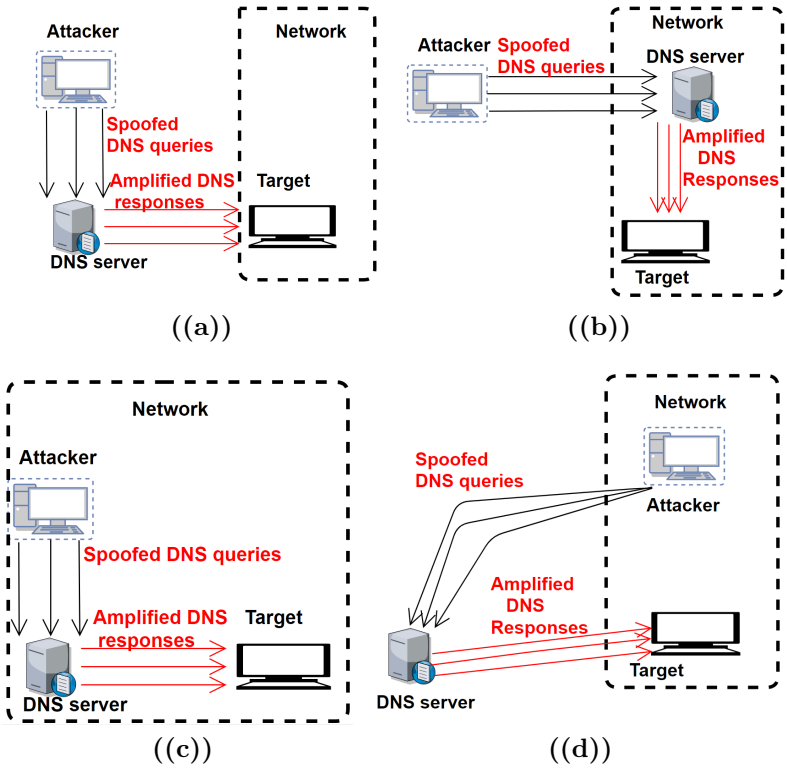


Figure. 2.4. Different attacks setup without WisdomSDN.

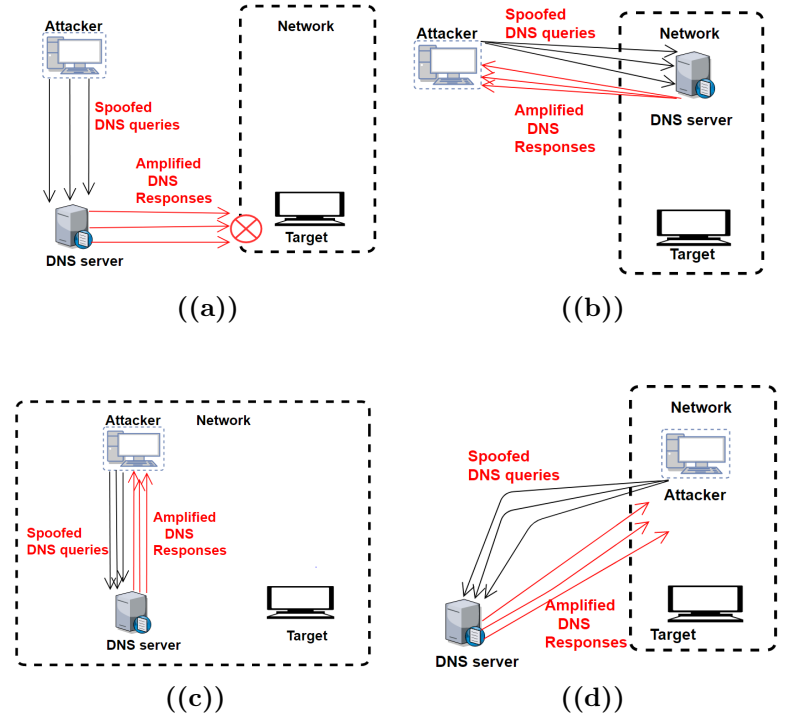


Figure. 2.5. Different attacks setup with WisdomSDN.

Tableau 2.1. Notations

Notations	Definition
MAC_{src}	The MAC source address of a packet
MAC_{dst}	The MAC destination address of a packet
IP_{src}	The IP source address of a packet
IP_{dst}	The IP destination address of a packet
$Port_{src}$	The $Port$ source of a packet
$Port_{dst}$	The $Port$ destination of a packet
IP_{proto}	The transport protocol of a packet (TCP/UDP)
S_j	OF switch S_j
ΔT	The monitoring interval
$F_{i,j}$	Flow f_i at a local OF switch S_j
$p_{i,j}$	The probability of flow f_i over all flows at local OF switch S_j
N	The total number of flows at local OF switch S_j
R	Set of real numbers
I	Set of positive integers

flow as a seven tuple: $\{MAC_{src}, MAC_{dst}, IP_{src}, IP_{dst}, Port_{src}, Port_{dst} = 53, Proto = UDP\}$

2.7.2. Entropy Calculation Scheme (ECS)

The concept behind ECS comes from Shannon's information theory [**51**]. ECS is used to extract network traffic features and calculates entropy values of each flow. When the victim's network is under attack, the number of packets for a given flow that have the same source IP address, denoted IP_{src}, will sharply increase causing a more concentrated distribution of source IP address; while normal state leads to a more dispersed probability distribution of source IP address. The higher entropy values, the more dispersed probability distribution of the source IP address, while low entropy values means the concentration of distribution of source IP address. If the source sends similar requests (*e.g.*, requests from the same source IP address IP_{src}), then its entropy will be low; this means that we are very likely in the presence of illegitimate flow. Therefore, we use ECS to measure the changes of network traffic. A flow is characterized by a sequence of packets that have similar properties reaching the same OF switch S_j during each monitoring interval ΔT. Let $F_{i,j}$ denotes flow f_i at local OF switch S_j; it is defined as follows:

$$F_{i,j}(IP_{src_i},S_j) = \{< IP_{src_i},S_j,t > | S_j \in S, i,j \in I, t \in R\} \tag{2.7.1}$$

where IP_{src_i} is the source IP address of f_i, t is the current timestamp, and $S = \{S_j, j \in I\}$ are the set of OF switches.

Let $|F_{i,j}(IP_{src_i},S_j,t)|$ be the count number of packets of flow $F_{i,j}$ at time t. The variation of the number of packets for flow f_i at local OF switch S_j during ΔT is defined as follows:

$$N_{F_{i,j}}(IP_{src_i},S_j,t+\Delta T) = |F_{i,j}(IP_{src_i},S_j,t+\Delta T)| - |F_{i,j}(IP_{src_i},S_j,t)| \tag{2.7.2}$$

The probability $p_{i,j}$ of flow f_i over all flows at local OF switch S_j is expressed as follows:

$$p_{i,j}(IP_{src_i},S_j) = \frac{N_{F_{i,j}}(IP_{src_i},S_j,t+\Delta T)}{\sum_{i=1}^{N} N_{F_{i,j}}} \tag{2.7.3}$$

where $\sum_{i=1}^{N} p_{i,j}(IP_{src_i},S_j) = 1$. Let IP_{src} be the random variable that represents the number of flows during time interval ΔT. We define the entropy of flows at a local OF switch S_j as follows:

$$H(IP_{src}) = -\sum_{i=1}^{N} p_{i,j}(IP_{src_i},S_j)\log_2 p_{i,j}(IP_{src_i},S_j) \tag{2.7.4}$$

Lemma 1. *The upper and lower bound of $H(IP_{src})$ are, respectively, 0 and $\log_2 N$ (see inequality (2.7.5)).*

$$0 \leq H(IP_{src}) \leq \log_2 N \tag{2.7.5}$$

Démonstration. Let $f(x) = \log_2 x, x \geq 0$. $f(x)$ is a monotonically increasing concave function. Let X be the random variable for flow distribution at an OF switch. Applying Jensen's inequality [**52**] to $f(x)$, we have $Ef(X) \leq f(EX)$. Let $P(X) \doteq \{p_1, p_2, ..., p_n\}$ be the distribution of flows at the OF switch. Then, $\forall\, p_i, 0 \leq p_i \leq 1, \sum_{i=1}^{N} p_i f(x_i) \leq f(\sum_{i=1}^{N} p_i x_i)$, where $\sum_{i=1}^{N} p_i = 1$. Especially $H(IP_{src}) = \sum_{i=1}^{N} p_i \log_2(\frac{1}{p_i}) \leq \log_2(\sum_{i=1}^{N} p_i \frac{1}{p_i}) = \log_2 N$. Then, $H(IP_{src}) \leq \log_2 N$. Since $\log_2 \frac{1}{p_i} \geq 0, \forall\, p_i, 0 \leq p_i \leq 1$. Then, $H(IP_{src}) \geq 0$, and further Eq. (5.5.6) holds.

□

In order to normalize the entropy values, to have a measurement metric which is totally independent from the number of distinct entropy values, we divide them by the maximum value which is $\log_2 N$, as is demonstrated in Eq. (5.5.6). Therefore, the normalized entropy values are in [0, 1] and are defined as follows:

$$H(IP_{src})' = \frac{H(IP_{src})}{\log_2 N} \tag{2.7.6}$$

Lemma 2. *When the network is under DNS amplification attack, the upper bound of entropy variation at local OF switch S_j decreases sharply in comparison to the normal, non-attack, case.*

DÉMONSTRATION. As we proved in Eq. (2.7.5), the entropy of flows reaches the upper bound, $\log_2 N$, when the probability distribution of source IP address is almost even. The variation number of packets for each flow f_i is almost stable, specifically, $p_1 = p_2 = p_3 = ... = p_N$, and it reaches the lower bound, $H(IP_{src}) = 0$, when the probability distribution of source IP address is almost uneven. Especially, $p_i = 1, 0 \leq i \leq N, p_k = 0, \forall 0 \leq k \leq N, k \neq i$. Since the entropy is a monotonic function [**51**]; therefore, when the victim's network is under DNS amplification attack, the distribution of source IP address moves toward the extreme unbalanced point. As result, the upper bound of the entropy variation decreases sharply.

□

Theorem 1. *We divide the state of the network into two segments: normal state (i.e., non-attack), and under DNS amplification attack. We denote $H_{leg}(IP_{src})$ and $H_{illeg}(IP_{src})$ as, respectively, the entropy value of each flow at local OF switch S_j in normal state and under attack. When the attacker starts an attack towards a specific victim (i.e., target of DNS amplification attack), the number of packets that have the same IP_{src} will increase quickly, which leads to a significant decrease of entropy, especially, $H_{leg}(IP_{src}) \gg H_{illeg}(IP_{src})$. High entropy values lead to a dispersed probability distribution of flows, whilst low entropy values indicate a concentrated probability distribution.*

DÉMONSTRATION. Let $\Omega(x) = x \log_2 x$, $x \geq 0$. $\Omega(x)$ is a monotonically increasing convex function. Let X be the random variable for flow distribution at an OF switch S_j. Applying Jensen's inequality to $\Omega(x)$, we have $E\Omega(X) \geq \Omega(EX)$. Let $\Psi(X) \doteq \{\Psi_1, \Psi_2, ..., \Psi_n\}$ be the distribution of flows at an OF switch S_j, $\Psi^{leg}(X^{leg}) \doteq \{\Psi_1^{leg}, \Psi_2^{leg}, ..., \Psi_n^{leg}\}$ be the distribution of normal case and $\Psi^{illeg}(X^{illeg}) \doteq \{\Psi_1^{illeg}, \Psi_2^{illeg}, ..., \Psi_n^{illeg}\}$ be the distribution of attack case. As the network is stable in the normal case; therefore, $H_{leg}(IP_{src})$ is also stable. Then, $\forall i, 0 \leq i \leq N$, $\Psi_i^{leg} \ll \Psi_i^{illeg}$. More, $EX^{leg} \ll EX^{illeg}$ and $\Omega(EX^{leg}) \ll \Omega(EX^{illeg})$. Therefore, $E(\Omega(X^{leg})) \ll E(\Omega(X^{illeg}))$. Hence, we have: $-\sum_{i=1}^{N} \Psi_i^{leg} \log_2 \Psi_i^{leg} \gg -\sum_{i=1}^{N} \Psi_i^{illeg} \log_2 \Psi_i^{illeg}$. Finally, the result is $H_{leg}(IP_{src}) \gg H_{illeg}(IP_{src})$.

□

The attribute (*i.e.*, header fields of the packet) used to aggregate incoming flows at a local OF switch S_j depends on the scenario of the attack under consideration (*e.g.*, DNS amplification attack). For example, many attackers sends multiple illegitimate DNS requests from the same UDP port source with the same DNS request type (*e.g.*, ANY), as legitimate machines send legitimate DNS requests from random UDP port source with different DNS request types (*e.g.*, A, MX, NS, etc.). Consequently, we use $\{IP_{src}$, $Port_{src}$ and $ANY\}$ as attributes to aggregate DNS flows. Similarly, we can also define the UDP port source entropy, $H(Port_{src})'$ and ANY entropy, $H(ANY)'$. Finally, we represent the network traffic

features at the k^{th} time period as:

$$X_k = \{H(IP_{src})'_k, H(Port_{src})'_k, H(ANY)'_k\} \tag{2.7.7}$$

2.7.3. Bayes Network based Filtering scheme (BNF)

BNF aims to detect illegitimate DNS requests that overload the network and may exhaust TCAM in OF switches. First, we describe the flow representation. Then, we discuss BNF criterion of classification.

2.7.3.1. Flow representation. We represent each sample in BNF by a vector $x = (x_1, x_2, x_3)$ where x_1, x_2, x_3 are values taken, respectively, by random variables $H(IP_{src})'$, $H(Port_{src})'$ and $H(ANY)'$. When the network suffers from DNS amplification attack, the number of DNS requests sent from the same IP_{src}, with the same type (*e.g.*, ANY) and from the same $Port_{src}$, will increase sharply. This leads to a significant decrease of entropy values of, respectively, $H(IP_{src})'$, $H(Port_{src})'$ and $H(ANY)'$. Therefore, the vector X_k can better represent the DNS amplification attack characteristics.

2.7.3.2. Criterion of classification. BNF is a binary classifier that consider two classes of DNS requests: (1) legitimate DNS requests, denoted by leg, and (2) illegitimate DNS requests, denoted by $illeg$. The class of X_k, denoted by c, can be either leg or $illeg$ and is defined as follows:

$$c = \underset{c \in \{leg, illeg\}}{\arg\max}\ p(c|X_k)$$

Since $p(leg|X_k) + p(illeg|X_k) = 1$; thus, the selection criterion becomes:

$$X_k \text{ is illegitimate iff } p(illeg|X_k) \geq 0.5 \tag{2.7.8}$$

According to Bayes theorem [**53**], the probability of vector X_k to belong to class c is defined as follows:

$$p(C = c|X = X_k) = \frac{p(C = c).p(X = X_k|C = c)}{p(X = X_k)} \tag{2.7.9}$$

Using the total probability theorem, we conclude:

$$p(C = c|X = X_k) = \frac{p(C = c).p(X = X_k|C = c)}{\sum_{c \in \{leg, illeg\}} p(C = c)p(X = X_k|C = c)} \tag{2.7.10}$$

Therefore, the selection criterion can be expressed as follows: $X_k$is illegitimate iff:

$$p(C = c|X = X_k) = \frac{p(C = c).p(X = X_k|C = c)}{\sum_{c \in \{leg, illeg\}} p(C = c)p(X = X_k|C = c)} \geq 0.5 \tag{2.7.11}$$

$H(IP_{src})'$, $H(Port_{src})'$ and $H(ANY)'$ are conditionally independent variables given class c. Let $p_k(leg)$ and $p_k(illeg)$ denote, respectively, the conditional probabilities that vector X_k is legitimate and illegitimate.

Using Eq. 2.7.11, the selection criterion becomes as follows: X_k is illegitimate iff:

$$\frac{\prod_{k=1}^{n} p_k^{X_k}(illeg)(1-p_k(illeg))^{1-X_k} p(illeg)}{\prod_{k=1}^{n} p_k^{X_k}(illeg)(1-p_k(illeg))^{1-X_k} p(illeg) + \prod_{k=1}^{n} p_k^{X_k}(leg)(1-p_k(leg))^{1-X_k} p(leg)} \geq 0.5 \tag{2.7.12}$$

When $p(leg) = p(illeg)$, the selection criterion become as follows: $X_k$is illegitimate iff:

$$\frac{\prod_{k=1}^{n} p_k^{X_k}(illeg)(1-p_k(illeg))^{1-X_k}}{\prod_{k=1}^{n} p_k^{X_k}(illeg)(1-p_k(illeg))^{1-X_k} + \prod_{k=1}^{n} p_k^{X_k}(leg)(1-p_k(leg))^{1-X_k}} \geq 0.5 \tag{2.7.13}$$

BNF is trained and then used to classify the k^{th} vector X_k as either legitimate or illegitimate. By combining ECS and BNF, WisdomSDN can accurately detect illegitimate DNS requests in real time with low false positive rates while maintaining a high detection rate.

2.7.4. Machine Learning DDoS Detection Algorithm

After having explained our machine learning DDoS detection Module, in this section we summarize this calculation via an algorithm (see Algorithm 2). This algorithm is implemented on the top of the SDN controller as a REST application and allows, using sFlow protocol, the monitoring of each incoming flow; it defines some monitoring metrics (*e.g.*, address groups, attributes of flows aggregation, and thresholds) and commands the sFlow collector to deploy these monitoring metrics within the data plane using sFlow protocol. Using the collected data, the algorithm detects illegitimate flows.

2.8. DNS Mitigation Scheme (DM)

When BNF detects illegitimate DNS requests (*i.e.*, illegitimate traffic features vector X_k), a mitigation rule is executed in order to protect TCAM of OF switches. DM installs new OF rules into OF switches under DNS amplification attack; these rules have a high priority to match illegitimate flows and monitor their speed. OF protocol was designed without any support to QoS features; however, OF 1.3 introduces *m*eters [**54**] to OF. DM specifies for each flow entry in OF table a *meter*; *meter* entries with different *Meter_id* are deployed to monitor the speed of the classified illegitimate traffic features vector X_k by BNF; if the packet rate surpasses the band (*i.e.*, rate limiter); then, DM systematically drops the illegitimate DNS requests.

Algorithm 2: Machine Learning DDoS Detection Algorithm.

Input : Aggregated DNS requests from OF Ingress port
Output: legitimate or illegitimate

1. Define address groups, the attribute to aggregate flows denoted as *att* (*i.e.*, $IP_{src}, Port_{src}, ANY$) and initialize the monitoring interval ΔT.
   2. Identify flow f_i, $\forall 0 \leq i \leq N$, and set the count number of packets for each flow f_i to zero.
   3. When the monitoring interval ΔT is over, the entropy values are calculated as follows:
   for *each flow* f_i *at a local OF switch* S_j **do**
2. $\quad N_{F_{i,j}}(att_i,S_j,t+\Delta T) = |F_{i,j}(att_i,S_j,t+\Delta T)| - |F_{i,j}(att_i,S_j,t)|$
3. **end**
4. **for** $i \leftarrow$*1 to* N **do**
5. $\quad p_{i,j}(att_i,S_j) = \frac{N_{F_{i,j}}(att_i,S_j,t+\Delta T)}{\sum_{i=1}^{N} N_{F_{i,j}}}$
   $\quad H(IP_{src}) += -p_{i,j}(IP_{src_i},S_j)\log_2 p_{i,j}(IP_{src_i},S_j)$
   $\quad H(Port_{src}) += -p_{i,j}(Port_{src_i},S_j)\log_2 p_{i,j}(Port_{src_i},S_j)$
   $\quad H(ANY) += -p_{i,j}(ANY_i,S_j)\log_2 p_{i,j}(ANY_i,S_j)$
6. **end**
7. *4. Calculate the normalized entropy values as follows:*
   $H(IP_{src})' = \frac{H(IP_{src})}{\log_2 N}$
   $H(Port_{src})' = \frac{H(Port_{src})}{\log_2 N}$
   $H(ANY)' = \frac{H(ANY)}{\log_2 N}$
   5. Calculate the network traffic features at the k^{th} *time period as:*
   $$X_k = \{H(IP_{src})'_k, H(Port_{src})'_k, H(ANY)'_k\}$$
   6. **if** *p(illeg* $|X_k) \geq 0.5$ **then**
8. $\quad$ *notifies DM*
   $\quad$ *Go to step 2*
9. **else**
10. $\quad$ *Go to step 2*
11. **end**
12.

2.9. Evaluation of WisdomSDN

In this section, we present the evaluation of WisdomSDN. First, we introduce the experimental environment. Then, we evaluate the performance of WisdomSDN.

2.9.1. Experimental Environment

PAS is implemented in OF switch (*i.e.*, OpenvSwitch); while Machine Learning DDoS Detection Module (*i.e.*, FSC, ECS and BNF) and DM are implemented as applications on

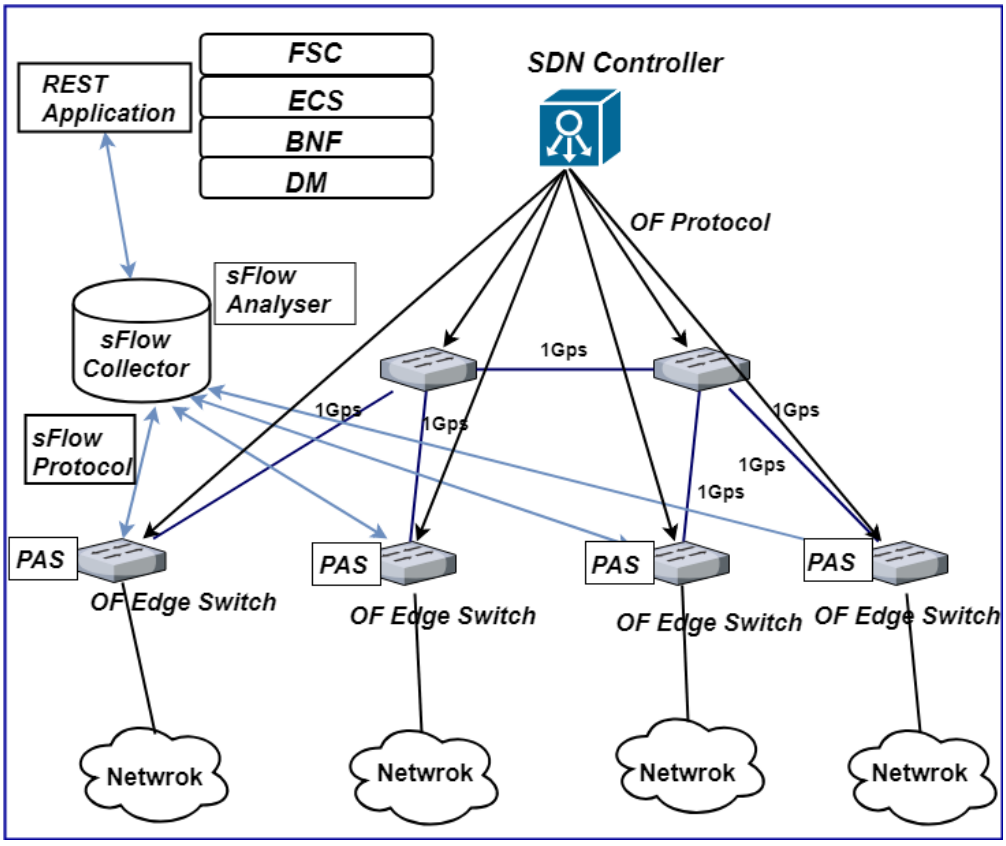
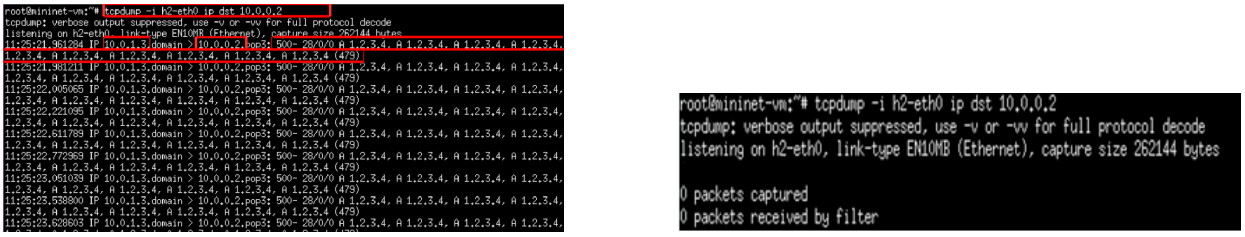


Figure. 2.6. Experimental Environment.

the top of the SDN controller. PAS systematically blocks amplified illegitimate DNS responses. Using FSC, ECS extracts network traffic features; then, BNF detects automatically illegitimate DNS requests. Finally, DM mitigates the illegitimate DNS requests. In order to emulate a real network environment, we use Mininet, a popular SDN emulation tool. Mininet uses Linux containers and virtual OF switches (*e.g.*, OpenvSwitch) to allow realistic virtual networks of switches and hosts to be constructed using a virtual machine (VM). In our testing environment, the network monitor (*i.e.*, sFlow-RT) and the SDN controller (*i.e.*, Floodlight) are installed on the host-system. We run our experiments on a PC with CPU Intel Core i7-8750H-2.2 GHz and 16GB RAM. Fig. 2.6 shows the experimental environment (*i.e.*, components of the testbed); it consists of: (a) SDN controller (*i.e.*, Floodlight); (b) sFlow collector (*i.e.*, sFlow-RT): to perform the monitoring of network traffic features (sFlow-RT can perform monitoring of 7500 switch ports in data center networks); (c) REST applications: it executes FSC, ECS, BNF and DM; and (d) 6 OF switches, that are connected together via 1 Gbps links. Each OF network contains more than 20 hosts; multiple hosts in our setup topology are simulated to act as Open Resolvers and send amplified DNS responses (we tested several attacks setup: when the attacker is external (see Figs. 2.4(a) and 2.4(b)), and then when it is internal (see Fig. 2.4(c) and 2.4(d)) and other hosts are

Tableau 2.2. Parameter values.

	Average traffic rate (Mbps)	Sampling rate	Attack rate(pkts/s)
Exp.1	100	1/64	200-500
Exp.2	500	1/256	1000-2500



((a)) ((b))

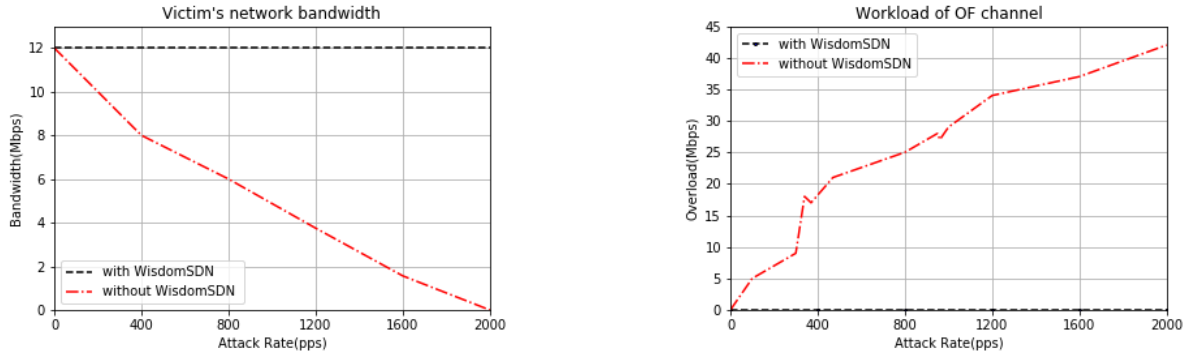
Figure. 2.7. Capture on victim's network without and with wisdomSDN.

legitimate hosts. In order to test the scalability of WisdomSDN, we varied the rate of attack from 100 to 500 Mbps. Table 2.2 shows the parameter values for our experimentations.

For the attack script, Scapy's Python library [**55**] module of Python that makes easy to forge, send, receive and manipulate network packets, is used to forge DNS request. NodeJs [**56**] is used to create the DNS server using dnsd package; in order to emulate a real DNS amplification attack, the DNS server is implemented to send amplified DNS responses. Tcpdump [**57**], Wireshark [**58**] and Iperf [**59**] are used to capture and analyzes the network traffic in real-time. For each scenario presented in Figs. 2.4 and 2.5, we tested several cases of attacks and made test measurements considering both cases: (1) without wisdomSDN; and (2) with wisdomSDN.

Fig. 2.7(a) shows that, without wisdomSDN, the victim (*i.e.*, host h2) is flooded with amplified illegitimate DNS responses. After deploying WisdomSDN, we re-launched the attack. Fig. 2.7(b) shows that, the victim does not receive any amplified illegitimate DNS responses.

Figs. 2.8(a) and 2.8(b) show the protection of the victim's bandwidth resources and OF channel by WisdomSDN. Fig. 2.8(a) shows the victim's bandwidth resources consumption with and without WisdomSDN. Without WisdomSDN, with the increase of attack rate, the victim's network bandwidth consumption decreases sharply (victim's network bandwidth goes to zero when the attack rate reaches 2000 packets per second (pps)). With WisdomSDN, the SDN controller set flow rules to OF switch (see Algorithm 1) to process proactively DNS packets and to: (1) systematically drop the amplified DNS responses; (2) perform one-to-one mapping between DNS requests and DNS responses; and (3) send each DNS response according to the original port, if the source IP address of DNS request has been spoofed,



((a)) ((b))

Figure. 2.8. Performance evaluation of WisdomSDN in terms of: (a) bandwidth consumption; (b) OF channel workload.

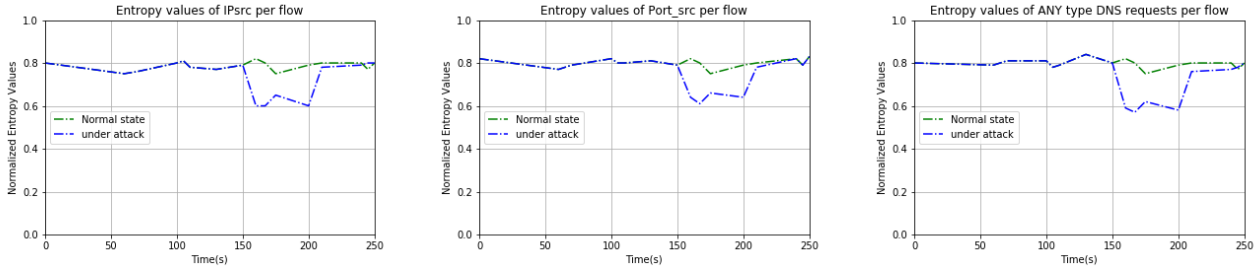
the attacker will receive the illegitimate DNS response. Thus, the victim is totally protected and his network bandwidth is not used by the traffic generated by the attack (12Mbps of available bandwidth) even if the attack rate reaches 2000 pps. PAS not only saves the victim's network bandwidth, but also reduces the load of OF channel. Fig. 2.8(b) shows OF channel's workload variations with and without WisdomSDN. Without WisdomSDN, with the increase of attack rate, the load of OF channel increases sharply. With WisdomSDN, we use our proactive mechanism (*i.e.*, PAS) and not a reactive one. In PAS, each OF switch makes filtering of illegitimate DNS responses from all DNS traffic without sending amplified traffic to SDN controller. Thus, saving the load of OF channel.

Fig. 2.9(a) shows that, without WisdomSDN, the attack rate reaches more than 2000 DNS requests per second. Thus, even if the victim is protected using PAS, the OF switch can be overwhelmed. This occurs when the attacker is within the network of the victim and tries to overwhelm TCAM of OF switches with a large number of illegitimate DNS requests. However, when our machine Learning DDoS detection module is deployed, the traffic, generated by the attack. is effectively monitored using FSC; when ECS and BNF classify the flow as illegitimate, it is stopped and DM automatically mitigates the traffic attack (rate limit). Fig. 2.9(b) shows that the time taken to mitigate the attacks is less than 13 seconds. Thus, we can effectively and quickly recover the network in short time.

To examine the effectiveness of ECS, we set a simulation interval of $250s$; then, we launch the attack during the interval of $150-200s$. Figs. 2.10(a), 2.10(b) and 2.10(c) show, respectively, the normalized entropy values of IP source address, UDP port source and ANY. These normalized entropy values decreases rapidly in the interval of attack $150-200s$. Thus, ECS can better represents the attack and indicates that we are very likely in the presence of illegitimate flow. Therefore, BNF can detect the attack with a high detection rate and low false positive rates.

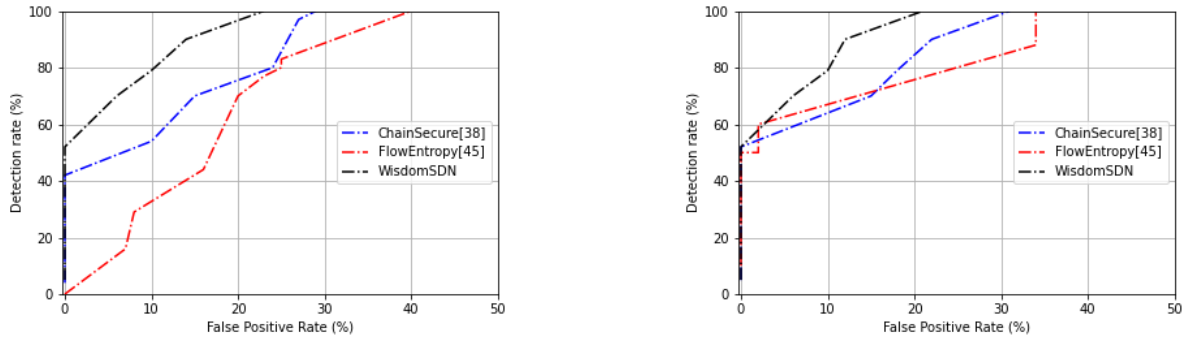
((a)) ((b))

Figure. 2.9. Performance evaluation of WisdomSDN in terms of: (a) DNS request's traffic before and after enabling WisdomSDN; and (b) Time of DM mitigation.



((a)) ((b)) ((c))

Figure. 2.10. Normalized entropy values of: (a) source IP address (IP_{src}); (b) UDP port Source ($Port_{src}$); and c) ANY DNS requests.



((a)) ((b))

Figure. 2.11. ROC curves for the: (a) 100 Mbps case; (b) 500 Mbps case.

2.9.2. Performance Evaluation

The performance of BNF is measured using ROC curves. The ROC curve shows the True Positive Rate (TPR) called sensitivity or Detection Rate (DR) according to the False Positive Rate (FPR). To evaluate the detection rate of BNF in high traffic rate, we conducted

two experiments (*i.e.*, $100Mbps$ and $500Mbps$) and we compared BNF in terms of detection rate and FPR with ChainSecure [**38**] and FlowEntropy [**45**]. To measure the performance of BNF, we use confusion matrix (see Table 2.3); it provides several metrics that can help to study the performance of BNF. We also define DR and FPR as follows:

Tableau 2.3. Confusion matrix.

	Classified as illegitimate	Classified as legitimate
illegitimate flows	TP (True Positives)	FN (False Negatives)
legitimate flows	FP (False Positives)	TN (True Negatives)

$$Se = DR = \frac{TP}{TP + FN}, 1 - Sp = FPR = \frac{FP}{TN + FP}$$

where, TP represent the illegitimate flows that are correctly classified as illegitimate, FN represent the illegitimate flows that are identified as legitimate, FP represent the legitimate flows that are classified as illegitimate, and TN represent the legitimate flows that are correctly identified as legitimate. Fig. 2.11(a) shows that WisdomSDN achieves around 100% detection rate for $100Mbps$ case while it has just 23% of FPR, while [**38**] and [**45**] achieve the same detection rate but with respectively 31% and 40% of FPR. Fig. 2.11(b) shows that WisdomSDN achieves around 100% detection rate for 500 Mbps case while it has just 21% of FPR, while [**38**] and [**45**] achieve the same detection rate but with respectively 30% and 34% of FPR.

2.10. Conclusion

SDN is an emerging technology that brings numerous benefits by decoupling the control plane from data plane. On one hand, the separation of the control plane from the data plane allows for more control over the network and brings new capabilities to deal with large forms of DDoS attacks. On the other hand, this separation introduces new challenges regarding the security of the control plane. This chapter aims to deal with DNS amplification attack while maintaining the SDN secure (*i.e.*, protecting the resources of data plane (*i.e.*, Ternary Content Addressable Memory (TCAM) of OF switches). For this aim, first, we proposed PAS, a proactive and stateful scheme that performs a one-to-one mapping between DNS request and DNS response in order to: (1) protect the victim from DNS amplification attack; and (2) protect the resources of the SDN controller. Then, we proposed a machine learning DDoS detection module that consists of: FSC, ECS and BNF in order to detect illegitimate DNS requests and protect TCAM of OF switches. Finally, DM is designed to mitigate illegitimate DNS requests. In our simulations, we set a fixed idle and hard timeouts of flow rules. For large values, flow rules stay in OF table for a long time which can exhaust

TCAM of OF switches, while too small values, may lead to the dropping of legitimate DNS responses. For future work, we intend to design a novel optimization algorithms to set dynamically those timeouts. This optimization can be based on the capacity of OF table, traffic rate and workload of both data plane and control plane. In the following chapter, we extend this work considering inter-domain mitigation based on a decentralized architecture (*e.g.*, Blockchain) ensuring two levels of mitigation (*i.e.*, intra-domain and inter-domain DDoS mitigation).

Chapitre 3

Cochain-SC: An Intra and Inter-Domain Ddos Mitigation Scheme Based on Blockchain Using SDN and Smart Contract

3.1. Abstract

With the exponential growth in the number of insecure devices, the impact of Distributed Denial-of-Service (DDoS) attacks is growing rapidly. Existing DDoS mitigation schemes are facing obstacles due to low flexibility, lack of resources and high cost. The new emerging technologies, such as blockchain, introduce new opportunities for low-cost, efficient and flexible DDoS attacks mitigation across multiple domains. In this chapter, we propose a blockchain-based approach, called Cochain-SC, which combines two levels of mitigation, intra-domain and inter-domain DDoS mitigation. For intra-domain, we propose an effective DDoS mitigation method in the context of software defined networks (SDN); it consists of 3 schemes: (1) Intra Entropy-based scheme (I-ES) to measure, using sFlow, the randomness of data inside the domain; (2) Intra Bayes-based scheme (I-BS) to classify, based on entropy values, illegitimate flows; and (3) Intra-domain Mitigation (I-DM) scheme to effectively mitigate illegitimate flows inside the domain. For inter-domain, we propose a collaborative DDoS mitigation scheme based on blockchain; it uses the concept of smart contracts (*i.e.*, Ethereum's smart contracts) to facilitate the collaboration among SDN-based domains (*i.e.*, Autonomous System: AS) to mitigate DDoS attacks. For this aim, we design a novel and secure scheme that allows multiple SDN-based domains to securely collaborate and transfer attack information in a decentralized manner. Combining intra-domain and inter-domain DDoS mitigation, Cochain-SC allows an efficient mitigation along the path of an ongoing attack and an effective mitigation near the origin of the attack. This allows reducing the enormous cost of forwarding packets, across multiple domains, which consist mostly of useless amplified attack traffic. To the best of our knowledge, Cochain-SC is

the first scheme that proposes to deal with both intra-domain and inter-domain DDoS attacks mitigation combining SDN, blockchain and smart contract. The implementation of Cochain-SC is deployed on Ethereum official test network Ropsten. Moreover, we conducted extensive experiments to evaluate our proposed approach; the experimental results show that Cochain-SC achieves flexibility, efficiency, security, cost effectiveness and high accuracy in detecting illegitimate flows, making it a promising approach to mitigate DDoS attacks.

Keywords: DDoS; Entropy; Bayes Classifier; SDN; intra-domain mitigation; inter-domain collaboration; Blockchain technology; Smart contracts.

Status: This journal paper is published. Z. Abou El Houda, A. S. Hafid and L. Khoukhi, **"Cochain-SC: An Intra- and Inter-Domain Ddos Mitigation Scheme Based on Blockchain Using SDN and Smart Contract,"** IEEE Access, vol. 7, pp. 98893-98907, 2019, doi: 10.1109/ACCESS.2019.2930715

3.2. Introduction

In recent years, security threats of DDoS attacks have been increasing causing severe collateral damage to network operators as well as Internet service providers (ISPs). Some recent dramatic incidents (*e.g.*, DDoS attacks against US banks [**60**]) show that DDoS attacks are becoming powerful, devastating and more destructive plaguing network operators and ISPs in a stealthy way [**61**]. Indeed, the rapid growth in the number of insecure devices, with an estimated 75 billion devices by the end of 2025 [**9**], can facilitate and enhance the capability of attacks. For example, on the 2 of October 2016, a huge and dramatic attack, exceeding a rate of 1 Tbit/s, was conducted against Dyn Domaine Name System (DNS) services. As a consequence, many popular Internet services, *e.g.*, Amazon and GitHub [**14**] became disconnected for several hours [**6**]. Such incidents damage ISPs and cost millions of dollars of lost revenues for enterprises [**62**].

Recently, SDN has attracted tremendous attention as a novel technology that facilitates network management and provides new capabilities to manage and deploy networks dynamically. SDN separates data and control planes; this separation allows for more control over the network and brings a new way to deal with various form of DDoS attacks.

To deal with DDoS attacks inside a domain (*i.e.*, AS), we propose an effective DDoS mitigation method in the context of SDN; it consists of 3 schemes: (1) an intra Entropy-based scheme (I-ES) to measure, using sFlow [**63**], the randomness of data inside the SDN based domain; (2) an intra Bayes-based scheme (I-BS) to classify, based on entropy values, illegitimate flows; and (3) an intra-domain Mitigation (I-DM) scheme to effectively mitigate

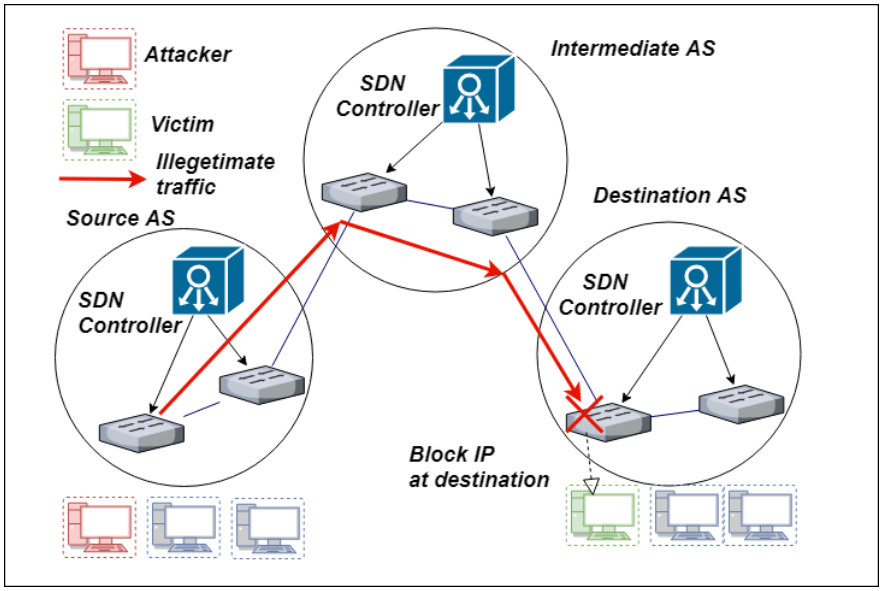


Figure. 3.1. Concept of DDoS attacks across multiple SDN domains.

illegitimate flows inside the domain. However, an intra-domain DDoS mitigation alone cannot mitigate DDoS attacks on the paths to the victim because the attackers and the target are not necessarily in the same AS (see Fig. 3.1). Thus, there is a need to an efficient inter-domain collaboration to: (a) effectively reduce the enormous cost of forwarding packets, across multiple domains, which are mostly useless amplified attack traffic; and (b) block the attack close to its source. Existing collaborative DDoS mitigation schemes suffer from low flexibility and high cost; more importantly, they are centralized. The centralized solution, by its nature, causes single-point-of-failure and is vulnerable to DDoS attacks that can make it difficult, or even infeasible, to share information, among ASs, and make effective decisions to mitigate the attacks.

The new emerging technologies, such as blockchain and smart contract, open new opportunities for low-cost, efficient and flexible collaboration across multiple ASs to mitigate DDoS attacks. Indeed, blockchain has been shown to provide a decentralized collaboration in trustless network environments. Blockchain technology (*e.g.*, Bitcoin [**64**], Ethereum [**8**] and Zcash [**65**]) has proven its effectiveness and success in achieving high level of security and transparency in financial field [**66**] as well as non-financial fields (*e.g.*, Healthcare [**67**], Decentralized IoT [**68**], Decentralized Data Sharing [**69**], and online advertising [**70**]). The inter-domain collaboration process may exploit the power of the decentralized approach by replicating data on each network's node, thus enforcing integrity and reliability of the whole ecosystem.

For inter-domain, we propose a collaborative DDoS mitigation scheme based on blockchain. This scheme uses the concept of smart contracts (*i.e.*, Ethereum's smart contracts)

to facilitate the collaboration among SDN-based domains (*i.e.*, AS) to mitigate DDoS attacks. We design a novel and secure scheme that allows multiple SDN-based domains to securely collaborate and transfer attack information in a decentralized manner. This allows an SDN-based domain to effectively collaborate and inform its neighboring domains about ongoing DDoS attacks. Thus, by combining our scalable intra-domain mitigation approach based on SDN and inter-domain DDoS mitigation scheme, we are not only able to efficiently mitigate DDoS attacks within the target's domain, but also to share attack information in a fully decentralized manner resulting in effective decision making, by multiple domains, to mitigate against DDoS attacks.

This chapter presents the design, specification and implementation of a blockchain-based approach called Cochain-SC in which two levels of mitigation are combined (*i.e.*, intra-domain and inter-domain DDoS mitigation). Cochain-SC provides an efficient mitigation along the path of an ongoing attack and effective mitigation near of source of the attack. The implementation of Cochain-SC is deployed on Ethereum official test network Ropsten [**71**], an open blockchain platform.

The main contributions of this chapter can be summarized as follows:

- We design a decentralized secure DDoS mitigation scheme (Cochain-SC) based on blockchain using smart contract; it supports two levels of mitigation: intra-domain and inter-domain DDoS mitigation.
- We propose a smart contract-based scheme, that makes use of Ethereum's smart contract technology, to realize a decentralized, secure, flexible and low-cost collaboration, among multiple SDN-based domains, to mitigate DDoS attacks.
- We propose an Intra Entropy-based scheme (I-ES), to measure the randomness of data inside the domain using sFlow.
- We propose an Intra detection scheme, called Intra Bayes-based scheme (I-BS), to automatically detect network traffic anomalies inside the domain.
- We propose an Intra-Domain Mitigation (I-DM) scheme to effectively mitigate illegitimate flows inside the domain.
- We evaluate the performance of Cochain-SC in terms of flexibility, efficiency, security and cost effectiveness. The experiments results show that Cochain-SC can effectively mitigate the attack inside the domain with high accuracy, low false positive rates and a low overhead, and achieves the requirements of the new generation of flexible, secure, efficient, and low-cost inter-domain DDoS collaboration schemes.

The rest of the chapter is organized as follows. Section 3.3 presents related work. Section 3.4 presents an overview of Cochain-SC. Section 3.5 presents our intra-Domain DDoS mitigation approach, which consists of an Intra Entropy-based scheme (I-ES), an intra Bayes-based scheme (I-BS) and an Intra-Domain Mitigation (I-DM) scheme. Section 3.6 presents

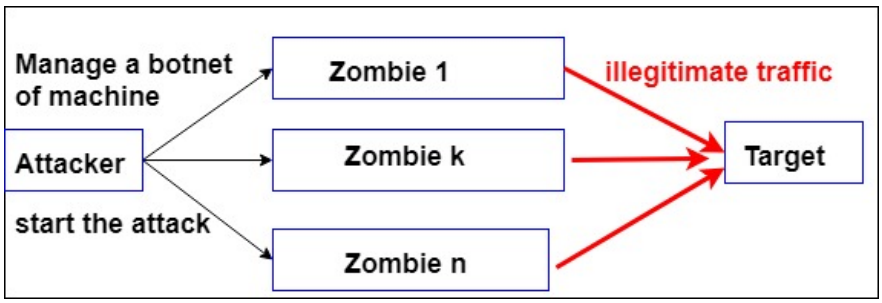


Figure. 3.2. DDoS attacks.

our inter-domain DDoS collaboration scheme. Section 3.7 presents the implementation of Cochain-SC. Section 3.8 evaluates Cochain-SC. Finally, Section 3.9 concludes the chapter.

3.3. Background And Related Work

Several schemes have been proposed in the literature to mitigate DDoS attacks. In this section, we briefly introduce DDoS attacks. Then, we overview existing DDoS mitigation schemes that can be classified in two categories: (1) intra-domain DDoS mitigation schemes; and (2) inter-domain DDoS mitigation/collaboration schemes.

3.3.1. DDoS Attacks Techniques

DDoS attacks aim to overwhelm target's resources, such as network bandwidth and computer CPU, with illegitimate flows. To launch DDoS attacks, the attacker (*e.g.*, bot master), generally, needs to control a large number of compromised devices (called zombies). Each zombie sends a huge volume of illegitimate traffic, to deny services, to legitimate users of the target (see Fig. 3.2). There are two major categories of DDoS attacks: real source IP-based attacks (typical DDoS attacks) and DRDoS (Distributed Reflection Denial of Service) attacks. In a typical DDoS attack, the bot master orders a large number of zombies to directly flood the target. DRDoS attacks consist of bot master, zombies and reflectors. Each zombie is ordered by the bot master to send a large number of packets, in which the source IP address is replaced with the victim's IP address, to other devices known as reflectors; upon receipt of these packets, the reflectors send the victim a huge volume of illegitimate traffic. In this chapter, we consider typical DDoS attacks. Also, we combine an intra-domain DDoS mitigation scheme using machine learning scheme (*i.e.*, I-BS) to enhance the accuracy of the detection model. Also, we integrate an inter-domain DDoS collaboration scheme that allows for an efficient mitigation along the path of an ongoing attack and effective mitigation near to the origin of the attack.

3.3.2. Intra-Domain DDoS Mitigation Schemes

Several intra-domain DDoS mitigation schemes have been proposed; in the following, we present some of the most prominent as well as their limitations.

The BCP 38 standard [**72**] proposes a filtering method that requires every ISP to: (a) verify that the packets from its network use valid prefixes (IP addresses); and (b) filter these packets that use forged source addresses that are outside its range of legitimate addresses. This type of solution has proven to be effective against IP-spoofing attacks. However, it does absolutely nothing to deal with real source IP-based attacks. In [**73**], Rodrigo et al. proposed a flow-based intrusion detection scheme (*i.e.*, Self Organizing Maps) using OpenFlow (OF) protocol to gather traffic flow statistics. This scheme [**73**] did not consider the overhead to the control plane caused by the flow collecting process using OF protocol; moreover, performance analysis does not include the overall system performance. In [**74**], Mehdi et al. proposed an anomaly detection scheme in the context of SDN using OF protocol. However, the scheme was focused only on the home environment (*i.e.*, small-scale setup); in large-scale environments, a high rate data traffic to SDN controller may overload the control plane. In [**75**], Yu et al. proposed OpenSketch, a software defined traffic measurement scheme. OpenSketch provides an efficient method to collect measurement data through a three-stage pipeline (hashing, filtering, and counting). However, OpenSketch sends all the counters to the SDN controller for analysis, which may overload the control plane. In [**76**], Wang et al. proposed an entropy-based scheme in OF switches; it focuses only on detection, but it cannot find the victim or the illegitimate hosts to, eventually, block them. In [**77**], Lim et al. proposed a DDoS mitigation scheme for botnet-based attacks that runs on SDN controller; it requires a large amount of communications between the control plane and data plane to protect the victim. Moreover, this scheme [**77**] not only makes the SDN controller vulnerable to DDoS attacks but also requires a high latency to cooperate with the SDN controller. In [**78**], K. Giotis et al. combined sFlow protocol with OF protocol in order to detect DDoS attacks reducing the communication overhead between data plane and control plane. This scheme [**78**] works well; however, it has high false positive rates.

To address the weaknesses of existing solutions [**72**–**78**], we propose an efficient and scalable intra-domain DDoS mitigation scheme to detect and mitigate DDoS attacks. In our scheme, we combine entropy calculation using sFlow with SDN functionalities to block illegitimate traffic. The proposed solution employs sFlow protocol to separate flow monitoring from the forwarding logic; this makes it much more scalable compared to existing native OF schemes [**72**–**78**]. And using machine learning scheme (*i.e.*, I-BS), our intra-domain DDoS mitigation scheme is much accurate in comparison with the ones using sampling technology [**78**].

3.3.3. Inter-Domain DDoS Mitigation Schemes

As DDoS attacks evolve rapidly and become more devastating, cooperation among several domains has become necessary to ensure sophisticated mitigation in order to cope with large scale DDoS attacks.

Several mitigation schemes have been proposed to effectively collaborate in order to deal with DDoS attacks. However, only a few of them have been considered for widespread deployment because of their implementation complexity and limited effectiveness. F. Guo et al. [**79**] proposed a mechanism that deploys filters at the border of the network to block incoming packets with source IP addresses that do not belong to the network. However, the effectiveness of this method depends on global deployment across the Internet. This method has "neighborhood policy" that requires every ISP to participate in order to provide the list of IP addresses that does not belong to its network. In [**80**], IETF (Internet Engineering Task Force) proposed the development of a new collaborative protocol called DOTS (DDoS Open Threat Signaling) in order to advertise DDoS attacks. The inter-domain collaboration scheme used in DOTS leverages the mitigation by sharing resources among organizations. DOTS protocol consists of customers (*i.e.*, DOTS clients) and controller (*i.e.*, DOTS server). When an attack is detected, the customer requests the mitigation service from the controller that is responsible for inter-domain communication and coordination. The implementation of the inter-domain DDoS mitigation can be either distributed or centralized. However, the effectiveness of DOTS depends on global deployment which may be disregarded due to implementation complexity. Moreover, the process of collaboration can be easily compromised. To counter this, digital certificates can be used; however, it is costly to setup and maintain certificates. In addition, if centralized implementation is used, it will cause single-point-of-failure.

In [**81**], Steinberger et al. proposed a similar scheme to DOTS [**80**]; it uses flow-based event exchange format to simplify the deployment and the collaboration between domains. This scheme [**81**] also requires a secure public-key infrastructure (PKI). PKI-based systems are costly to setup and maintain [**82**]. In [**83**], Giotis et al. proposed a collaborative DDoS mitigation scheme across multiple SDN based domains. They extend Border Gateway Protocol (BGP) protocol to repost incidents as URIs in BGP signals. However, any modification to BGP is a challenging endeavor. Moreover, the incident report latency may be large given that domains do not report in real-time. More importantly, this scheme [**83**] does not verify the authenticity of incident reports resulting in a scheme that is vulnerable to spoofed incident reports from illegitimate domains. In [**84**], Bahman et al. proposed CoFence, a DDoS defense mechanism that facilitates collaboration among network function virtualization (NFV) based domains. In CoFence, when a NFV-based domain is under attack, it redirects the traffic to other NFV-based domains to filter the packets. First, CoFence has

a privacy issue since it redirects the traffic to other NFV-based domains. Moreover, this process of redirection will also increase incident report latency. Many other schemes have been proposed (*e.g.*, [**62**, **85**]); however, the complexity of deployment and overhead, that is generated, remain challenging issues in these schemes. In [**86**], Rodrigues et al. proposed a scheme, which uses blockchain and smart contracts, to advertise blacklisted IP addresses. However, this scheme [**86**] requires a central entity to issue certificates of ownership of IP addresses, when calling (*i.e.*, storing data) smart contracts. Moreover, it is concerned only with inter-domain DDoS mitigation and not with intra-domain DDoS mitigation. In [**87**], Mathis et al. proposed an OpenFlow-based firewall to provide security to blockchain nodes. They implemented their solution as a module, in SDN controller, that uses SDN functionalities to filter network traffic; it provides access control functionality and protects blockchain nodes from DoS attacks. However, a very high packet rate from switches to SDN controller may overload the control plane.

To address the shortcomings of existing solutions [**62**, **79**–**81**, **83**–**87**], we propose an efficient, secure, low-cost, easy-to-deploy and decentralized inter-domain collaboration scheme; it allows multiple SDN based domains to securely collaborate and transfer attack information (*i.e.*, suspicious IP addresses) in a decentralized manner based on blockchain using smart contract. The use of new technologies (*e.g.*, SDN, blockchain and smart contract) introduces new opportunities for secure, efficient and flexible DDoS attacks collaboration across multiple SDN based domains. Indeed, with these technologies, one can avoid the complexity of developing new protocols and/or the modification of existing ones (*e.g.*, [**83**]); in addition, it removes the need to use a central entity in contrast to [**86**] and enforces permissions to participate in the collaboration.

3.4. Cochain-SC: An Overview

In this section, we present an overview of Cochain-SC. More specifically, we explain how Cochain-SC can combine two levels of mitigation, intra-domain and inter-domain DDoS mitigation, allowing for an efficient mitigation along the path of an ongoing attack and an effective mitigation near to the origin of attack. Inter-domain DDoS mitigation scheme assumes multiple SDN based domains (*e.g.*, ASs: A, B, C, D, E and F) to collaborate as shown in Fig. 3.3. SDN based domains communicate with each other via our proposed inter-domain collaboration scheme based on blockchain using smart contract. First, the organization (*i.e.*, owner of the smart contract) needs to create the collaboration contract. Then, it adds the authorized participants (*i.e.*, collaborators). Therefore, when attackers, which are distributed across multiple domains, generate an attack towards the victim (*e.g.*, hosted at AS C; see Fig. 3.3), our intra-domain DDoS mitigation scheme detects and mitigates the attack inside the domain; it also stores the suspicious IP address, denoted ip_address, in the smart

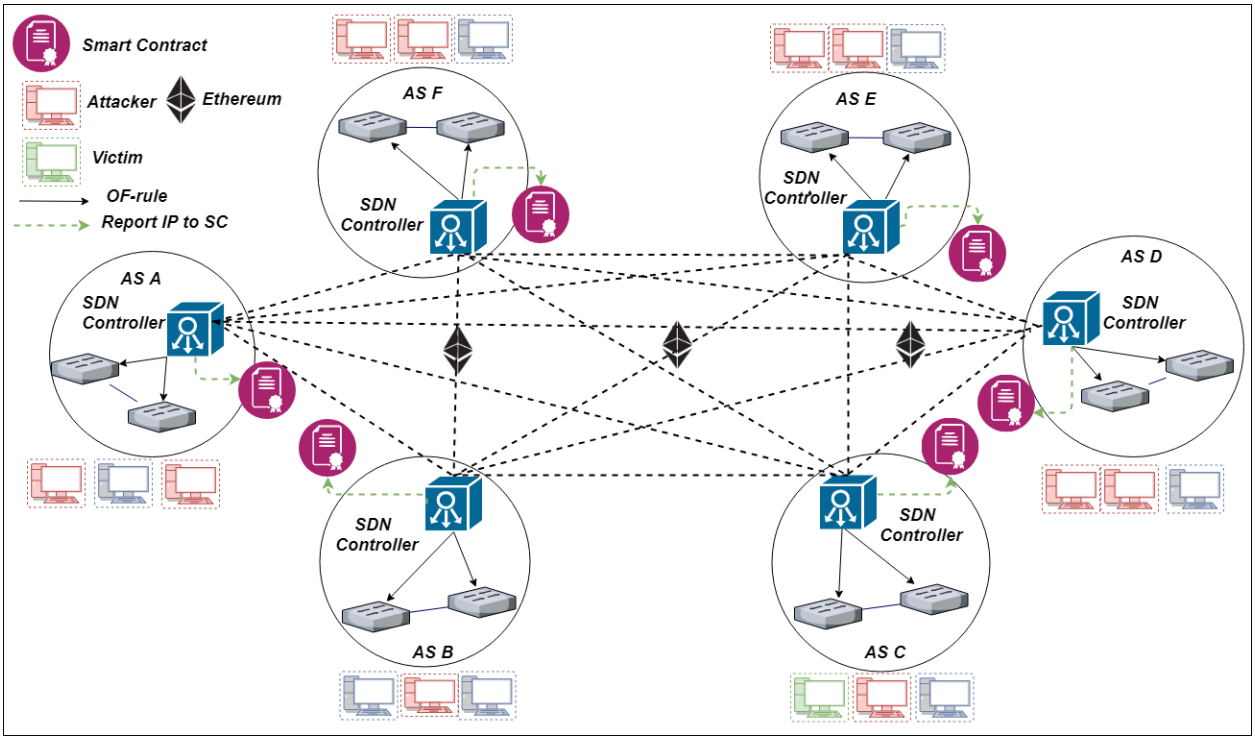


Figure. 3.3. Cochain-SC blockchain-based framework.

contract. When the next block is mined, each of the authorized participants of the collaborative scheme will have access to the list of suspicious IP addresses to be blocked; this will allow for an efficient mitigation along the path of an ongoing attack and an effective mitigation near to the origin of attack. To report/receive attack informations, each SDN based domain runs an Ethereum client (*e.g.*, geth client [**88**]). In the following, we describe Cochain-SC in more details. First, we describe our intra-domain DDoS mitigation scheme which is composed of 3 schemes (*i.e.*, I-ES, I-BS and I-DM). Then, we describe our inter-domain DDoS mitigation scheme.

3.5. Intra-Domain DDoS Mitigation Scheme

3.5.1. Design Overview

When designing the our intra-domain DDoS mitigation scheme, we did consider the following objectives: (a) the scheme should ensure a full protection and should be accurate in detecting attacks inside the domain; this is ensured via a novel detection scheme (*i.e.*, I-BS), based on an intra Entropy-based scheme (I-ES); (b) attacks should be effectively mitigated using our intra-Domain Mitigation (I-DM) scheme; and (c) the method should be scalable.

3.5.2. System Architecture

The architecture of intra-domain DDoS mitigation method consists of four main modules: (1) Intra Entropy-based Scheme (I-ES) to measure the randomness of data inside the domain using sFlow; (2) Intra Bayes-based Scheme (I-BS) to classify, using entropy values, illegitimate DDoS flows; (3) Intra-Domain Mitigation (I-DM) scheme to effectively mitigate illegitimate flows inside the domain; and (4) a blockchain layer, used in our inter-domain DDoS mitigation collaboration.

Intra-domain DDoS mitigation scheme has two main phases: (1) an intra-domain machine learning DDoS detection and mitigation module. This module has the objective to detect, in real-time, illegitimate flows; it consists of I-ES, I-BS and I-DM; and (2) a blockchain module, used in inter-domain DDoS mitigation. I-ES aims to measure the randomness of data inside the victim's domain (*e.g.*, AS C; see Fig. 3.3) using network traffic flow features. I-BS has the objective to detect, in real-time, illegitimate flows based on stateful network traffic features calculation (I-ES calculation). This module is running as an application on the top of the SDN controller (*i.e.*, application layer). Also, we use in our process of detection/mitigation, the REST API [**48**] to manage any SDN controller and block illegitimate traffic. I-DM aims to effectively mitigate illegitimate traffic inside the domain. OF was not designed to support QoS features; however, OF 1.3 introduces *m*eters [**54**] to the OF protocol. Each flow entry specifies *meter*; *meter* entries with different *Meter_id* are deployed to monitor the speed of the classified illegitimate flows by I-BS; if the flow rate exceeds *band* (rate limiter), I-DM drops suspected flows.

3.5.3. Machine Learning DDoS Detection and Mitigation Module

In this section; first, we describe the details of our information collection method based on flow packet sampling using sFlow; then, we describe I-ES, to measure the randomness of data inside the domain and extract network features; finally, we descibe I-BS, to detect illegitimate flows.

3.5.3.1. Flow Statistics Collection Scheme. There are two commonly methods for collecting information: the first method is based on OF protocol and the second method is based on flow monitoring. In this chapter, we choose to use flow monitoring methods based on sFlow protocol. In the following, we describe each of these methods and justify our choice. To detect DDoS attacks in SDN, most existing solutions [**72**–**78**] propose to collect and send, periodically, the features of the flows (*e.g.*, number of received packets and duration of matched flows) to SDN controller using OF protocol. Collection of features, using OF protocol, can be initiated when the SDN controller sends a feature request *(ofp_flow_stats_request)* to OF switches which respond by sending the flow table content *(ofp_flow_stats_reply)*. This

method can collect the overall traffic of flow information passing through the data plane. However, this method can overload the control plane and exhausts the bandwidth between OF controller and OF switches; furthermore, it may exhaust Ternary Content Addressable Memory (TCAM) in OF switches. Therefore, OF based method is not adequate for detecting high rate DDoS attacks. To address the shortcomings of the method described above, we decided to monitor flows using flow monitoring method based on sFlow protocol. This is more efficient, scalable and does not consume bandwidth between SDN controller and OF switches. sFlow performs flow aggregation that is required during DDoS attacks when the number of flow entries is very high. The sFlow collector (sFlow-RT [**47**]) receives periodically packet samples from each sFlow agent embedded in data plane (data plane devices) and updates the counters of each flow during the monitoring interval. Afterwards, periodically, I-ES calculates entropy values and I-BS detects automatically illegitimate flows.

3.5.3.2. Intra Entropy-based scheme (I-ES). The main idea behind I-ES comes from Shannon's information theory [**51**]. The entropy calculation measures the disorder/randomness of incoming data (*i.e.*, the incoming flow for a given time period). I-ES runs as an application on the top of the controller and uses sFlow protocol; it collects traffic information and computes the entropy of each flow. When a victim's domain (*e.g.*, AS C, see Fig. 3.3) is under DDoS attacks, the number of packets that have the same IP address destination, denoted IP_{dst} (*i.e.*, victim's IP address) increases resulting in a concentrated distribution of IP_{dst}; while the normal state of victim's network leads to a more dispersed probability distribution of IP_{dst}. High entropy values mean highly dispersed probability distribution of IP_{dst}, while low entropy values mean a concentration of IP_{dst}. Therefore, we use I-ES to measure the changes of traffic information inside the victim's domain during monitoring interval ΔT. In this work, we define a flow as a seven tuple: $\{MAC_{src}$, MAC_{dst}, IP_{src}, IP_{dst}, $Port_{src}$, $Port_{dst}$, $Proto\}$ Let $F_{i,j}$ denote flow f_i at local OF switch S_j; it is defined as follows:

$$F_{i,j}(IP_{dst_i},S_j) = \{< IP_{dst_i},S_j,t > | S_j \in S, i,j \in I, t \in R\} \tag{3.5.1}$$

where IP_{dst} is the destination IP address of f_i, t it the current timestamp, and $S = \{S_j, j \in I\}$ the set of OF switches.

Let $|F_{i,j}(IP_{dst_i},S_j,t)|$ be the count number of packets of flow $F_{i,j}$ at time t. The variation of the number of packets for flow f_i at local OF switch S_j during ΔT is defined as follows:

$$N_{F_{i,j}}(IP_{dst_i},S_j,t+\Delta T) = |F_{i,j}(IP_{dst_i},S_j,t+\Delta T)| - |F_{i,j}(IP_{dst_i},S_j,t)| \tag{3.5.2}$$

The probability $p_{i,j}$ of flow f_i over all flows at local OF switch S_j is expressed as follows:

$$p_{i,j}(IP_{dst_i},S_j) = \frac{N_{F_{i,j}}(IP_{dst_i},S_j,t+\Delta T)}{\sum_{i=1}^{N} N_{F_{i,j}}} \tag{3.5.3}$$

where $\sum_{i=1}^{N} p_{i,j}(IP_{dst_i},S_j) = 1$.

Let IP_{dst} be a random variable that represents the number of flows during the time interval ΔT. We define the entropy of flow f_i at local OF switch S_j as follows:

$$H(IP_{dst}) = -\sum_{i=1}^{N} p_{i,j}(IP_{dst_i},S_j)\log_2 p_{i,j}(IP_{dst_i},S_j) \tag{3.5.4}$$

In order to have a measurement metric that is independent from the number of distinct values, we divide the entropy values by the upper bound value that is $\log_2 N$. Therefore, the normalized entropy values are between [0, 1] and are defined as below:

$$H(IP_{dst})' = \frac{H(IP_{dst})}{\log_2 N} \tag{3.5.5}$$

The attribute (*e.g.*, IP_{src}, IP_{dst}) to aggregate flows depends on the attack (*e.g.*, DRDoS, DDoS) under investigation. When the network suffers from DDoS attack, the number of the flows that have the same destination IP address IP_{dst} (*i.e.*, target of attack) increases sharply leading to a significant decrease of entropy values; while the source IP addresses entropy values are relatively concentrated. Attackers generates illegitimate traffic from the same UDP/TCP port source number, as legitimate users generate legitimate traffic from random UDP/TCP port source number; therefore, this attribute also can better represent the DDoS attack characteristics. Consequently, we use $\{IP_{dst}$, $Port_{src}$ and $Port_{dst}$ $\}$ as the attribute to aggregate flows. Finally, we represent the network traffic features at the k^{th} time period as:

$$X_k = \{H(IP_{dst})'_k, H(Port_{src})'_k, H(Port_{dst})'_k\} \tag{3.5.6}$$

3.5.3.3. Intra Bayes-based scheme (I-BS). I-BS is a binary classifier that uses I-ES calculation to classify the k^{th} vector X_k as either legitimate or illegitimate. I-BS receives vector X_k and classifies it using the probability of illegitimacy. Once X_k is classified as illegitimate, I-BS notifies I-DM to deploy the mitigation action against this illegitimate vector X_k. In the following, we detail I-BS; first, we briefly describe the flow representation. Then, we discuss in more details the criterion classification of I-BS.

In I-BS, each sample is represented by a vector $x = (x_1, x_2, x_3)$ where x_1, x_2, x_3 are values taken, respectively, by random variables $H(IP_{dst})'$, $H(Port_{src})'$ and $H(Port_{dst})'$. Each of these random variables indicates, respectively, entropy values of destination IP address, UDP/TCP port source and UDP/TCP port destination.

3.5.3.4. Criterion of classification. I-BS considers two classes of vectors: (1) legitimate vectors denoted by leg and (2) illegitimate vectors denoted by $illeg$. The class of k^{th} vector X_k, denoted by c, can be either leg or $illeg$ and is represented as follows:

$$c = \underset{c\in\{leg,illeg\}}{\arg\max}\ p(c|X_k) \tag{3.5.7}$$

We have $p(leg|X_k) + p(illeg|X_k) = 1$; thus, the selection criterion is defined as follows:

$$X_k \text{ is illegitimate iff: } p(illeg|X_k) \geq 0.5 \tag{3.5.8}$$

Using Bayes theorem [**53**], the probability of vector X_k to belong to class c is defined as follows:

$$p(C = c|X = X_k) = \frac{p(C = c).p(X = X_k|C = c)}{p(X = X_k)} \tag{3.5.9}$$

According to the total probability theorem, we have:

$$p(C = c|X = X_k) = \frac{p(C = c).p(X = X_k|C = c)}{\sum_{c\in\{leg,illeg\}} p(C = c)p(X = X_k|C = c)} \tag{3.5.10}$$

Therefore, the selection criterion is equivalent to: X_k is illegitimate iff

$$p(C = c|X = X_k) = \frac{p(C = c).p(X = X_k|C = c)}{\sum_{c\in\{leg,illeg\}} p(C = c)p(X = X_k|C = c)} \geq 0.5 \tag{3.5.11}$$

$H(IP_{dst})'$, $H(Port_{src})'$ and $H(Port_{dst})'$ are conditionally independent variables given class c. Let $p_k(leg)$ and $p_k(illeg)$ denote the conditional probabilities that the k^{th} vector X_k is receptively legitimate and illegitimate.

Using Eq.(5.6.3), the selection criterion can be expressed as follows: X_k is illegitimate iff :

$$\frac{\prod_{k=1}^{n} p_k^{X_k}(illeg)(1 - p_k(illeg))^{1-X_k} p(illeg)}{\prod_{k=1}^{n} p_k^{X_k}(illeg)(1 - p_k(illeg))^{1-X_k} p(illeg) + \prod_{k=1}^{n} p_k^{X_k}(leg)(1 - p_k(leg))^{1-X_k} p(leg)} \geq 0.5 \tag{3.5.12}$$

When $p(leg) = p(illeg)$, the selection criterion becomes as follows: X_k is illegitimate iff:

$$\frac{\prod_{k=1}^{n} p_k^{X_k}(illeg)(1 - p_k(illeg))^{1-X_k}}{\prod_{k=1}^{n} p_k^{X_k}(illeg)(1 - p_k(illeg))^{1-X_k} + \prod_{k=1}^{n} p_k^{X_k}(leg)(1 - p_k(leg))^{1-X_k}} \geq 0.5 \tag{3.5.13}$$

I-BS is trained and then used to classify k^{th} vector X_k as either legitimate or illegitimate. By combining I-ES and I-BS, Cochain-SC can accurately detect the attack in real time with low false positive rates while maintaining a high detection rate.

3.5.4. Intra-Domain Mitigation (I-DM) Scheme

When I-BS detects DDoS attack, a mitigation action is performed to protect the victim. For this aim, new OF rules are installed, using the API of the SDN controller, into the OF switch under attack; these rules have a high priority to match suspicious packets and monitor their speed. I-DM has the purpose to effectively mitigate illegitimate traffic. A flow entry can specify a meter; meter entries with different Meter_id are deployed to monitor the speed of the classified illegitimate flows by I-BS; if the packet rate surpasses the band (rate limiter), then we drop suspected packets (see Fig. 3.4). In our simulations, we set an

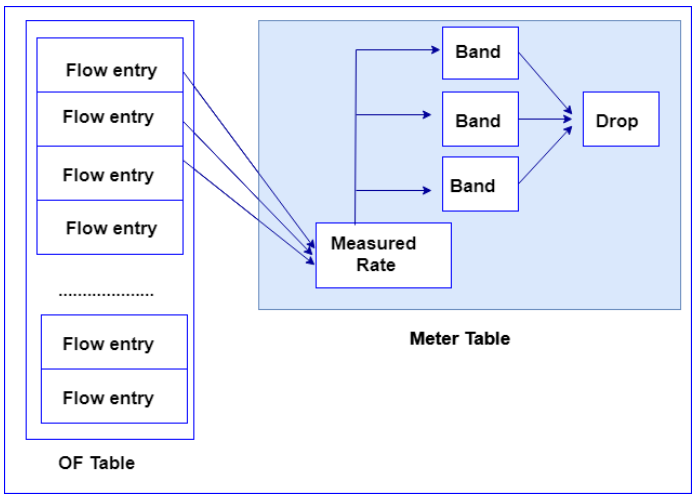


Figure. 3.4. Table Model in OF switches

adaptive band; at the beginning the band is fixed to 1000 packets per second, then it can be adjusted based on the capacity of OF table, traffic rate and workload of both data and control plane.

3.6. Inter-Domain DDoS Mitigation Scheme

3.6.1. Overview

Cochain-SC has the objective to ensure sophisticated mitigation across multiple domains and cope with large scale DDoS attacks. Inter-domain DDoS mitigation assumes multiple SDN based domains to collaborate. Fig. 3.5 shows a high-level architecture of our inter-domain DDoS mitigation scheme. ASs (SDN based domains) are classified into 3 types of network domains, source domain, intermediate network domains and destination domain. The source domain is the network (*i.e.*, AS) in which the attacker starts the attack. Intermediate network domains forward illegitimate traffic. The destination domain is the domain where the victim is hosted. The SDN controller of each AS can either report or retrieve the list of illegitimate IPs (see Fig. 3.5). We have leveraged SDN and blockchain to both mitigate the attack inside the domain and effectively collaborate to: (a) reduce the enormous cost of forwarding packets, across multiple domains, that compose amplified attack traffic; and (b) block the attack close to its source.

Ethereum blockchain is a decentralized platform for developing smart contracts. The language to write these contracts is Turing complete (*e.g.*, Solidity [**89**]); this allows users to create and run smart contracts (of any complexity) on the blockchain. From the computing prospective, bitcoin network provides distributed data storage for managing transactions

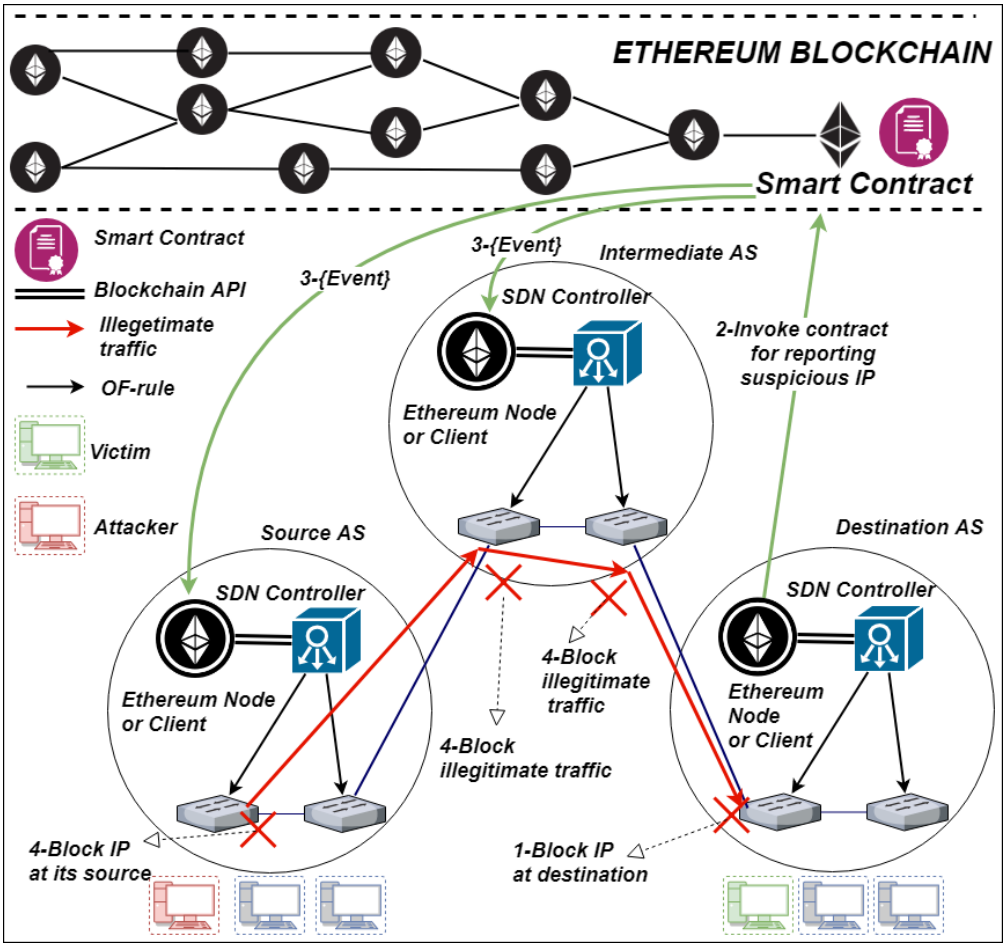


Figure. 3.5. High-level architecture of inter-domain DDoS mitigation scheme.

while the Ethereum network not only provides distributed data storage but also offers computing resources; each node of the Ethereum network executes the code (*e.g.*, smart contract) that is deployed by participants using Ethereum Virtual Machine (EVM). Fig. 3.5 shows the main steps executed by the proposed approach: (1) Once our intra-domain DDoS mitigation scheme detects the attack, using I-ES and I-BS, it mitigates the attack inside the domain using I-DM and blocks the illegitimate traffic; (2) The SDN controller of the domain under the attack (Destination AS), sends a transaction, if it is authorized, to the smart contract in order to report suspicious IP adresses; (3) once the transaction is confirmed (*i.e.*, the transaction is in the new block appended to the blockchain), an event is emmitted by the contract and is received by the registered/authorized collaborators of the smart contract (*e.g.*, SDN controller of source and intermediate network domains); and (4) upon receipt of the event, the collaborators block the illegitimate traffic.

3.6.2. Cochain-sc's Smart Contract

3.6.2.1. Application scenario. We consider an organization (*i.e.*, contract owner) that would like to manage a collaboration process between different ASs around the world. First, it creates Cochain-SC's smart contract and deploys it on the Ethereum blockchain. The use of blockchain in the collaboration process allows for transparency while maintaining "pseudonymity". Then, the organization adds, via the smart contract, the collaborators into the system. It includes the collaborator's address and some other information (*e.g.*, collaborator notes). When a collaborator (*e.g.*, victim's SDN controller) detects and mitigates an attack, it sends a transaction to the smart contract to add the suspicious IP address. The smart contract allows (1) the organization (owner of the contract) to add collaborators to the contract; (2) the organization to manage and modify the collaboration process in a transparent manner; (3) the organization to delete collaborators from the collaboration process if needed; (4) collaborators to report suspicious IP address in a secure and efficient manner; and (5) collaborators to delete the reported IP address from the contract if needed. In the following, we present the design of our smart contact.

3.6.2.2. Cochain-sc's design. The smart contract is programmed using solidity language. In our smart contract, we use the following global variables: now: the time passed in seconds since 1970; msg.sender: the msg object represents the transaction that is sent; and the msg.sender is the address of the user that sent the transaction to the smart contract. In Ethereum, there are two types of accounts: (1) Externally Owned Account (EOA): it has public and private keys; it can send transactions to transfer ether or to smart contracts and (2) contract account: it runs code and has no public/private key. The smart contract, called Collaboration contract, is deployed by the organization. In the following, we first describe the Collaboration Contract Initialization; then, we provide the functions of the Collaboration Contract.

Collaboration Contract Initialization: This process defines the state variables of the contract.

[1] The Collaboration Contract Owner of address types, which defines the address of organization responsible of the collaboration process.
[2] The status of the Collaboration Contract: active or inactive. The contract owner is responsible for activating or deactivating the contract.
[3] The name of the contract owner.
[4] numberOfcollaborators: defines the number of collaborators.
[5] numberOfrecords: defines the number of records (*i.e.*, ip_addresses).
[6] collaboratorsAdr: stores the addresses of the collaborators. The aim of this array is to reduce the cost of finding and removing a specific colloborator from the collaborators mapping (see variable 8).

[7] recordsAdr: stores the records of suspicious ip_address. The aim of recordsAdr is to reduce the cost of finding and removing a specific record from the records mapping (see variable 9).

[8] collaborators: defines a mapping collection from the address of the collaborator to a corresponding Collaborator struct.

[9] Records: defines a mapping collection from the ip_address to a corresponding record struct.

[10] OnlyOwner of modifier type. We applied this modifier to the function that (adds/removes) the collaborators (to/from) the smart contract and the function that either activates or deactivates the smart contract; thus, only the owner of the contract can invoke these functions (adds/removes) the colloborators or (activates/deactivates) the contract.

[11] OnlyCollaborators of modifier type. It takes as input the address of the caller and checks if the caller is authorized to exectue the function that the modifier is applied to it. We applied this modifier to the function that (adds/removes) the records (to/from) the smart contract; thus, only the collaborators (owner included) of the contract can invoke (adds/removes) the records.

The Collaboration Contract mainly provides the following functions where c denotes an instance of Collaborator and r an instance of record:

isCollaborator(c.EOA): This function takes as input the Externally Owned Account (c.EOA) of a collaborator and returns true if the collaborator exists in the collaboration contract; otherwise, it returns false. This function will be invoked at the moment when the owner tries to add/ remove the collaborator to/from the collaboration contract. Algorithm 3 illustrates the logic of this function.

Algorithm 3: isCollaborator

Input : c.EOA
Output: bool

```
1 if collaboratorsAdr.length == 0 then
2 |  return false;
3 else
4 |  if collaboratorsAdr[collaborators[c.EOA].index] == c.EOA then
5 |  |  return true;
6 |  else
7 |  |  return false;
8 |  end
9 end
```

isRecord(r.IP): This function takes as input ip_address of a record and returns true if the ip_address exists in the collaboration contract; otherwise, it returns false. This function will be invoked at the moment when one of the collaborators tries to add/remove the record to/from the collaboration contract. Algorithm 4 illustrates the logic of this function.

addCollaborator(c.EOA, c.Infos): This function can only be invoked by the owner of the smart contract to add collaborators; it takes as input the Externally Owned Account (c.EOA) of the collaborator and the information about the collaborator (c.Infos) and adds the collaborator to smart contract (*i.e.*, to collaboratorsAdr array (see variable 6 in the Initialization part)) as well as the timestamp of when the collaborator was added. This happens if the contract is activated and the colloborator' identity is authenticated. Algorithm 5 illustrates the logic of this function.

Algorithm 4: isRecord

```
   Input  : r.IP
   Output: bool
1  if recordsAdr.length == 0 then
2  |  return false;
3  else
4  |  if recordsAdr[records[r.IP].index] == r.IP then
5  |  |  return true;
6  |  else
7  |  |  return false;
8  |  end
9  end
```

removeCollaborator(c.EOA): This function can only be invoked by the owner of the smart contract to remove collaborators; it takes as input the Externally Owned Account (c.EOA) of the collaborator and removes the collaborator from the smart contract. Algorithm 6 illustrates the logic of this function.

addRecord(r.IP): This function can only be invoked by either the owner of the smart contract or the collaborator that has already been added in the smart contract to report suspicious ip_address. It takes as input the suspicious ip_address and adds the record to the smart contract (*i.e.*, to recordsAdr array (see variable 7 in the Initialization part)). Algorithm 7 illustrates the logic of this function.

removeRecord(r.IP): This function can only be invoked by either the owner of the smart contract or the collaborator to remove records; it takes as input an IP address and removes the corresponding record, if it exists, from the smart contract. Algorithm 8 illustrates the logic of this function.

ChangeStatus(bool status): This function can only be invoked by the owner of the smart

Algorithm 5: addCollaborator

```
   Input  : c.EOA, c.Infos
   Output: null
 1 if msg.sender is not owner then
 2 |  throw;
 3 end
 4 if status is not true then
 5 |  throw;
 6 end
 7 if isCollaborator(c.EOA) == true then
 8 |  throw;
 9 else
10 |   length ←collaboratorsAdr.push(c.EOA)
   |   collaborators[c.EOA]←Collaborator(c.EOA,c.Infos, now, length-1)
   |   emit CollaboratorAdded(c.EOA, c.Infos)
   |   numberOfCollaborators++
   |   end
11 |
```

Algorithm 6: removeCollaborator

```
   Input  : c.EOA
   Output: null
 1 if msg.sender is not owner then
 2 |  throw;
 3 end
 4 if status is not true then
 5 |  throw;
 6 end
 7 if isCollaborator(c.EOA) == false then
 8 |  throw;
 9 else
10 |   rowToDelete ←collaborators[ c.EOA ].index
   |   keyToMove←collaboratorsAdr[length-1]
   |   collaboratorsAdr[rowToDelete]=keyToMove
   |   collaborators[keyToMove].index=rowToDelete
   |   collaboratorsAdr.length --
   |   emit CollaboratorRemoved(c.EOA)
   |   numberOfCollaborators --
   |
11 end
```

contract to either activate or deactivate the smart contract. Algorithm 9 illustrates the logic of this function.

Algorithm 7: addRecord

```
Input  : r.IP
Output: null
1  if msg.sender is not Collaborator then
2  |  throw;
3  end
4  if status is not true then
5  |  throw;
6  end
7  if isRecord(r.IP) == true then
8  |  throw;
9  else
10 |   length ←recordsAdr.push(r.IP)
   |   records[r.IP]←Record(r.IP, msg.sender, now, length-1)
   |   emit RecordAdded(r.IP, msg.sender)
   |   numberOfRecords++
   |
11 end
```

Algorithm 8: removeRecord

```
Input  : r.IP
Output: null
1  if msg.sender is not Collaborator then
2  |  throw;
3  end
4  if status is not true then
5  |  throw;
6  end
7  if isRecord(r.IP) == false then
8  |  throw;
9  end
10 if records[r.IP].submitter is not equal msg.sender then
11 |  throw;
12 else
13 |   rowToDelete ←records[ r.IP ].index
   |   keyToMove←recordsAdr[length-1]
   |   recordsAdr[rowToDelete]=keyToMove
   |   records[keyToMove].index=rowToDelete
   |   recordsAdr.length--
   |   emit RecordRemoved(r.IP, msg.sender)
   |   numberOfRecords--
   |
14 end
```

Algorithm 9: ChangeStatus

```
Input  : status
Output: null
if msg.sender is not owner then
    throw;
end
if status is true then
    status ←false
     emit StatusChanged("Smart Contract Deactivated")
end
if status is false then
    status ←true
     emit StatusChanged("Smart Contract activated")
end
```

3.7. Implementation

In this section, we present the evaluation of our intra-domain DDoS mitigation scheme. Then, we describe the process of the implementation and deployment of the Collaboration Contract.

3.7.1. Experimentation validation of intra-domain DDoS Mitigation scheme

In the following, we describe the experimental environment of our intra-domain DDoS mitigation scheme. We implemented I-ES and I-BS as applications on the top of the SDN controller. Using sFlow protocol, I-ES extracts network traffic information and I-BS detects automatically illegitimate traffic inside the domain. To emulate a real network environment, we use mininet [**32**], a popular SDN emulation tool. Mininet uses Linux containers and virtual OF-switches (*e.g.*, OpenVswitch [**31**]) to allow realistic virtual networks of hosts and switches to be constructed using a virtual machine. Mininet is installed on a VirtualBox [**90**] VM; VM is connected to the Internet through Network Address Translation (NAT) (for software installation and updates), and a host-only adapter is configured on VM to enable it to communicate with the host-system. Additionally, Secure Shell (SSH) is used to allow access to the VM for running different software at the same time. In our testing environment, the network monitor (*i.e.*, sFlow-RT) and the SDN controller (*i.e.*, Floodlight [**50**]) are installed on the host-system and run Ethereum geth client (1.8.20-stable client). We run our experiments on a PC with CPU Intel Core i7-8750H-2.2 GHz and 16GB RAM.

Fig. 3.6 shows that our intra-domain DDoS mitigation scheme is implemented in the victim network. In our experiment, we conduct a set of DDoS attacks towards a victim

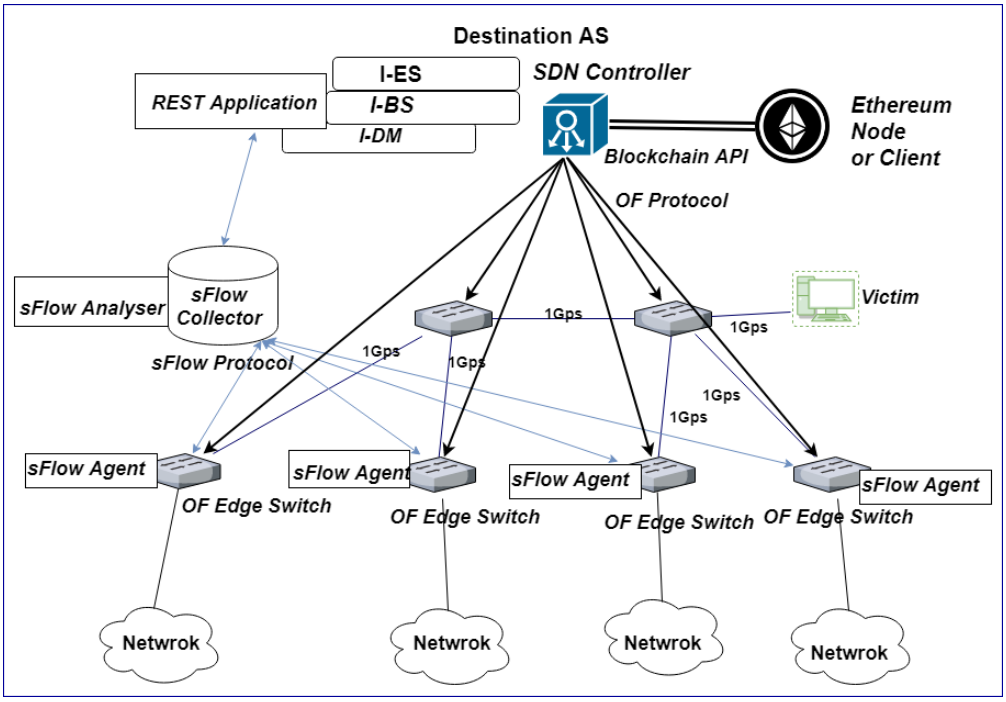


Figure. 3.6. Experimental Environment

Tableau 3.1. Transaction details of Cochain-SC

Details of Cochain-SC Creation Transaction in Ropsten test network	
TxHash	0xf68f79e7cb678645d3b5837bda8b33e4a3d9ebcfd78709a3df255c0a23059b25
Block Height	4738376 (182 Block Confirmations)
Timestamp	(Jan-01-2019 08:13:32 AM +UTC)
From	0xa70836a9a115f774cb848134d0f8b2473e27d181
To	0xcd0df692d251b0a82e63e7fafda2c72aa6b0a8f9
Gas Used by Tx	2225130

hosted in this domain. When I-BS detects the attack: (a) I-DM protects the victim inside the domain and blocks illegitimate traffic using OF protocol; and (b) the SDN controller invokes the smart contract for reporting suspicious IP. Fig. 3.6 shows the 4 components of the testbed: (a) OF controller (*i.e.*, Floodlight): it provides elementary connectivity which can be canceled using the Static Flow Pusher API; (b) sFlow network monitor (*i.e.*, sFlow-RT): it performs monitoring of 7500 switch ports in data center networks; (c) REST application: it executes I-ES and I-BS; and (d) 6 OF switches, the bandwidth of each link is set to 1 Gbps.

Each OF network contains more than 20 hosts; multiple hosts are simulated to launch the attack towards the victim hosted inside the domain. The rate of the attack varies from 100 to 500 Mbps; the objective is to test the scalability of the proposed solution. The sampling rate used in sFlow is 1/64. For the attack script, Scapy's Python library [**55**] is used to

generate DDoS flooding attack traffic from zombie hosts towards the victim as well as the simulated legitimate traffic. We use Hping3 [**91**] command line to Simulate DDoS attacks (*i.e.*, UDP/ICMP DDoS attacks).

3.7.2. Deployment of the Collaboration Contract

Once the smart contract is deployed, it can be self-executed without any human intervention. The deployment process is elaborated using truffle framework [**92**], a decentralized application development framework. First, we code the collaboration contract using the high-level language programming solidity. Then, we compile the contract into EVM byte code; once the contract gets compiled, it generates the EVM byte code and Application Binary Interface (ABI). Afterwards, we deploy the smart contract to the blockchain. Initially, we have deployed the smart contract on a private blockchain using Ganache [**93**], an Ethereum simulator used for testing the smart contract in a fast way. Then, we have deployed the smart contract on Ethereum official test network Ropsten. Fig 3.7 shows the contract lifecycle. Once deployed, the contract can be invoked using ABI definition and the address of the contract. If needed, the contract can be deleted (cannot be invoked anymore). We tested the implementation of Cochain-SC using both private (Ganache simulator) and public blockchain (Ethereum official test network Ropsten). Table 3.1 shows Cochain-SC creation transaction in Ropsten official test network. The details of a given transaction can be found using Ropsten Etherscan [**94**].

3.8. Cochain-SC: Experimental Results and Characteristics

3.8.1. Experimental Results

To test I-ES, we simulate the attack within an interval of 250 seconds. We launch the attack during the interval [150, 200]; Fig. 3.8(a) shows that the normalized entropy decreases rapidly at the start of the interval. The time taken by Cochain-SC operations is smaller than 13 seconds (see Fig. 3.8(b)).

3.8.2. Performance Evaluation

To measure the performance of I-BS, we define the detection rate (DR) and false positive rate (FPR) as follows:

$$DR = \frac{TP}{TP + FN}, FPR = \frac{FP}{TN + FP}$$

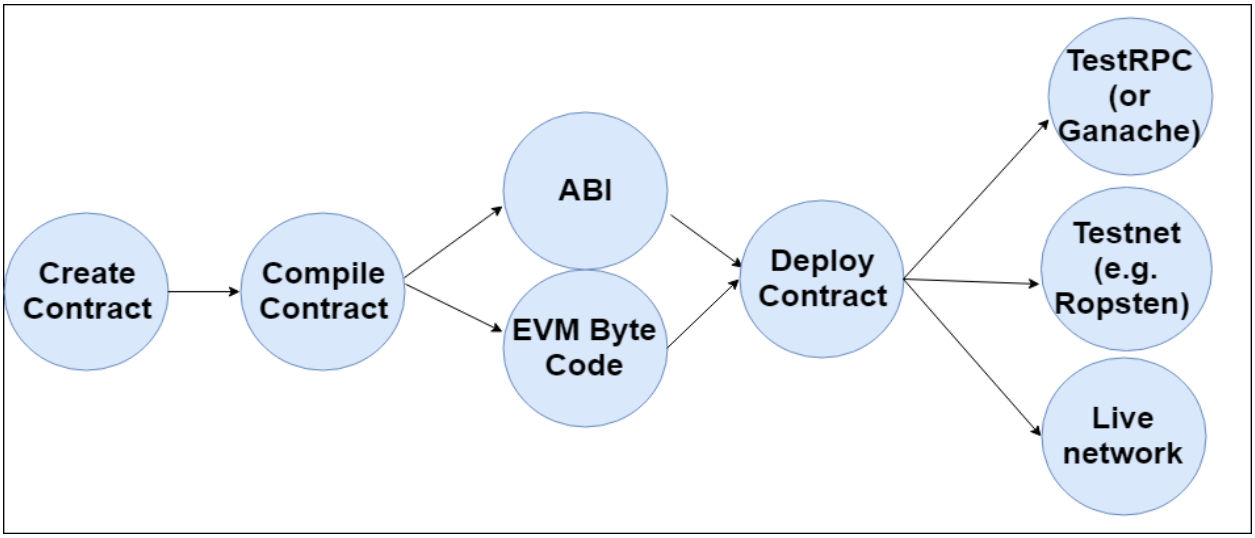
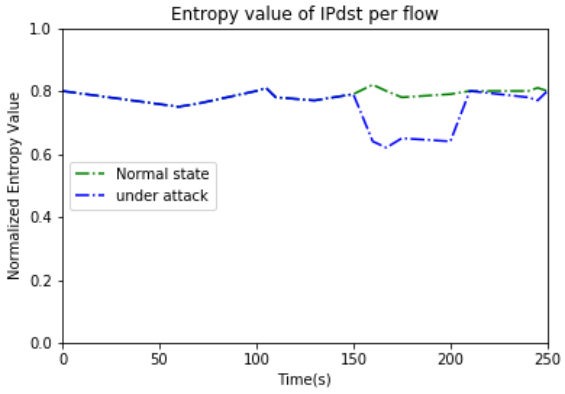
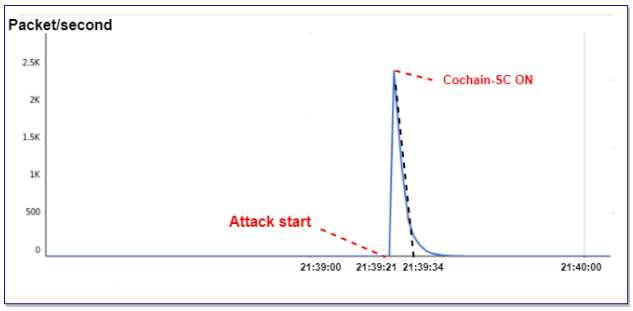


Figure. 3.7. The contract lifecycle.



((a))



((b))

Figure. 3.8. Performance evaluation of Cochain-Sc in terms of: (a) Normalized entropy value of IP_{dst} per flow; and (b) attack mitigation

where, TP (True Positives) represent the illegitimate flows that are correctly identified as illegitimate, FN (False Negatives) represent the illegitimate flows that are classified as legitimate, FP (False Positives) represent the legitimate flows that are identified as illegitimate,

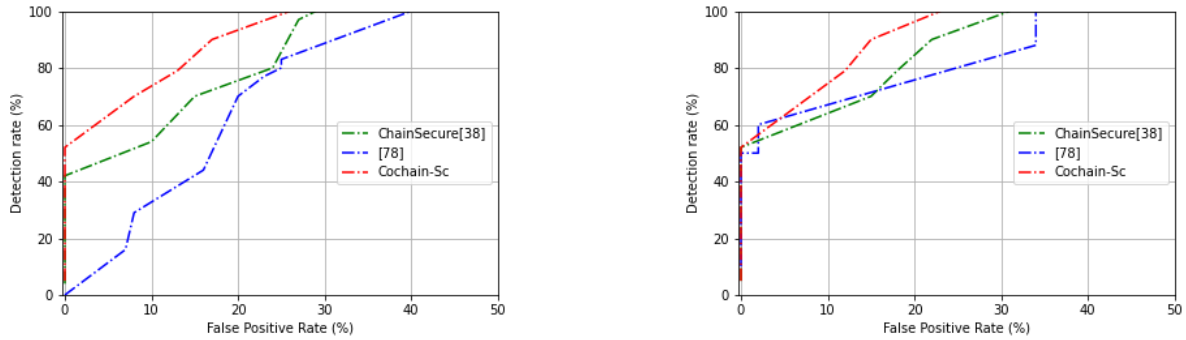


Figure. 3.9. ROC curves: (a) the 100 Mbps case; and (b) the 500 Mbps case.

and TN (True Negatives) represent the legitimate flows that are classified as legitimate. The performance of I-BS is evaluated using ROC curve. Moreover, we conducted two experiments (*i.e.*, 100Mbps and 500Mbps) and we compared I-BS in terms of accuracy and FPR with ChainSecure [**38**] and one of the most prominent related schemes [**78**]. Fig. 3.9(a) shows that Cochain-Sc achieves around 100% detection rate for 100 Mbps while it has just 26 % of FPR, while [**38**] and [**78**] achieve the same detection rate but with respectively 31% and 40% of FPR. Fig. 3.9(b) shows that Cochain-Sc achieves around 100% detection rate for 500 Mbps while it has just 23% of FPR, while [**38**] and [**78**] achieves the same detection rate but with respectively 30% and 34% of FPR.

3.8.3. Characteristics

The main objective of Cochain-SC is to provide a secure, easy-to-deploy, low-cost, efficient and flexible DDoS attacks mitigation scheme based on Ethereum using smart contract. In this section, we answer the question: how does Cochain-SC provide these features? Note that efficiency of cochain-SC has been shown in the experiments presented above.

3.8.3.1. Flexibility/Easy_to_deploy. Cochain-SC provides two levels of flexibility: (1) Cochain-SC provides the organization with the flexibility to easily add/remove collaborators to/from the system using addCollaborator()/removeCollaborator() functions. Similarly, collaborators can easily add/remove records to/from the system using addRecord()/removeRecord() functions; (2) Cochain-SC provides the organization (contract owner) with the flexibility to easily join or leave the system. To join the system, the organization needs to deploy the Collaboration contract. To leave the system, the organization can easily deactivate the contract using ChangeStatus() function. All these updates can be verified by anyone in the network (*i.e.*, Ethereum).

Tableau 3.2. Cochain-SC creation and functions costs

Function	**Gas Used**	**Actual Cost(ether)**	**USD**
create Cochain-Sc	2225130	0.00222513	0.296
addCollaborator()	140000	0.00014	0.018
removeCollaborator()	38450	0.00003845	0.005
addRecord()	102669	0.000102669	0.013
removeRecord()	39367	0.000039367	0.005
changeStatus()	28929	0.000028929	0.003

3.8.3.2. Security/Eligibility. Only authorized collaborators that have permissions can report suspicious IP address. This is achieved by Cochain-SC using modifiers. For example, the modifier "OnlyOwner" allows only the owner of the contract to execute the addCollaborator(), removeCollaborator() and changeStatus() functions. If a malicious/compromised user tries to execute these functions in order to either add illegitimtae collaborators to report fake IP or remove legitimate collaboartors, the execution will fail and no action will be recorded on the blockchain. The same restriction rule applies for the "OnlyCollaborators" modifier for the execution of addRecord() and removeRecord() functions; only collaborators (and also the contract owner) can add/remove the records.

3.8.3.3. Low Cost. In this section, we estimate the cost of the creation of the collaboration contract as well as the execution of each function used in Cochain-SC. When we conducted the experiment, the gasPrice was set to $1Gwei$, where $1Gwei = 10^9\ wei = 10^{-9}ether$, and 1 ether was equal to 133.42 USD. Table 3.2 shows the cost of the execution of different functions in Cochain-SC. We observe that the highest cost corresponds to the creation of Cochain-SC at 0.296 USD. However, it is performed only once to setup the collaboration system. All functions, provided by the smart contract, have low costs. Thus, Cochain-SC is cost effective compared to exiting related schemes.

3.8.3.4. Analysis. First, Cochain-SC preserves pseudonymity and does not allow traceability of identities of collaborators (*e.g.*, IP address of the collaborator). It is important to preserve the pseudonymity of collaborators; otherwise, they may be a target of DDoS attacks. Moreover, Cochain-SC does not suffer from single point of failure problem since it runs on Ethereum. Furthermore, it is decentralized scheme; thus, there is no need to a centralized authority (or a third party) to maintain the collaboration system; the reliability and availability of the records, recorded on the blockchain, are guaranteed. [**86**] is the only scheme that uses blockchain to advertise blacklisted IP addresses. However, this scheme [**86**] requires a central entity to issue certificates of ownership of IP addresses. Moreover, this scheme [**86**] is concerned only with inter-domain DDoS mitigation and not with intra-domain DDoS mitigation. However, Cochain-SC combines two levels of mitigation, intra-domain and inter-domain DDoS mitigation.

3.9. Conclusion

In this chapter, we proposed a blockchain-based framework called Cochain-SC which combines two levels of mitigation, intra-domain and inter-domain DDoS mitigation. For intra-domain, we combined I-ES with I-BS in order to detect in real-time illegitimate flows, and I-DM to effectively mitigate illegitimate flows inside the domain. For inter-domain, we proposed a smart contract-based framework that makes use of Ethereum's smart contract technology to facilitate the collaboration among SDN-based domain peers. The collaboration contract has been tested/evaluated and deployed on Ethereum official test network Ropsten; the appendix shows Cochain-SC address in Ropsten. Note that other blockchains, such EOS [**95**], can be used to implement our scheme. We decided to use Ethereum because it is the most popular blockchain, that supports the concept of smart contract, with most devout developers and is the second largest in terms of market value.

3.10. Appendices

The collaboration contract was deployed on the Ropsten Testnet of Ethereum with the following address: Organization Owner of account address: 0xa70836a9a115f774cb848134d0f8b2473e27d181
Cochain-SC address:0xCd0Df692D251B0a82E63e7FaFdA2c72aa6B0A8f9
Using this address, the transactions can be seen at: https://ropsten.etherscan.io/

Chapitre 4

SDNBoost: An SDN-based Boosting Ensemble Learning Framework for Advanced Network Attack Mitigation

4.1. Abstract

Advanced threats continue to grow at a rapid rate plaguing individuals and Internet Service Providers (ISPs) in a stealthy way. Intrusion detection systems (IDSs) have become vital to ensure network security. However, traditional IDSs based on pattern-matching/signature-filtering techniques are facing obstacles, due to lack of efficiency, making it difficult to protect against zero-day attacks. To alleviate this issue, several Machine/Deep learning (ML/DL)-based IDSs have been proposed to enhance the detection performance of traditional IDSs. However, they suffer from high bias and/or high variance. Ensemble learning (EL) is considered as a recent development in ML that aims to combine multiple ML models in order to build more flexible (*i.e.*, less bias) and less data-sensitive (*i.e.*, less variance) ML models. In this paper, we propose a scalable, efficient, and lightweight adaptive boosting EL framework, called SDNBoost, to effectively detect and mitigate network security threats in the context of software defined networks (SDN). SDNBoost consists of three modules. The first module is a novel data collection and optimized feature selection module that aims to reduce computational complexity while effectively improving detection performance. This module consists of (a) A Network Data flow Collection scheme (NDC) to gather the features of network data flow in a scalable and efficient way using sFlow protocol; (b) An optimized feature selection scheme that investigates using both of linear (*i.e.*, Pearson Correlation-based Feature Selection (PCFS)) and non-linear measures (*i.e.*, Tree-based Information gain Feature Selection (TIGFS) and Gradient Boosting Feature Selection (GBFS)); the objective is to select the most informative features to reduce training and testing time complexity; and (c) A hyperparameter optimization scheme to enhance

the overall EL model's performance. The second module is an ensemble adaptive voting module that uses three boosting EL techniques, namely Adaptive Boosting (AdaBoost), Gradient Tree Based Boosting (GTBM), and eXtreme Gradient Boosting (XGBoost) to effectively detect network security threats. The thirs module is an attack mitigation module (AMM) to effectively mitigate network security threats. We conduct extensive experiments to evaluate SDNBoost using different real-world network security threats; the experimental results, using two well-known public network security datasets, namely NSL-KDD and UNSW-NB15, show that SDNBoost outperforms state-of-the-art contributions by 12% higher accuracy and F1 score while significantly reducing computational complexity.

Keywords: IDS; Ensemble learning; gradient boosting, optimized feature selection; SDN.

Status: This journal paper is Submitted. Z. Abou El Houda, A. S. Hafid and L. Khoukhi, **"SDNBoost: An SDN-based Boosting Ensemble Learning Framework for Advanced Network Attack Mitigation,"** IEEE Transactions on Cognitive Communications and Networking, 2021.

4.2. Introduction

Due to the recent technological developments and rapid digital adoption, Internet has become an essential element to individuals and organizations to build online businesses such as banking, online education, and electronic commerce. This era of digitization is improving human's ability to work and perform activities (*e.g.*, Transaction Banking (TB), healthcare assistance, and online education) at any time and from anywhere. The increased dependency on the Internet is accompanied by numerous risks related to security and privacy issues *e.g.*, theft of sensitive data and service disruption of some critical industry segments (*e.g.*, finance and healthcare). These major network security threats have been increasing in sophistication and strength and are predicted to cost huge financial losses of about $20 Billion (USD) by 2021 [**96**], for both educational and business organizations. Moreover, the emergence of Internet of Things (IoT) botnets such as Mirai, as well as the rapid growth in the number of insecure IoT devices, with an estimation of 75 billion connected devices by the end of 2025 [**9**], can provide attackers with more sophisticated tools (*e.g.*, botnet as-a-service) to conduct large scale attacks. Therefore, cyber-security mechanisms need to be carefully designed to cope with the new emerging network security threats ranging from phishing attacks to ransomware attacks to data leakage. To this end, ISPs deploy different security mechanisms such as encryption techniques, user authentication, access control mechanisms in addition to stateful/stateless firewalls to protect networks and end-users. However, it has been shown

that these techniques are not sufficient to protect against the new emerging security threats (*i.e.*, zero-day attacks).

Thus, there is a need for highly effective intrusion detection systems (IDSs) to secure networks from all kind of unauthorized activities that may cause damage to data confidentiality, data integrity, or data availability. IDSs play a key role in ensuring security of the network and are considered as the first line of defense. IDSs aim to effectively and timely detect network security threats with less false positive rates. In general, IDSs can be divided into two categories: (1) Signature Filtering-based Intrusion Detection Systems (Misuse-Based detection systems, SFIDSs) that try to detect network intrusions by using predefined attack patterns/signatures of well-known attacks; and (2) Anomaly-based Intrusion Detection Systems (A-IDSs) that try to learn normal behaviors and consider any deviation from established normal behaviors as intrusions. SFIDSs are vulnerable to zero-day attacks (*i.e.*, unforeseen/unusual attacks) due to the lack of new/up-to-date attack patterns, while A-IDSs suffer from high false positive rates. In addition, a significant effort is required to continuously improve and evolve such IDSs.

The recent emergence of ML/DL techniques have revolutionized many areas and have achieved promising performances in many fields such as speech recognition and computer vision. IDSs also adopted these techniques to timely and effectively detect network security threats. With this adoption, ML/DL-based IDSs can effectively detect existing and new network security threats without continuously evolving/updating the patterns/signatures of traditional IDSs. However, most of state-of-the-art ML/DL-based IDSs use a single learning technique, such as neural networks and support vector machines, to classify/recognize network security threats. Despite achieving high detection performance in training data, these techniques fail to generalize and perform poorly on new/unseen data (*i.e.*, test data). Also, they suffer from a high bias and/or a high variance leading, in most cases, to over-fitting and/or under-fitting.

Recently, Ensemble learning (EL) has emerged as a new sub-field of ML that aims to improve the generalization ability of a single learner by combining multiple ML models, referred to as base learners. In general, EL can be divided into two categories. The first category concerns homogeneous EL that uses the same type of base learners; it includes (a) bagging: A parallel EL method in which base learners are used in parallel (*e.g.*, Random Forest); and (b) boosting: A sequential EL method in which the base learners are used in sequence (*e.g.*, Adaboost). The second category concerns heterogeneous EL that uses different types of base learners, called stacked generalization.

In 1979, Dasarathy and Sheela [**97**] proposed one of the earliest contributions on EL systems; their proposed scheme enhances the detection performance of a single learner through the use of a composite of classifiers (*i.e.*, Nearest neighbor (NN) and Linear Model). In

1990, Hansen and Salamon [**98**] proposed a cross-validation method that improves neural networks generalization performance through the use of ensembles of similar networks. Schapire [**99**] reported that a strong classifier can be generated by combining multiple weak learners through the use of boosting techniques. In 1992, Wolpert [**100**] proposed one of the earliest architectures that uses stacking techniques to minimize the generalization error of single learners. In 1993, Drucker and Schapire [**101**] proposed a boosting algorithm that uses a Probably Approximately Correct (PAC) learning model to construct a strong learner. They used neural networks (NNs) as a base learner to construct an ensemble of NNs that significantly improves detection performance in optical character recognition (OCR) domain. In 1996, Breiman [**102**] proposed bagging, referred to as bootstrap aggregation, that uses bootstrap sampling techniques with voting aggregations to reduce the variance of single learners. In bagging, first a subset of training data is randomly selected with replacement from the entire dataset. Then, this subset is used to train a particular classifier. Finally, different classifiers are combined using a majority voting decision.

Since then, research in EL systems has been expanded and they have been widely applied in different fields. Also, it has been shown that EL techniques outperform single based learners in handling more complicated tasks [**103**–**107**]. The use of multiple learning models will certainly reduce the risk of a poor generalization that a single learner can have. Also, in some applications, the amount of generated data is too large to be handled by a single learner, and the use of only a single model in such applications can lead to over-fitting. Moreover, it has been shown that EL techniques are effective in the opposite problem *i.e.*, having little data [**108**]. Re-sampling techniques can be used to select a random overlapping subset of the dataset, and use each subset to train a particular model. Then, one has to combine the results of each model, using a voting scheme, to produce a strong learner. Thus, EL becomes an ideal choice to build robust ML models that can effectively reduce variance and bias while improving detection/prediction performance. EL has the potential to greatly improve the efficiency of ML/DL- based IDSs to cope with the new emerging network security threats.

Boosting is considered as one of the most significant development of EL. Boosting techniques aim to transform weak learners, learners that slightly perform better than a random guessing, into a strong learner. In boosting, a sequence of weak learners are trained using a weighted version of the dataset in such a way that a high weight is given to previously misclassified samples. Thus, the model can give more attention to these misclassified data samples in the next round of training. Once the training is done, the output of each learner is combined through a weighted majority voting to construct a final strong learner. It has been shown that boosting techniques are robust against under-fitting issues, when only too little data is available and over-fitting issues, when too large data is available [**108**].

Software defined networks (SDN) is a novel paradigm that leverages network programmability to solve the limitations of conventional networks. SDN provides new capabilities,

through a logically centralized component, to cope with the new emerging security threats ranging from DDoS attacks to poisoning attacks to data leakage.

In this context, we propose SDNBoost, a novel multi-module optimized EL framework that combines optimized feature selection schemes with advanced EL techniques (*i.e.*, Boosting) to reduce computational complexity while effectively improving detection performance. In particular, SDNBoost consists of a robust adaptive voting module that combines three boosting EL techniques, namely Adaptive Boosting (AdaBoost), Gradient Tree Based Boosting (GTBM), and eXtreme Gradient Boosting (XGBoost), to effectively detect network security threats in SDN. Irrelevant network data flow features can cause long-term problems (*e.g.*, slow down the process of ML model training and prevent ML models from making accurate classification). Thus, we integrate a scalable, lightweight and optimized feature selection module to reduce computational complexity (*i.e.*, training and testing time complexity) and increase the overall detection performance (*e.g.*, accuracy and F1 score). This module consists of (a) A Network Data flow Collection scheme (NDC) to gather the features of network data flow in a scalable and efficient way using sFlow protocol; (b) An optimized feature selection scheme that investigates using both of linear measures (*i.e.*, Pearson Correlation-based Feature Selection (PCFS)) and non-linear (*i.e.*, Tree-based Information gain Feature Selection (TIGFS) and Gradient Boosting Feature Selection (GBFS)); the objective is to select the most informative/relevant features from the collected network data flow to reduce training and testing time complexity; and (c) A hyperparameter optimization scheme to enhance the overall model's performance. Finally, we design an attack mitigation module (AMM) to effectively mitigate network security threats. To evaluate the effectiveness of SDNBoost, we conduct extensive experiments using a variety of real-world network security threats including DDoS attacks. The experimental results, using two well-known public network security datasets, namely NSL-KDD [**11**] and UNSW-NB15 [**109**, **110**], show that SDNBoost achieves efficiency and high accuracy in detecting and mitigating the new emerging security threats. SDNBoost outperforms recent state-of-the-art contributions by 12% higher accuracy and F1 score while significantly reducing computational complexity, making it a promising framework to mitigate the new emerging network security threats in SDN.

The main contributions of this paper can be summarized as follows:

- We propose a novel multi-module optimized EL framework that combines optimized feature selection with advanced EL techniques using SDN.
- We propose a novel network data flow collection scheme (NDC) to gather the features of network data flow in a scalable and efficient way using the sFlow protocol.

- We propose an optimized feature selection scheme that investigates using both of linear and non-linear measures to select the most informative/relevant features from the collected network data flow.
- We propose a hyperparameter optimization scheme to enhance the overall model performance.
- We propose a novel ensemble adaptive voting module that uses three boosting EL techniques, namely AdaBoost, GTBM, and XGBoost to effectively detect network security threats.
- We propose an attack mitigation module (AMM) to effectively mitigate network security threats.
- We evaluate the performance of SDNBoost in terms of accuracy, F1 score, and computational complexity. We compare the performance of SDNBoost with state-of-the-art contributions. The experiments results show that SDNBoost can effectively detect and mitigate attacks with high accuracy/F1 score while significantly reducing computational complexity.

The remainder of this chapter is organized as follows. In Section 4.3, we present a review of related work. Then, we present a general overview of SDNBoost in Section 4.4. In Section 4.5, we describe our data collection and optimized feature selection module. Section 4.6 presents our ensemble adaptive voting module. Section 4.7 presents AMM. In Section 4.8, we evaluate SDNBoost. Finally, Section 4.9 concludes the chapter.

4.3. Related Work

The new emerging security threats continue to evolve and are becoming more devastating. Several state-of-the-art contributions have integrated ML/DL techniques to improve the efficiency of traditional IDSs. In the following, we overview the most representative ML/DL based IDSs and their security issues. Traditional IDSs use pattern-matching/attack-signatures filtering techniques to distinguish between normal and abnormal data samples. Traditional IDSs are limited by their need of new/up-to-date attack patterns, which make them vulnerable to zero-day attacks. The recent emergence of ML/DL techniques have revolutionized the security field; since then, several IDSs have adopted ML/DL to detect network intrusions.

Lin et al. [**111**] proposed a new character-level intrusion detection system that uses convolutional neural networks (CharCNN-IDS) to detect network anomalies. CharCNN-IDS consists of (1) A data processing scheme that processes network data flow at the character-level using word embedding techniques; and (2) An optimized convolutional neural networks model that uses the output of the data processing scheme to classify network data flow samples as either normal or abnormal. They evaluated the effectiveness of CharCNN-IDS in

binary and multi-class classification using the NSL-KDD dataset. Tang et al. [**112**] proposed a Deep Neural Network (DNN) model for anomaly detection in the context of SDN. They used a data preprocessing module to extract the most relevant features from the collected network data flow. They evaluated the effectiveness of their proposed model using the NSL-KDD dataset. Yin et al. [**113**] proposed the design and the implementation of a novel intrusion detection system that uses recurrent neural networks (RNN-IDS) to detect network anomalies. The authors have studied the performance of RNN-IDS in binary and multi-class classification using the NSL-KDD dataset. The experimental results did show that RNN-IDS outperforms shallow ML methods in terms of accuracy in both binary and multi-class classification. Wu et al. [**114**] proposed a novel intrusion detection model that uses convolutional neural networks (CNNs). The authors used CNNs to automatically select network traffic features from the input data; to solve the issue of imbalanced classes in the used dataset, they set a coefficient (*i.e.*, cost function weight) to each data sample based on its number. The proposed model reduces false positive rates as well as the calculation cost while improving the accuracy. Ambusaidi et al. [**115**] proposed a novel intrusion detection model called Least Square Support Vector Machine based IDS (LSSVM-IDS). LSSVM-IDS has two stages. First it uses a Mutual Information (MI) scheme to study the linear and non-linear dependence between data input features and data output classes. Then, it gives the most relevant features to the Least Squares SVM (LS-SVM) model for training and decision making (*i.e.*, classification). They evaluated the effectiveness of LSSVM-IDS using three well-known public network security datasets, KDD'99, NSL-KDD, Kyoto 2006+. Ashfaq et al. [**116**] proposed a novel fuzziness based Semi Supervised Learning (SSL) scheme that uses a single hidden layer feed-forward neural network (SLFN) along with a sample categorization scheme to detect network anomalies. Singh et al. [**117**] proposed a scalable peer-to-peer attack detection scheme that uses Random Forest to detect network anomalies. The proposed scheme consists of (1) A distributed framework for dynamic data network extraction; and (2) A classification module that uses RF model to detect, in real-time, network intrusions. The authors evaluated the performance of their proposed scheme using the CAIDA dataset.

To increase the detection performance of single ML/DL models, several contributions have combined supervised and unsupervised techniques to build an efficient intrusion detection model. Al-Qatf et al. [**118**] proposed an efficient DL framework, called self-taught learning (STL-IDS), that uses a sparse autoencoder (SAE) along with a support vector machines (SVM) method; the objective is to (1) Encode network data flow; and (2) Detect and mitigate network anomalies. STL-IDS uses a feature selection method and a dimensionality reduction scheme to reduce the training time complexity while improving the prediction accuracy using the SVM model. Ludwig [**119**] designed a neural network (NN) ensemble model that uses deep belief neural network (DBN), DNN, and extreme learning machine models to detect the different attacks using the NSL-KDD dataset. He evaluated/tested

the effectiveness of the network ensemble model, using the NSL-KDD dataset, in terms of detection rate, false alarm rate, and Area Under Curve (AUC). Shone et al. [**120**] proposed a novel deep neural networks model that combines a layer-wise unsupervised scheme with Random Forest (RF) classifier to detect anomalies in the network. The layer-wise unsupervised scheme consists of a stacked non-symmetric deep neural networks autoencoder (NDAEs) that allows to learn/extract more complex relationships between data input features. They implemented the proposed scheme on the top of Tensorflow framework and evaluated using the KDD'99 and the NSL-KDD datasets. Wang et al. [**121**] proposed a novel hierarchical spatial-temporal feature-based intrusion detection system called HAST-IDS. HAST-IDS consists of (1) A deep CNN model to learn the low-level spatial features of network traffic; and (2) An LSTM (short-term memory networks) to learn high-level temporal features. They evaluated the effectiveness of HAST-IDS using two well-known public network security datasets (*i.e.*, DARPA and ISCX2012). Gao et al. [**122**] proposed an adaptive ensemble learning method that combines common ML models (*i.e.*, Decision Trees (DT), SVM, logical regression (LR), k-nearest neighbors (KNN), Adaboost, RF, and deep neural networks) to increase the detection rate of traditional ML models. Also, they have integrated an ensemble adaptive voting scheme. They tested and evaluated the effectiveness of the proposed framework using the NSL-KDD dataset. Ahmad et al. [**123**] proposed an intrusion detection model that uses multiple ML techniques, namely, RF, SVM, and extreme learning machine (ELM). The authors evaluated the performance of each of these techniques using the NSL–KDD dataset. McDermott et al. [**124**] proposed a novel intrusion detection model that uses a feed-forward back propagation neural network and a support vector machine to detect anomalies in Wireless Sensor Networks (WSNs). The authors have compared the detection performance of the two techniques using the NSL-KDD dataset. Moustafa et al. [**125**] proposed a novel architecture that uses an Outlier Gaussian Mixture (OGM) scheme to detect web attacks. The proposed architecture consists of (1) A data pre-processing module that uses an Association Rule Mining (ARM) scheme to dynamically extract informative features; and (2) A classification module that uses an OGM scheme to detect network anomalies. The authors evaluated the effectiveness of OGM using two well-known public network security datasets (*i.e.*, Web Attack and UNSW-NB15). Moustafa et al. [**126**] proposed a new threat intelligence scheme that uses beta mixture-hidden Markov models (MHMMs) to detect network anomalies in the context of Industry 4.0. The proposed scheme consists of (1) A novel smart management module to handle/manage heterogeneous network data flow includes data from actuators and IoT sensors; and (2) A novel threat intelligence module that uses MHMMs to detect network anomalies. The authors evaluated the effectiveness of MHMMs using two well-known public network security datasets, CPS dataset of sensors and UNSW-NB15.

To allow for an efficient and easy to deploy ML/DL based IDSs, several contributions have used SDN. Ravi et al. [**127**] proposed a novel mechanism to detect and mitigate Link

Flooding Attacks (LFA) in SDN, called BALANCE. BALANCE consists of (1) A Service Based Hybrid SDN (SBHS) to collect network data flow features using OpenFlow (OF) protocol; (2) A detection scheme that is implemented in the SDN controller to detect LFA; and (3) an LFA bot mitigation scheme that is implemented in the SDN controller to mitigate LFA. The authors evaluated the effectiveness of BALANCE using the Mininet emulator. Ali et al. [**128**] proposed a three-tier intrusion detection and prevention system (IDPS) to detect Distributed Denial of Service (DDoS) attacks in SDN. The proposed IDPS includes packet validation, user validation, and flow validation. The authors evaluated the effectiveness of IDPS, using the OMNeT++ emulator, in terms of detection/failure rate, throughput, and delay. Chen et al. [**129**] proposed a new detection model that uses extreme gradient boosting (XGBoost) to detect network anomalies in SDN. The proposed model uses OF protocol to collect network data flow features. Then, the SDN controller (*i.e.*, POX controller) uses these network data flow features to detect anomalies using XGBoost classifier. The authors evaluated the effectiveness of their proposed model using the KDD'99 dataset. To handle large size and high dimensional network data flow features, He et al. [**130**] proposed two clustering-based schemes to detect network anomalies in SDN. The first scheme uses an unsupervised cluster technique to extract network data flow features. The second scheme uses a density-based clustering method to detect network anomalies using a reduced dimension dataset. The authors evaluated the performance of their proposed scheme using the KDD'99 dataset.

Based on our analysis of these existing contributions, we found that a number of these solutions [**127**–**130**] are computationally expensive. Also, most of them [**111**–**130**] suffer from high bias and/or high variance; this usually results in inefficient intrusion detection models when encountering new attacks (*i.e.*, zero-day attacks). To address the shortcomings of these existing solutions [**111**–**130**], we propose a novel multi-module optimized EL framework that uses multiple boosting techniques to build more flexible (*i.e.*, less bias) and less data-sensitive (*i.e.*, less variance) ML/DL based IDSs. Moreover, we introduce a novel data collection and optimized feature selection module to gather network data flow feature in a scalable and efficient way and to reduce computational complexity while effectively improving detection performance. SDNBoost is much scalable and accurate in comparison with the ones that use OF schemes [**127**–**130**], and outperforms recent state-of-the-art contributions by 12% higher accuracy and F1 score while significantly reducing computational complexity.

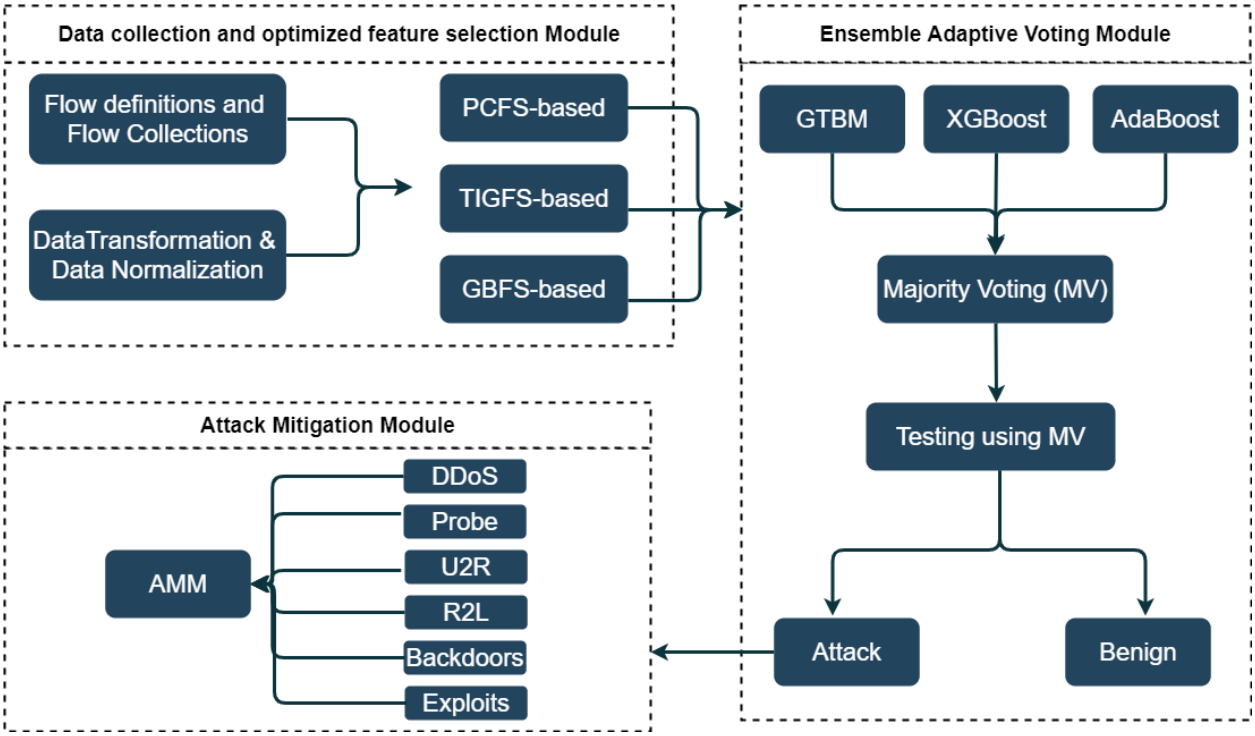


Figure. 4.1. Proposed Multi-Module EL-based IDS Framework

4.4. SDNBoost: AN Overview

4.4.1. Design Overview

This section presents an overview of SDNBoost. More specifically, we explain how SDNBoost ensures the detection and mitigation of the new emerging security threats, allowing for an accurate detection and an effective attack mitigation. When designing SDNBoost, we did consider the following goals/objectives. First, SDNBoost should ensure/guarantee a full protection from the new emerging security threats. Unlike existing ML/DL based IDSs [**111**–**130**], that use a single learner to handle intrusion detection, SDNBoost uses EL techniques to to build more flexible and less data-sensitive ML/DL based IDSs. Also, SDNBoost integrates a scalable, lightweight and optimized feature selection module to reduce computational complexity while improving the overall detection performance. Second, these anomalies/threats should be effectively and timely mitigated, using AMM, and the overall system has to be as secure/scalable as possible.

Fig. 4.1 shows our proposed multi-module optimized EL framework that combines optimized feature selection schemes with advanced EL techniques; it includes three modules. The first module is a novel data collection and optimized feature selection module that aims to reduce computational complexity and increase the overall detection performance. This

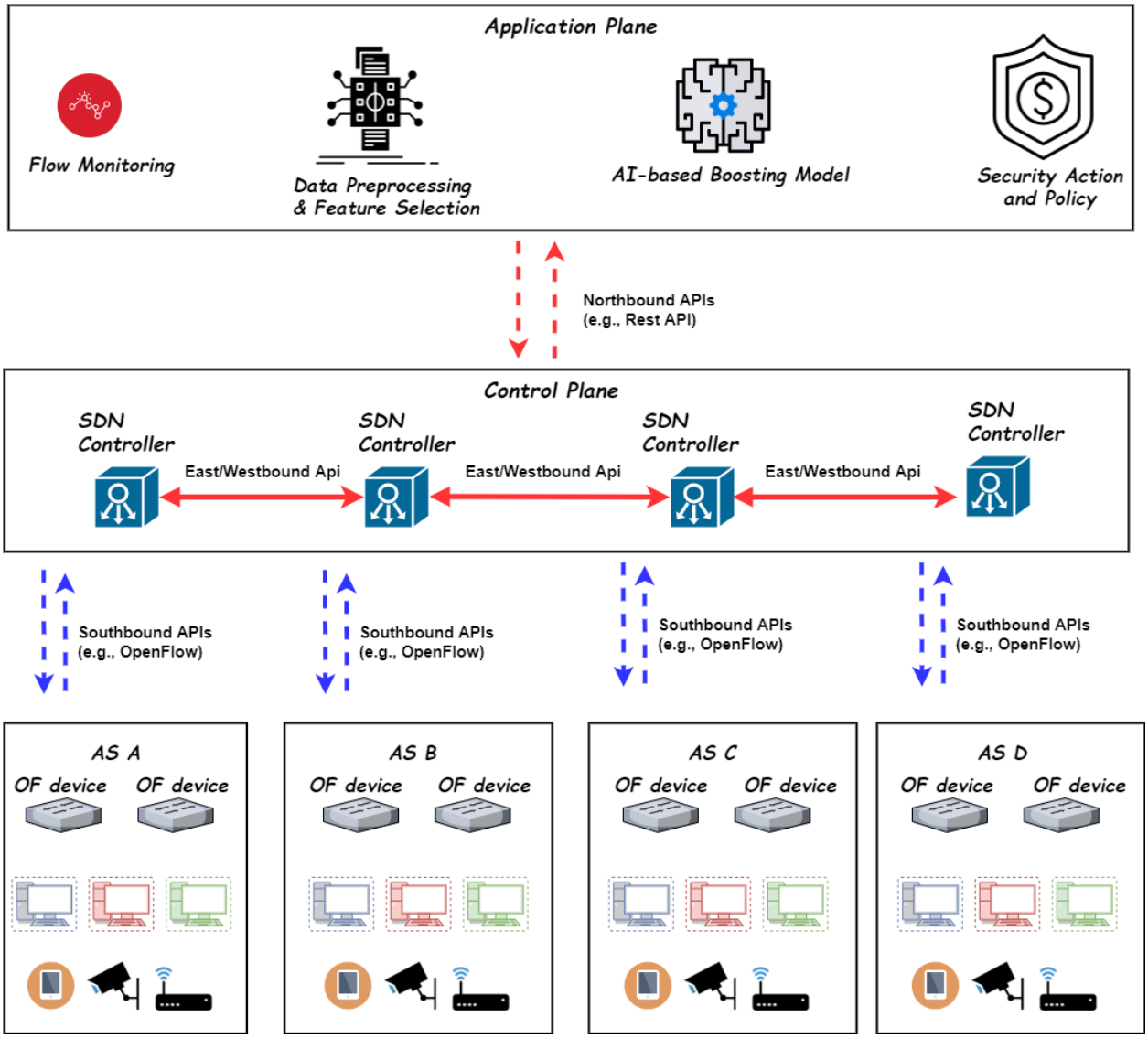


Figure. 4.2. SDNBoost's System Architecture

module consists of (a) NDC to gather the features of network data flow in a scalable and efficient way using sFlow protocol; (b) An optimized feature selection scheme that uses PCFS, TIGFS, and GBFS measures, to select the most informative/relevant features; and (c) A hyperparameter optimization scheme to enhance the overall model performance. The second module is an ensemble adaptive voting module that uses AdaBoost, GTBM, and XGBoost to effectively detect network security threats. The third module is an attack mitigation module (AMM) that effectively mitigates network security threats.

4.4.2. System Architecture

In this section we introduce the SDNBoost architecture. More specifically, we explain how SDNBoost operates. Fig. 4.2 shows the system architecture of SDNBoost. SDNBoost

consists of three planes. The first plane is the data plane that consists of multiple forwarding elements referred to as OpenFlow (OF) devices; each OF device uses the control rules, issued by the control plane (*i.e.*, SDN controller) via southbound Application Programming Interfaces (API) such as OpenFlow protocol or Netconf [**131**], to ensure the forwarding, monitoring, and processing of network data packets according to these control rules. The second plane is the control plane that consists of multiple control elements referred to as SDN controllers; each SDN controller manages a geographical distributed data plane through a southbound API and deploys the policies generated by the application plane, through a Northbound API such as Rest API [**48**], into the data plane. The control plane provides an abstractions of lower-level data plane resources utilization (*e.g.*, cpu usage, throughput, and latency) to the application plane. The third plane is the application plane that consists of multiple REST applications that generate security policies; it includes: (a) A scalable and efficient network data collection scheme to monitor, using sFlow protocol, the features of network data to obtain the global view of the network; (b) An optimized feature selection scheme that uses PCFS, TIGFS, and GBFS measures, to select the most informative/relevant features; (c) An ensemble adaptive voting module that uses AdaBoost, GTBM, and XGBoost to effectively detect network security threats; and (d) A security action and policy scheme (*i.e.*, AMM) that effectively and timely mitigates network security threats. The Northbound API (*i.e.*, REST API) is used in the detection/mitigation process to offer the inter-operability to use/manage any type of SDN controller (*e.g.*, Ryu OpenFlow controller [**3**], Floodlight OpenFlow controller [**50**]).

In the following, first we introduce our data collection and optimized feature selection module. Then, we present our ensemble adaptive voting module. Finally, we present our attack mitigation module (AMM).

4.5. Data Collection and Optimized Feature Selection Module

Data Collection and Optimized Feature Selection module aims to reduce computational complexity while effectively improving detection performance; it consists of: (1) NDC to gather the features of network data flow in a scalable and efficient way using sFlow protocol; (2) an optimized feature selection scheme that uses PCFS, TIGFS and GBFS to select the most informative/relevant features from the collected network data flow; and (3) an hyperparameter optimization scheme to enhance the overall model's performance.

4.5.1. Network data flow collection scheme (NDC)

Data collection is the first step of network security threats detection. In SDN, two commonly techniques are used to collect the features of network data flow (*e.g.*, count number of bytes per flow and count number of packet per ingress port). The first technique uses OpenFlow protocol while the second one uses flow sampling techniques such as Sflow protocol. In OpenFlow based technique, the control plane (*i.e.*, SDN controllers) periodically sends state requests (*i.e.*, $ofp\_flow\_stats\_request$) to data plane elements (*i.e.*, OF devices). Each OF device responds with one or multiple flow statistics reply messages (*i.e.*, $ofp\_flow\_stats\_reply$) that contain network flow statistics. To this end, each OF device needs to maintain/store a large number of network data flow entries. However, this will exhaust the resources of OF devices (*i.e.*, Ternary Content Addressable Memory (TCAM) of OF devices). Also, a large number of flow statistics reply messages sent by OF devices to SDN controllers may congest the bandwidth between OF devices and SDN controllers and exhaust the OF channel by the attack traffic. Furthermore, SDN controllers can not receive the flow statistics reply messages timely, making this technique (*i.e.*, OF-based technique) not suitable to detect effectively high-rate attacks. To alleviate the issues of OF-based technique, NDC makes use of flow sampling techniques to gather the features of network data flow in a scalable and efficient way using sFlow protocol. NDC is more scalable and neither consumes excessive bandwidth between OF devices and SDN controllers nor exhaust the OF channel by the attack traffic. NDC defines the flow monitoring metrics, such as flow aggregation attributes and thresholds, and uses an sFlow collector (*i.e.*, sFlow-RT [**47**]) to deploy these metrics in network data plane elements. In NDC, the sFlow collector periodically, at each interval ΔT, sends state requests to sFlow agents embedded in OF devices. Each sFlow agent responds with a set of sFlow Datagrams (*e.g.*, flow/counter samples) that are analyzed by our ensemble adaptive voting module to produce real-time and accurate decisions. NDC allows for high-speed traffic monitoring and is capable of monitoring networks at 100 Gbps and beyond, without impacting the performance of OF devices and without increasing the network load [**132**]. Once the network data flow is collected, we encode categorical features into a numeric values using a label encoder and one hot encoding technique. Then, we re-scale the values of the features values according to Eq. (4.5.1) using a standardization technique.

$$X_{stand} = \frac{X_i - \mu}{\sigma} \tag{4.5.1}$$

where X_i denotes the input feature, μ and σ denote, respectively, the mean and standard deviation values for each input feature value.

4.5.2. Optimized Feature Selection Scheme

Redundant/irrelevant network data flow features can slow down the process of training and prevent an EL model from making accurate decisions, especially when dealing with large-scale and high-dimensional data systems. SDNBoost investigates using both of linear and non-linear feature selection techniques, namely PCFS, TIGFS, and GBFS. SDNBoost selects the most informative features and explores their effect on EL model performance and training/testing time complexity.

4.5.2.1. Pearson Correlation-based Feature Selection (PCFS). PCFS measures the statistical relationship/association, referred to as linear correlation, between input features and the target class. PCFS uses a linear correlation coefficient to rank input features and selects the highest ones (*i.e.*, the highest correlated features with the target class). For an input feature F with a value f and a target class Y with a value y, the PCFS measure is defined as follows:

$$PCFS(F,Y) = \frac{Cov(F,Y)}{\rho_F \rho_Y} \tag{4.5.2}$$

where *cov* is the covariance, ρ_F is the standard deviation of F, and ρ_Y is the standard deviation of Y. Thus, PCFS can be expressed as follows:

$$PCFS(F,Y) = \frac{\sum_{j=1}^{n}(f_j - \bar{f})(y_j - \bar{y})}{\sqrt{\sum_{j=1}^{n}(f_j - \bar{f})^2}\sqrt{\sum_{j=1}^{n}(y_j - \bar{y})^2}} \tag{4.5.3}$$

where n is the number of input features, $\bar{f}$ is the mean of f, and $\bar{y}$ is the mean of y. A correlation coefficient close to 1, for a particular input feature f_j with a target y_j, indicates that f_j is highly correlated with y_j. A correlation coefficient close to 0, for a particular input feature f_j with a target class y_j, indicates that f_j is highly uncorrelated with y_j.

4.5.2.2. Tree-based Information gain Feature Selection (TIGFS). TIGFS uses entropy and mutual information concepts to select the most informative/relevant input features. TIGFS ranks input features based on entropy calculation. In particular, for an input feature F and a target class Y, TIGFS ranks F based on the amount of knowledge/information that can be gained about Y. In fact, if F and Y are independent, their TIGFS score (*i.e.*, TIGFS(F;Y)) is almost zero. Thus, F does not contain any knowledge/information about Y and it consists of an irrelevant feature that may prevent an EL model from making accurate decisions. The TIGFS measure is defined as follows:

$$TIGFS(F;Y) = H(F) - H(F/Y) \tag{4.5.4}$$

where $H(F)$ is the entropy of an input feature F, $H(F|Y)$ is the conditional entropy of an input feature F given a target class Y. Thus, TIGFS can also be expressed as follows:

$$TIGFS(F;Y) = \int_F \int_Y p_{F,Y}(F,Y) log \frac{p_{F,Y}(F,Y)}{p_F(F)p_Y(Y)} df dy \tag{4.5.5}$$

where $p_{F,Y}(F,Y)$ is the joint probability density function of input feature F and a target class Y, and $p_F(F)$ (resp. $p_Y(Y)$) are marginal density functions of F and (resp. Y), respectively.

4.5.2.3. Gradient Boosting Feature Selection (GBFS). GBFS uses a feature importance metric to select the most informative/relevant input features. In GBFS, a forest of trees is constructed using multiple boosted decision trees; each boosted tree trains on a sub-set of data. To construct an optimal boosted tree, a random sub-set of features is selected. Then, we construct multiple splits using different features. In particular, GBFS uses Gini impurity function to select the best features that maximize the information/knowledge for a particular split in the boosting tree. GBFS ranks input features based on an importance score that indicates how valuable an input feature in making key decisions. The more an input feature is valuable in the construction of the optimal boosted trees, the higher its importance score. For a particular decision tree T, a measure of relevance for the k^{th} input feature f_k is defined as follows [**133**]:

$$GBFS_k(T) = \sum_{t=1}^{N-1} \widehat{i}_t^2 I(f_t = k) \tag{4.5.6}$$

where N is the number of internal nodes of the boosted tree, $\widehat{i}^2$ is an estimated improvement in a loss error function (*i.e.*, squared error risk), and f_t is the splitting input feature that is associated with an internal node t.

Then, the feature importance is averaged across all the boosting decision trees as follows:

$$GBFS_k = \frac{1}{M} \sum_{i=1}^{M} GBFS_k(T_i) \tag{4.5.7}$$

where M is the number of trees.

4.5.3. Hyperparameter Optimization Scheme

Hyperparameter Optimization, called also hyperparameter tuning, is an approach that searches for a set of well-performing/optimal hyperparameters that can be used to achieve the best detection performance for a ML model. To enhance the overall model's performance, we use random search (RS), as hyperparameter optimization scheme. RS uses a bounded space of hyperparameter values and select randomly combinations of these hyperparameters from that space. RS consists of finding hypparameters that minimize a cost function. We define RS optimization as follows:

$$\min_{hyper \in R^n} f(hyper) \tag{4.5.8}$$

where f is the cost function to be minimized and $hyper \in \mathrm{R}^n$ is a set of hyperparameters (*i.e.*, candidate solutions) drawn from a space. It has been proven that RS is more efficient for hyper-parameter optimization than other approaches (*e.g.*, Grid Search (GS), Manual Search (MS)) [**134**]. Also, RS allows for a parallel optimization and reduces computational complexity.

4.6. Ensemble Adaptive Voting Module

This module aims to effectively detect network security threats in SDN; it consists of a robust adaptive boosting EL framework, that combines three boosting EL techniques, namely, Adaptive Boosting (AdaBoost), Gradient Tree Based Boosting (GTBM), and eXtreme Gradient Boosting (XGBoost).

4.6.1. Adaptive Boosting (AdaBoost)

AdaBoost stands for adapative boosting; it is a boosting EL technique that uses decision trees for classification. In AdaBoost, we fit a sequence of weak learners, called stumps (*i.e.*, single layer decision trees), to a weighted version of the dataset in such a way that a high weight is given to previously misclassified data samples. Thus, the model can give more attention to these misclassified data samples in the next round of training. Once the training is done, the output of each learner is combined through a weighted majority voting to construct a final classification model.

AdaBoost begins with assigning, to each data sample in the training dataset, an equal weight as follows:

$$W_i^{(0)} = \frac{1}{N} \tag{4.6.1}$$

where N is the number of data samples (*i.e.*,$\{(x_i,y_i)\}_{i=1}^{N}$).

Afterwards, we train a first stump $H_1(x)$ using the weighted version of the dataset and we compute the total error at iteration t as follows:

$$E_t = \frac{\sum_{i=1}^{N} W_i^{(t)} \mathbb{I}[H_t(x_i) \neq y_i]}{\sum_{i=1}^{N} W_i^{(t)}} \tag{4.6.2}$$

where E_t denotes the normalized sum of the weights of misclassified data samples at iteration t. Then, we compute the amount/importance of *say* that this stump has in the final classification as follows:

$$\alpha_t = \log(\frac{1 - E_t}{E_t}) \tag{4.6.3}$$

where α_t denotes the amount/importance of *say* (*i.e.*, weight) of a stump at iteration t. The lower the total error (*i.e.*, perfect stump), the higher the amount/importance of *say* the stump will have in the final classification. The higher the total error, the lower the amount/importance of *say* the stump will have in the final classification.

Then, we update the weights of misclassified data samples as follows:

$$W_i^{(t+1)} = W_i^{(t)} \exp\{\alpha_t \mathbb{I}[H_t(x_i) \neq y_i]\} \tag{4.6.4}$$

The weights of misclassified data samples will be higher in the next iteration. Thus, the model can give more attention to these misclassified data samples in the next iteration of training. After this update, the weights of the data samples are normalized.

The final model for classification becomes as follows:

$$H(x) = sign[\sum_{t=1}^{T}(\alpha_t H_t(x))] \tag{4.6.5}$$

where T is the number of stumps. The final model $H(x)$ makes the final classification and chooses the class that receives the highest total vote. Algorithm 10 shows the pseudo-code of AdaBoost. This algorithm is implemented in the application layer (*i.e.*, on the top of the SDN controller) as a REST application.

Algorithm 10: AdaBoost Algorithm

Input: Sequence of N data samples $\{(x_i,y_i)\}$, $i = 1,...,N$

Init data sample weights $W_i^{(0)}$ to $\frac{1}{N}$
for *$t \leftarrow 1$ to T* **do**
  Train a stump $H_t(x)$ using data samples weighted by $W_i^{(t-1)}$
  Compute $E_t = \frac{\sum_{i=1}^{N} W_i^{(t)} \mathbb{I}[H_t(x_i) \neq y_i]}{\sum_{i=1}^{N} W_i^{(t)}}$
  Evaluate $\alpha_t = \log(\frac{1-E_t}{E_t})$
  Update weights $W_i^{(t+1)} = W_i^{(t)} \exp\{\alpha_t \mathbb{I}[H_t(x_i) \neq y_i]\}$
end
Make final classifications: $H(x) = sign[\sum_{t=1}^{T}(\alpha_t H_t(x))]$

4.6.2. Gradient Tree Based Boosting (GTBM)

GTBM is another boosting EL model that uses gradient descent as an optimization algorithm to find optimal values that minimize a differentiable cost/loss function. More specifically, the objective is to find an approximation $\hat{y}$ to a function $H(x)$ that minimizes a cost/loss function. We used cross entropy loss function $\mathcal{L}$ as in Eq. (4.6.6) as a loss function.

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} y_i * log(\hat{y}_i) \tag{4.6.6}$$

where y_i is the ground truth vector (*i.e.*, observed/true value) for i^{th} class, $\hat{y}_i$ is the predicted value for i^{th} class, and N is the number of data samples.

GTBM begins with initializing the model with a constant value as follows:

$$H_0(x) = \arg\min_{\gamma} \sum_{i=1}^{N} \mathcal{L}(y_i,\gamma) \tag{4.6.7}$$

where N is the number of data samples (*i.e.*,$\{(x_i,y_i)\}_{i=1}^{N}$) and γ is the predicted value. Then, we compute pseudo-residuals for i^{th} data sample at iteration t as follows:

$$r_{it} = -[\frac{\partial \mathcal{L}(y_i,H(x_i))}{\partial (H(x_i))}]_{H(x)=H_{t-1}(x)} \tag{4.6.8}$$

Afterwards, we fit a weak learner (*e.g.*, decision tree) $f_t(x)$ to pseudo-residuals (*i.e.*, train a decision tree learner on the new dataset that consists of $\{(x_i,r_{it})\}_{i=1}^{N}$). Then, for each leaf in the new decision tree, we compute an output value at iteration t as follows:

$$\gamma_t = \arg\min_{\gamma} \sum_{i=1}^{N} \mathcal{L}(y_i,H_{t-1}(x_i) + \gamma f_t(x_i)) \tag{4.6.9}$$

The output value for each leaf is the value of γ that minimizes this summations. To find the optimal value of γ, we use a second-order Taylor polynomial approximation to optimize the loss function as follows:

$$\mathcal{L}(y_i,H_{t-1}(x_i) + \gamma f_t(x_i)) \simeq \sum_{i=1}^{N} [\mathcal{L}(y_i,H_{t-1}(x_i)) + g_i \gamma f_t(x_i) + \frac{1}{2} h_i \gamma^2 f_t(x_i)]$$

where $g_i = \frac{\partial \mathcal{L}(y_i,H(x_i))}{\partial (H(x_i))}$ and $h_i = \frac{\partial^2 \mathcal{L}(y_i,H(x_i))}{\partial^2 (H(x_i))}$ are first order (*i.e.*, gradient) and second order (*i.e.*, Hessian) gradient of the loss function.

Then, we compute the derivative of this loss function with respect to γ as follows:

$$\partial \mathcal{L}(y_i,H_{t-1}(x_i) + \gamma f_t(x_i)) \simeq \sum_{i=1}^{N} [g_i f_t(x_i) + h_i \gamma f_t(x_i)] \tag{4.6.10}$$

We compute the value of γ that optimizes the loss function at iteration t as follows:

$$\gamma_t = -\frac{\sum_{i=1}^{N} g_i f_t(x_i)}{\sum_{i=1}^{N} h_i f_t(x_i)} \tag{4.6.11}$$

Finally, we update the final model at iteration t as follows:

$$H_t(x) = H_{t-1}(x) + \gamma_t f_t(x_i) \tag{4.6.12}$$

Algorithm 11 shows the pseudo-code of GTBM. This algorithm is implemented in the application layer (*i.e.*, on the top of the SDN controller) as a REST application.

Algorithm 11: GTBM Algorithm

Input: Sequence of N data samples $\{(x_i,y_i)\}$, $i = 1,...,N$
A differentiable cost/loss function $\mathcal{L}(y,H(x))$
Init the model with $H_0(x) = \arg\min_\gamma \sum_{i=1}^{N} \mathcal{L}(y_i,\gamma)$
for $t \leftarrow 1$ *to* T **do**
  Compute $r_{it} = -[\frac{\partial \mathcal{L}(y_i,H(x_i))}{\partial (H(x_i))}]_{H(x)=H_{t-1}(x)}$
  Fit a weak learner $f_t(x)$ *to pseudo-residuals* $\{(x_i,r_{it})\}_{i=1}^{N}$
  Compute optimal γ_t *by solving:*
  $\gamma_t = \arg\min_\gamma \sum_{i=1}^{N} \mathcal{L}(y_i,H_{t-1}(x_i) + \gamma f_t(x_i))$
  Update the model as follows:
  $H_t(x) = H_{t-1}(x) + \gamma_t f_t(x_i)$
end
Output the final Model $H_T(x)$

4.6.3. EXtreme Gradient Boosting (XGBoost)

XGBoost has minor improvements over gradient boosting in the regularization strategy. In XGBoost, we use a more sophisticated regularization to avoid over-fitting issues. Thus, the function to optimize at iterations t becomes:

$$\mathcal{L}^t \simeq \sum_{i=1}^{N} \mathcal{L}(y_i,H_{t-1}(x_i) + f_t(x_i)) + \Omega(f_t) \tag{4.6.13}$$

where $\Omega(f_t) = \frac{1}{2}\lambda \mid\mid \xi \mid\mid$, λ is a regularization parameter and ξ is a leaf weight. To find the optimal value of ξ, we use a second-order Taylor polynomial approximation to optimize the loss function as follows:

$$\mathcal{L}^t \simeq \sum_{i=1}^{N} [\mathcal{L}(y_i,H_{t-1}(x_i)) + g_i f_t(x_i) + \frac{1}{2} h_i f_t(x_i)^2] + \frac{1}{2}\lambda \sum_{j=1}^{T} \xi_j^2$$

where $g_i = \partial \mathcal{L}(y_i,H(x_i))$ and $h_i = \partial^2 \mathcal{L}(y_i,H(x_i))$ are first order and second order gradient of the loss function, while T is the number of leaves in the tree.

Finally, we compute the optimal value of ξ that optimizes the loss function at iteration t as follows:

$$\xi_t = -\frac{\sum_{i=1}^{N} g_i}{\sum_{i=1}^{N} h_i + \lambda} \tag{4.6.14}$$

In order to take advantage of the strength of each of these algorithms (*i.e.*, AdaBoost, GTBM, and XGBoost), we propose a majority voting scheme that combines these 3 algorithms to form a stronger final EL model. This final EL model (*i.e.*, inference model) is deployed in the application layer as a REST application.

4.7. Attack Mitigation Module (AMM)

When our ensemble adaptive voting module detects network security threats in SDN, a mitigation action is performed to protect network devices. AMM installs blocking OF rules into OF switches under attack; each OF rule (*i.e.*, OF_DROP) has a high priority to match malicious data samples and drop them. For high attack traffic, AMM monitors the speed of incoming malicious data samples; if the packet rate surpasses a rate limiter, then, AMM systematically drops the malicious data samples. This is done using Meter table in OF 1.3 protocol [**54**].

4.8. Evaluation of SDNBoost

In this section, we present the evaluation of SDNBoost. First, we introduce the experimental environment. Then, we present the experimental results. Finally, we evaluate the performance of SDNBoost.

4.8.1. Experimental environment

The implementation of SDNBoost is done using scikit-learn [**135**], an open source library that integrates a wide range of supervised and unsupervised machine learning techniques. The data collection and optimized feature selection module (*i.e.*, PCFS, TIGFS, and GBFS), the ensemble adaptive voting module (*i.e.*, AdaBoost, GTBM, and XGBoost), and AMM are implemented in the application layer (*i.e.*, on the top of the SDN controller) as REST applications. To emulate a real network environment, we use Mininet [**32**], a SDN emulation tool that uses containers and virtual OpenFlow switches (*e.g.*, OpenVswitch [**31**], OpenFlow Switch [**136**]) to create a realistic virtual network. Our test environment consists of (1) SDN controllers (i.e., Floodlight [**50**]) to install OF rules to block malicious data samples; (2) Network monitors (i.e., sFlow-RT [**47**]) to monitor/gather the features of network data flow in a scalable and efficient way; and (3) Multiple OF switches and multiple hosts are simulated to act as legitimate users. For the attack traffic, we leverage two well-known public network security datasets, namely NSL-KDD and UNSW-NB15. These datasets contain recent real-world network security threats. The first one is an improvement of the classic KDD'99 dataset [**137**] that was created by MIT Lincoln Laboratory for DARPA Intrusion Detection Evaluation Program. However, KDD'99 dataset suffered from the huge number of redundant records which may lead, when training on this dataset, to a biased learning

Tableau 4.1. Details of NSL-KDD

Dataset	Records	Normal	DoS	Probe	R2L	U2R
NSL-KDDTrain	125973	67343	45927	11656	995	52
NSL-KDDTest	22544	9711	7558	2421	2754	200

Tableau 4.2. Details of UNSW-NB15

Records	Normal	Fuzzers	Analysis	Backdoors
2540044	2218761	24246	2677	2329

Tableau 4.3. Details of UNSW-NB15 (Continued)

DoS	Exploits	Generic	Reconnaissance	Shellcode	Worms
16353	44525	215481	13987	1511	174

model. The NSL-KDD dataset is a revised and cleaned-up version of KDD'99 dataset in which all redundant records were removed. NSL-KDD contains the main attack categories *i.e.*, Distributed Denial of Service (DDoS), User to Root (U2R), Probe (Probing) and Root to Local (R2L). The second one (*i.e.*, UNSW-NB15) is a synthetic dataset from Cyber range Lab of the Australian Centre for cyber Security (ACCS); it contains 100 GB of raw network packets that includes a variety of simulated attacks including DDoS attacks. UNSW-NB15 contains about three million connection records; it includes the following network attacks: analysis, fuzzers, DoS, backdoors, reconnaissance, generic, exploits, shellcode, and worms. We run our experiments on Google Colaboratory [**138**] using the Tesla T4 GPU on a PC with Intel Core i7-8750H-2.2 GHz, 16GB RAM, and GTX 1050 GPU.

4.8.2. Experimental Results

The performance of SDNBoost is evaluated using NSL-KDD and UNSW-NB15. NSL-KDD contains a reasonable number of data samples in the train set ($i.e., NSL - KDDTrain^{+}$) and in the test set ($i.e., NSL - KDDTest^{+}$). $NSL - KDDTrain^{+}$ contains 22 types of attacks, while $NSL - KDDTest^{+}$ contains 37 types of attacks; these additional attacks in the test dataset make the classification task more realistic and allow to test the generalization ability of the EL model. UNSW-NB15 includes 9 additional new types of attacks. Table 4.1 shows the total number of records in the NSL-KDD train and test sets. Table 4.2 and Table 4.3 shows the total number of records in UNSW-NB15.

NSL-KDD contains 41 input features, while UNSW-NB15 contains 49 input features. Three of the input features (*i.e.*, 'protocol_type', 'service' and 'flag') of NSL-KDD are non-numeric/categorical input features, while four of the input features (*i.e.*, 'proto', 'state',

Tableau 4.4. UNSW-NB15 labels

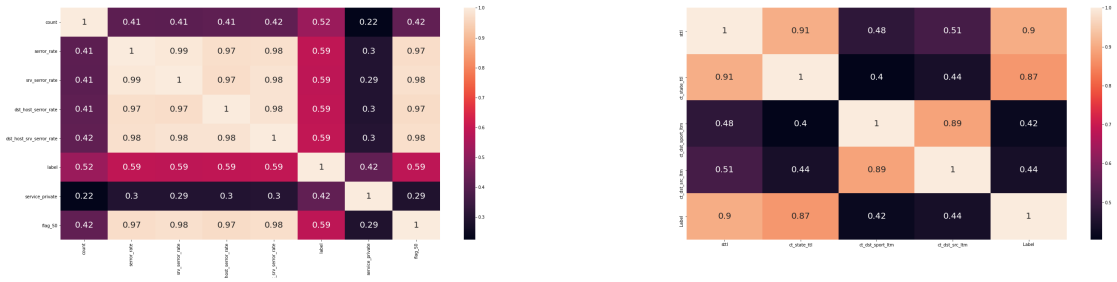
Label	Multi-class	Binary
Normal	0	0
Fuzzers	1	1
Analysis	2	1
Backdoors	3	1
DoS	4	1
Exploits	5	1
Generic	6	1
Reconnaissance	7	1
Shellcode	8	1
Worms	9	1

Tableau 4.5. NSL-KDD labels

Label	Multi-class	Binary
Normal	0	0
DoS	1	1
Probe	2	1
R2L	3	1
U2R	4	1

'service', and 'attack_cat') of the UNSW-NB15 are also non-numeric/categorical input features. For NSL-KDD, 'protocol_type' has three values (*i.e.*, 'tcp', 'udp', and 'icmp') whereas 'service' and 'flag' have, respectively, 70 and 11 distinct values. We encoded 'protocol_type' feature into numeric features using one hot encoding (*i.e.*, (0,0,1) for 'tcp', (1,0,0) for 'udp', and (0,1,0) for 'icmp'). Similarly, we encoded the two others categorical input features into numeric values. Thus, the 41 initial dimensional features are mapped into 122 dimensional features after this transformation. For UNSW-NB15, 'proto', 'state', 'service', and 'attack_cat' have, respectively, 135, 16, 13, and 10 distinct values. We encoded these categorical input features into numeric values using a label encoder technique. The data distributions of input features of the NSL-KDD and the UNSW-NB15 datasets vary widely. Some of the input features (*e.g.*, 'Sjit [0,1.48*10^6]' and 'stcpb [0,4.29*10^9]') have larger values than others (*e.g.*, 'swin [0,255]', 'num_failed_logins [0,5]', and 'Spkts [0,10646]'); this will impact the results as the final EL model will miss out important information in the features that have minimum values such as 'num_failed_logins'. Hence, we used a standardization technique to re-scale the input features values according to Eq. (4.8.1). As the last step in this preprocessing phase, the output feature ('Label'), that contains name of attack, is changed to numerical values (see Tables 4.4 and 4.5).

$$X'_i = \frac{X_i - Mean(X_i)}{stdev(X_i)} \tag{4.8.1}$$



((a)) ((b))

Figure. 4.3. PCFS scores of features for: (a) the $NSL-KDD$ dataset; and (b) the $UNSW-NB15$ dataset

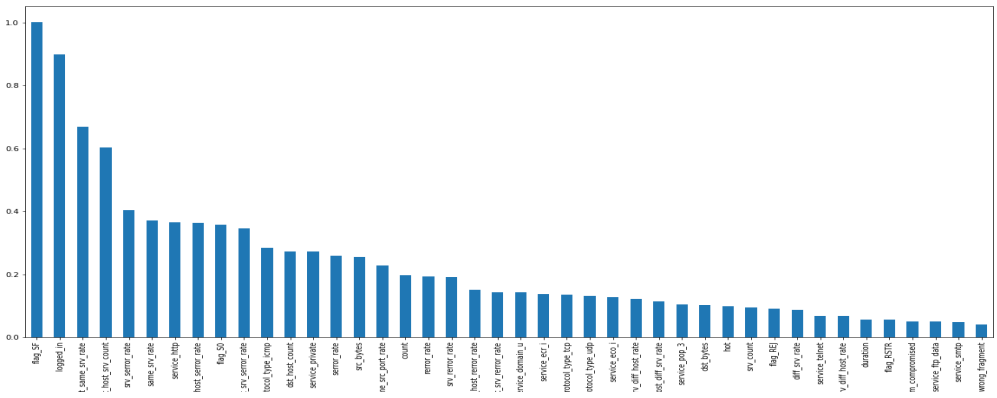


Figure. 4.4. TIGFS Score of features for $NSL-KDD$

where X_i denotes the input feature (*e.g.*, 'protocol_type'), $Mean(X_i)$ and $stdev(X_i)$ denote, respectively, the mean and standard deviation values for each input feature.

Once the preprocessing phase is done, we used our optimized feature selection module to select the most promising features and remove the redundant ones. Irrelevant features may slow down the process of training and prevent an EL model from making accurate decisions, especially when dealing with large-scale and high-dimensional data systems. SDNBoost uses linear and non-linear feature selection techniques, namely PCFS, TIGFS, and GBFS, to select the most informative features. We have applied PCFS on both datasets, Figs. 4.3(a) and 4.3(b) shows the highly correlated features with the target class (*i.e.*, features with the highest PCFS scores) for the NSL-KDD and the UNSW-NB15 datasets, respectively. For the NSL-KDD dataset, we observe that the most relevant features are those that represent the percentage of the connections that have 'SYN' errors (*i.e.*, 'serror_rate', 'srv_serror_rate', and 'host_serror_rate'). For the UNSW-NB15 dataset, we observe that the most relevant features are the time to live values (*i.e.*, 'sttl' and 'ct_state_ttl').

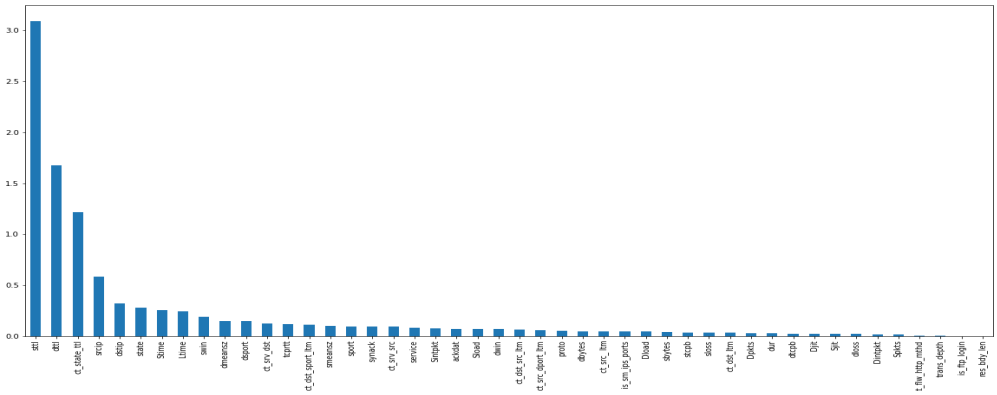


Figure. 4.5. TIGFS Score of features for $UNSW - NB15$

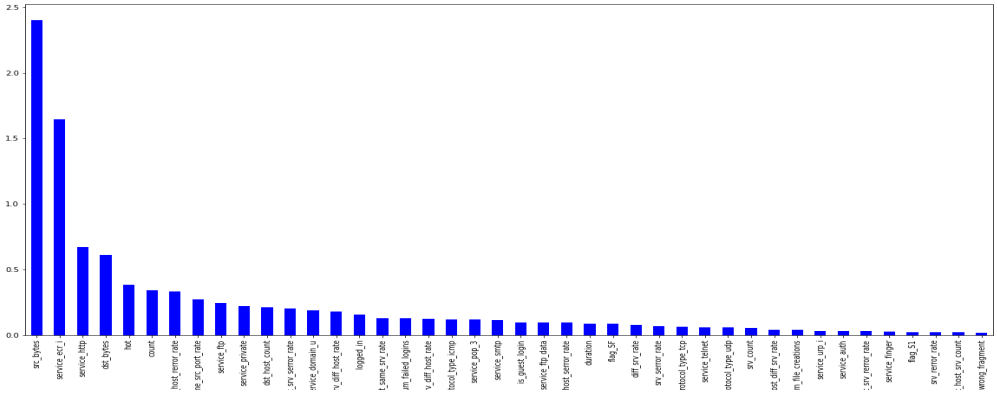


Figure. 4.6. GBFS Score of features for $NSL - KDD$

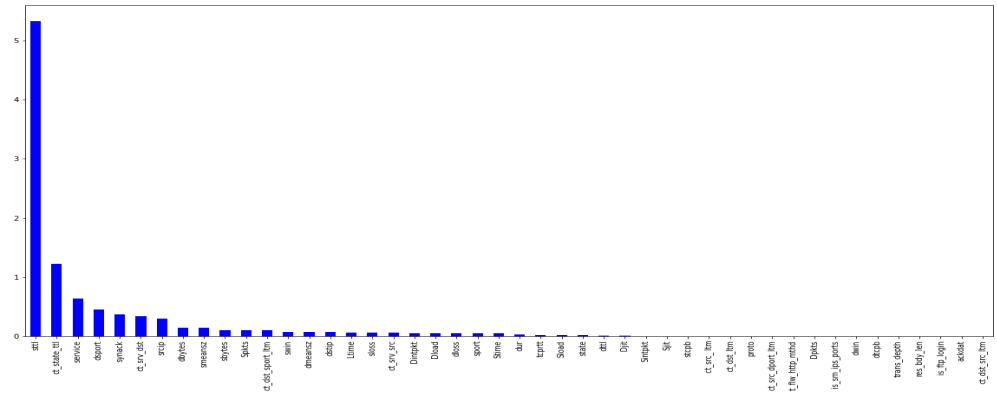
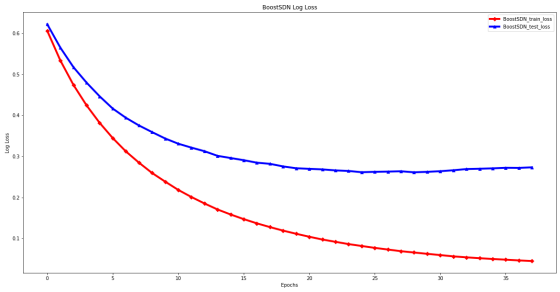
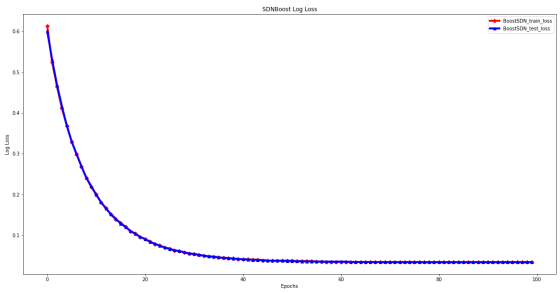


Figure. 4.7. GBFS Score of features for $UNSW - NB15$

Figs. 4.4 and 4.5 show the TIGFS score of features for NSLKDD and UNSW-NB15 datasets, respectively; they show the highest scoring features (*i.e.*, the features that have gained the highest amount of knowledge/information) in a descending order. We observe that more



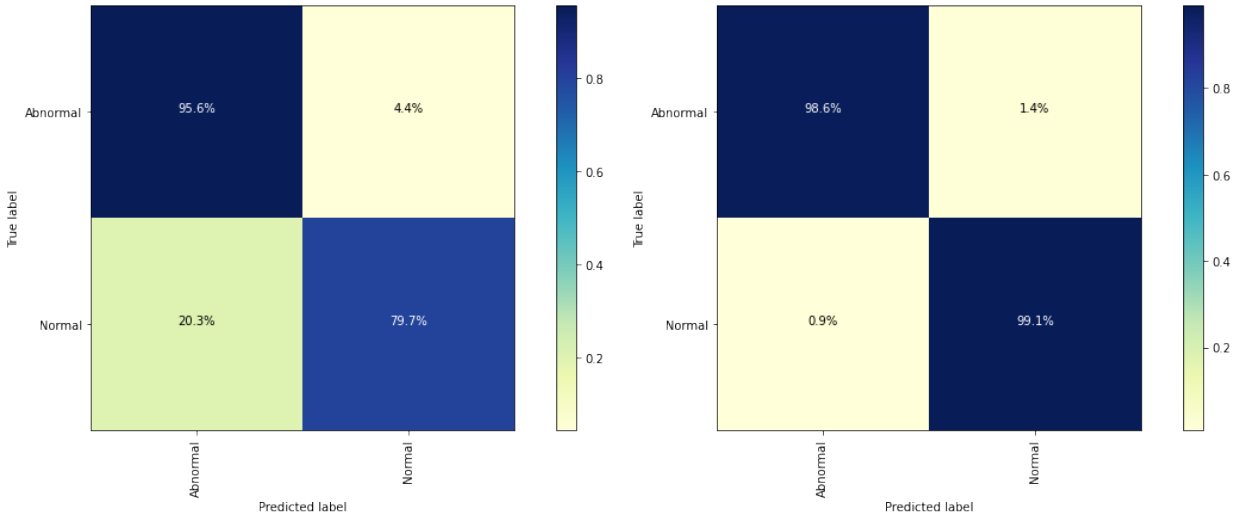
((a))



((b))

Figure. 4.8. Model loss for (a) $NSL - KDDTest^{+}$; and (b) UNSW-NB15

than 70% of the 122 features of NSLKDD are not contributing to make accurate classifications/decisions, while more than 75% of the 49 features of UNSW-NB15 are not contributing to make accurate classifications/decisions. Figs. 4.6 and 4.7 show the GBFS score of features for NSLKDD and UNSW-NB15 datasets, respectively; they show the highest scoring features (*i.e.*, the features that have gained the highest importance score) in a descending order. We observe that more than 80% of the 122 features of NSLKDD are not contributing to make accurate classifications/decisions, while more than 85% of the 49 features of UNSW-NB15 are not contributing to make accurate classifications/decisions. Our optimized feature selection module can greatly exclude the redundant/irrelevant features that can slow down the process of training and prevent an EL model from making accurate decisions. Thus, SDNBoost can significantly reduce computational complexity while maintaining/ensuring a high detection performance (*i.e.*, higher accuracy and F1 score). Figs. 4.8(a) and 4.8(b) show the learning curves of the EL model over epochs; they show the negative log likelihood loss values for NSLKDD and UNSW-NB15 datasets, respectively, during training and testing phases. We observe that the training and testing losses decrease, in both datasets, until they reach the minimum (*i.e.*, a point of stability for NSLKDD and almost zero for UNSW-NB15). In the



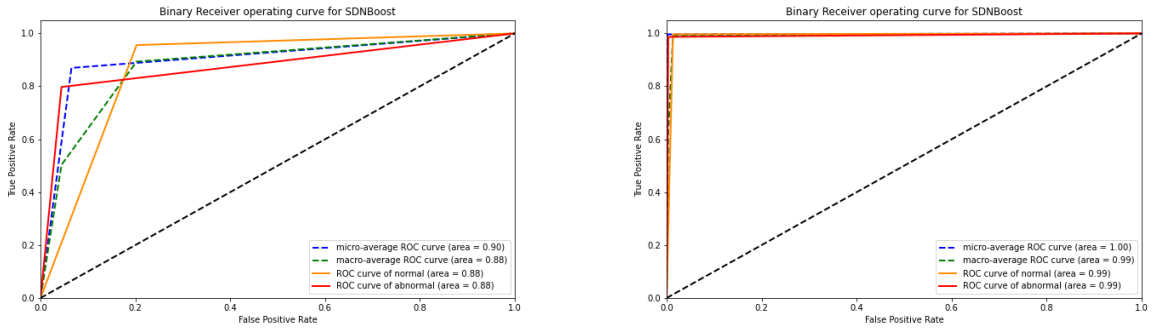
((a)) ((b))

Figure. 4.9. Confusion matrices for (a) $NSL - KDDTest^{+}$; and (b) UNSW-NB15

Tableau 4.6. Confusion matrix.

	Classified as abnormal	Classified as normal
abnormal data	TP (True Positives)	FN (False Negatives)
normal data	FP (False Positives)	TN (True Negatives)

following section, we study the performance of SDNBoost, and we compare our results with state-of-the-art ML/DL models.



((a)) ((b))

Figure. 4.10. ROC Curves for (a) $NSL - KDDTest^{+}$; and (b) UNSW-NB15

4.8.3. Performance Evaluation

We evaluate the performance of SDNBoost in terms of Accuracy, Precision, True Positive Rate (TPR) called Detection Rate (DR), Area Under the ROC Curve (AUC), and F1-score. Also, we measure the performance of SDNBoost on both datasets using ROC curves. ROC curves shows TPR according TP False Positive Rate (FPR). Finally, we use confusion matrices to show the complete performance of SDNBoost (see Table 4.6).

TP (True Positives) represent abnormal data samples that are correctly classified as abnormal data samples, FN (False Negatives) represent abnormal data samples that are classified as normal data samples, FP (False Positives) represent normal data samples that are classified as abnormal data samples, while TN (True Negatives) represent normal data samples that are correctly classified as normal data samples.

Accuracy is the percentage of the number of correct classifications over the total number of classifications. Accuracy is defined as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.8.2}$$

Precision is the percentage of the number of correct classifications of abnormal data samples over the total number of classified abnormal data samples. Precision is a good measure for imbalanced datasets; it evaluates specific class performance metrics. Precision is defined as follows:

$$Precision = \frac{TP}{TP + FP} \tag{4.8.3}$$

True Positive Rate (TPR) or Detection Rate (DR) is the percentage of the number of correct classification of abnormal data samples over the total number of presented abnormal data samples. TPR is defined as follows:

$$DR = TPR = \frac{TP}{TP + FN} \tag{4.8.4}$$

F1 score combines precision and detection rate metrics into a single metric; it is the harmonic mean of the precision and recall and it is defined as follows:

$$F1 = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{4.8.5}$$

False positive rate (FPR) is the percentage of the number of abnormal data samples incorrectly classified as normal data samples over the total number of normal samples. FPR is defined as follows:

$$FPR = \frac{FP}{TN + FP} \tag{4.8.6}$$

Tableau 4.7. Performance metrics of SDNBoost

Dataset	Accuracy	Precision	Recall	F1	Time(s)
NSL-KDDTest	87%	96%	88%	88%	4,35
UNSW-NB15	99%	99%	99%	99%	58.08

Tableau 4.8. Performance metrics of SDNBoost and state-of-the-art ML/DL models based on the available measures using $NSL - KDDTest^{+}$

Methods	Accuracy	Precision	Recall	F1	Time (second)
J48 [**139**]	0.81	N/A	N/A	N/A	N/A
NB [**139**]	0.76	N/A	N/A	N/A	N/A
RF [**139**]	0.80	N/A	N/A	N/A	N/A
MLP [**139**]	0.77	N/A	N/A	N/A	N/A
SVM [**139**]	0.70	N/A	N/A	N/A	N/A
CharCNN-IDS [**111**]	0.85	0.91	0.81	0.86	N/A
ResNet50 [**140**]	0.79	0.91	0.69	0.79	N/A
GoogleNet [**140**]	0.77	0.91	0.65	0.76	N/A
DNN [**112**]	0.75	0.83	0.75	0.74	N/A
RNN-IDS [**113**]	0.83	N/A	0.83	N/A	5516
LSSVM-IDS [**115**]	0.78	N/A	0.78	N/A	N/A
STL-IDS [**118**]	0.84	0.96	0.76	0.85	673.031
AdaMet [**122**]	0.85	0.86	0.85	0.84	N/A
SDNBoost	**0.87**	**0.96**	**0.88**	**0.88**	**4.35**

Tableau 4.9. Performance metrics of SDNBoost and state-of-the-art ML/DL models based on the available measures using UNSW-NB15

Methods	Accuracy	TPR	Time (second)
SSL [**116**]	0.86	0.85	N/A
RF [**117**]	0.93	0.92	N/A
FeedWSN [**124**]	0.92	0.91	N/A
OGM [**125**]	0.95	0.94	N/A
MHMM [**126**]	0.96	0.95	N/A
SDNBoost	**0.99**	**0,99**	**58.08**

We evaluate the performance of SDNBoost in both datasets *i.e.*, the NSL-KDD and the UNSW-NB15. When training the EL model, we tried to minimize the loss (see Figs. 4.8(a) and 4.8(b)) and maximize the detection performance in terms of accuracy and F1-score while significantly reducing computational complexity.

Figs. 4.9(a) and 4.9(b) show the confusion matrices on $NSL-KDDTest^{+}$ and UNSW-NB15 datasets, respectively. For $NSL-KDDTest^{+}$, SDNBoost achieves 87%, 96%, 88%, 88% in accuracy, precision, recall, and F1 score, respectively, with only 4.35 seconds of training time. For UNSW-NB15, SDNBoost achieves 99%, 99%, 99%, 99% in accuracy, precision, recall, and F1 score, respectively, with only 58.08 seconds of training time. Table 4.7 shows the detailed performance of SDNBoost.

The ROC curves show TPR according to FPR. The area under the ROC Curve (AUC) measures the degree of separability between the output classes. The higher the AUC score (*i.e.*, close to 1), the better the model in distinguishing between normal and abnormal data samples. Figs. 4.10(a) and 4.10(b) show the ROC curves on $NSL-KDDTest^{+}$ and UNSW-NB15 datasets, respectively. We obtain an AUC of 0.9 and 1 in $NSL-KDDTest^{+}$ and UNSW-NB15 datasets, respectively. The experiment results show that SDNBoost achieves promising results in both datasets. In the following, we compare our results with state-of-the-art ML/DL models.

4.8.4. Comparative Analysis

In this section, we compare the results obtained by SDNBoost with the state-of-the-art ML/DL models in both datasets. For $NSL-KDDTest^{+}$, Tavallaee et al. [**139**] measured the accuracy of the following ML models: J48, Naive Bayes (NB), Random Forest (RF), Multi-layer Perceptron (MLP), and Support Vector Machine (SVM). We also compare SDNBoost with the following recent state-of-the-art DL models: Character-level Intrusion based on convolutional neural networks (CharCNN-IDS) [**111**], convolutional neural networks (ResNet50, GoogLeNet) [**140**], Deep Neural Networks (DNN) [**112**], recurrent neural networks (RNN-IDS) [**113**], LSSVM-IDS [**115**], combined Sparse Autoencoder with SVM (STL-IDS) [**118**], and an adaptive ensemble learning method (AdaMet) [**122**]. For UNSW-NB15, we compare SDNBoost with the following recent state-of-the-art ML/DL models: SSL [**116**], RF [**117**], FeedWSN [**124**], OGM [**125**], and MHMM [**126**]. Tables 4.8 and 4.9 show the values of the metrics of SDNBoost and the state-of-the-art ML/DL models in the NSL-KDD and the UNSW-NB15, respectively. We observe that, in $NSL-KDDTest^{+}$, SDNBoost achieves the highest accuracy of 87% and the highest F1 score of 88% with only 4.35 seconds of training time. We also observe that, in UNSW-NB15, SDNBoost achieves the highest accuracy of 99% and the highest TPR of 99% with only 58.08 seconds of training time. The experimental results confirm that SDNBoost outperforms state-of-the-art contributions by 12% higher accuracy and F1 score while significantly reducing computational complexity. This makes SDNBoost a promising cyber-security framework that mitigates the new emerging security threats while significantly reducing the computational complexity.

4.9. conclusion

In this chapter, we proposed a novel multi-module optimized EL framework that combines optimized feature selection schemes with advanced EL techniques using SDN. First, we introduced a novel data collection and optimized feature selection module to reduce computational complexity while effectively improving detection performance. Then, we proposed an ensemble adaptive voting module that uses three boosting EL techniques, namely Adaptive Boosting (AdaBoost), Gradient Tree Based Boosting (GTBM), and eXtreme Gradient Boosting (XGBoost) to effectively detect network security threats. Finally, we designed AMM to effectively mitigate network security threats. The experimental results on two well-known public network security datasets *i.e.*, the NSL-KDD and the UNSW-NB15, showed that SDNBoost achieved high accuracy/F1 score in detecting new security threats while significantly reducing computational complexity, making it a promising framework to cope with the new emerging security threats in SDN.

Chapitre 5

CoFed: A privacy preserving collaborative DDoS Mitigation framework based on Federated Learning using SDN and blockchain

5.1. Abstract

Distributed denial-of-service (DDoS) attacks continue to grow at a rapid rate plaguing Internet service providers (ISPs) and network operators in a stealthy way. Thus, intrusion detection systems (IDSs) must evolve to cope with these increasingly sophisticated and challenging security threats. Traditional IDSs are prone to zero-day attacks since they are usually signature-based detection systems. The recent rise of machine learning (ML) and deep learning (DL) techniques can help strengthen these IDSs. However, the lack of up-to-date labeled training datasets makes these ML/DL based IDSs inefficient. The privacy nature of these datasets and widespread emergence of adversarial attacks make it difficult for major organizations to share their sensitive data. Federated Learning (FL) is gaining momentum from both academia and industry as a new sub-field of ML that aims to train a global statistical model across multiple distributed users, referred to as collaborators, without sharing their private data. Due to its privacy-preserving nature, FL has the potential to enable privacy-aware learning between a large number of collaborators. This chapter presents a novel framework, called CoFed, that allows multiple software defined networks (SDN) domains (*i.e.*, collaborators) to collaboratively build a global intrusion detection model without sharing their sensitive private data. In particular, CoFed consists of: (1) a novel distributed architecture that allows multiple SDN domains to securely collaborate in order to cope with sophisticated security threats while preserving the privacy of each SDN domain; (2) a novel Secure Multiparty Computation (SMPC) scheme to

securely aggregate local model updates; and (3) a blockchain based scheme that makes use of Ethereum's smart contracts to maintain the collaboration in a fully decentralized, trustworthy, flexible, and efficient manner. To the best of our knowledge, CoFed is the first work that leverages FL, SDN and blockchain technologies to mitigate the new emerging security threats in large scale. We conducted extensive experiments to evaluate CoFed using different real-world network attacks; the experimental results using the well-known public network security dataset NSL-KDD show that CoFed achieves efficiency and high accuracy in detecting the new emerging security threats in both binary and multi-class classification while preserving the privacy of collaborators, making it a promising framework to mitigate large scale new attacks in SDN.

Keywords: DDoS; IDS; Federated Learning; SDN; inter-domain mitigation; Blockchain; SMPC; Smart contracts.

Status: This journal paper is Submitted. Z. Abou El Houda, A. Senhaji Hafid and L. Khoukhi, **"CoFed: A privacy preserving collaborative DDoS Mitigation framework based on Federated Learning using SDN and blockchain,"** IEEE Transactions on Network Science and Engineering, 2021.

5.2. Introduction

Distributed Denial of Service (DDoS) attacks are considered as one of the top security threats and the most devastating cyber-attacks that are still bringing collateral damage to Internet service providers (ISPs) and network operators. Recent dramatic incidents that exceed a rate of 1 Tbit/s [**6**] of traffic attack showed that DDoS attacks have been increasing in strength and sophistication and becoming more destructive. These major security threats cost huge financial losses each year for both business and educational organizations. The proliferation of Internet of Things (IoT) botnets and the rapid growth in the number of insecure IoT devices, with an estimation of 75 billion connected IoT device by 2025 [**9**], can give rise to large scale and devastating DDoS attacks. The emergence of new techniques in cyber-attacks (*i.e.*, botnet as a service) provides attackers with more sophisticated tools to launch new attacks that may cause various types of damage *e.g.*, theft of sensitive data, service disruption of some critical domains (*e.g.*, healthcare and finance), and network intrusion. Therefore, there is a need for highly effective intrusion detection systems (IDSs) to secure networks from DDoS attacks and all kind of intrusive activities. IDSs play a vital role in ensuring network security and are considered as the first layer of defense against network security threats; IDSs aim to effectively and timely detect such security threats while having

a low rate of false positives. However, traditional IDSs are patterns/signature-based detection systems which makes it difficult to detect zero-day attacks (*i.e.*, unforeseen/unusual attacks). In addition, a significant effort is required in order to continuously improve and evolve such IDSs to detect zero-day attacks.

The recent advent of machine learning (ML) and deep learning (DL) techniques have revolutionized many fields such as vision and speech (*e.g.*, image recognition and speech recognition). The results achieved with these techniques in some fields have surpassed the human performance. IDSs adopted ML/DL techniques to effectively detect intrusive activities and security threats of network traffic. With the adoption of ML/DL techniques, IDSs can detect both existing and new attacks without continuously updating the rules of traditional IDSs. However, the lack of balanced and up-to-date labeled training datasets makes these ML/DL based IDSs inefficient when encountering new emerging security threats (*i.e.*, zero-day attacks). The privacy nature of these datasets as well as the widespread emergence of adversarial attacks [**10**], prevent major organizations from sharing their sensitive datasets with research communities in order to develop efficient and up-to-date ML/DL models that mitigate the new emerging security threats.

Recently, Federated learning (FL) has emerged as a new sub-field of ML that aims to train a global shared model across multiple distributed collaborators without centrally storing their sensitive data. In FL, instead of bringing sensitive data of each collaborator to a central server in order to train a ML model, each collaborator trains locally a shared global model using its local training data, and sends only the local model updates. FL can significantly reduce the privacy risks and the communication costs (*e.g.*, network bandwidth consumption) since only small messages are communicated (*i.e.*, model updates), in contrast to sending the entire raw training data over the network [**141**]. Moreover, it has been proven that FL is robust to the non-identically distributed (non-IID) and unbalanced data distributions [**142**].

FL has the potential to radically disrupt major privacy-sensitive domains such as healthcare and network security, in which the sensitivity constraints surround the data, with its promising key features of collaborative learning as well as privacy preserving. The adoption of FL in the context of collaborative intrusion detection will allow major cyber-security organizations such as Kaspersky [**143**] and McAfee [**144**] to collaboratively build an efficient and up-to-date intrusion detection model in a fully distributed manner. Thus, greatly improve the efficiency of traditional ML/DL based IDSs to cope with zero-day attacks while preserving the privacy of each collaborator (*i.e.*, collaborator's sensitive data). More especially, FL enables a privacy-aware collaborative learning between multiple collaborators, making it an ideal candidate for such collaborative IDSs. FL has distinct privacy advantages in comparison with centralized training approaches that persist raw data; it has proven its effectiveness in major fields [**145**–**153**], since only minimal updates are necessary to train a particular model. However, FL is still vulnerable to reverse-engineering attacks; indeed, these attacks

may extract sensitive information about the used training data only from the locally computed model updates sent by each collaborator. To alleviate this issue, we propose the use of a secure aggregation scheme, based on Secure Multiparty Computation (SMPC) approach, to securely aggregate the locally computed model updates from each collaborator in such a way that no one can see anyone's updates.

Recently, software defined network (SDN) has emerged as a novel technology that leverages network programmability to solve the limitations of conventional networks. SDN provides new capabilities through a logically centralized component, called SDN controller, to manage the network dynamically. SDN brings numerous benefits by decoupling the control plane from the data plane which allows for more control over networks and brings new capabilities to cope with security threats ranging from DDoS attacks to data leakage.

Based on our review of SDN and FL, we believe that their joint use will change the cybersecurity landscape. The recent rise of blockchain and smart contracts opens the door for a plethora of new decentralized applications that address the challenges faced by the current collaborative intrusion detection systems (*e.g.*, centralized management) that suffer from single-point-of-failure. Blockchain has proven its success and effectiveness in achieving high level of trustworthiness and decentralization in major fields (*e.g.*, Infrastructure-as-a-Service (IaaS) [**154**], Smart Grid [**155**], and healthcare [**67**]). Blockchain technology allows to design secure, trustworthy, transparent, and decentralized collaboration systems. In this chapter, we exploit the decentralized nature of blockchain in order to design a collaborative intrusion detection system that makes use of Ethereum's smart contracts to maintain a collaboration between SDN domains that is fully decentralized, trustworthy, flexible, and efficient. Thus, ensuring integrity, transparency, and reliability of the whole system.

This chapter presents the design, specification, and implementation of a novel framework, called CoFed, that leverages FL, SMPC, SDN and blockchain technologies to cope with the new emerging security threats on a large scale. CoFed is a distributed, efficient, trustworthy, flexible, and privacy-preserving framework that manages DDoS collaboration process across multiple SDN domains. First, we implemented CoFed using pysyft [**156**], a generic framework for privacy preserving deep learning, and we deployed it on Ethereum official test network Ropsten [**94**], an open blockchain platform. Then, we evaluated CoFed using the well-known public network security dataset NSL-KDD [**11**] that contains the main attack categories *i.e.*, Distrusted Denial of Service (DDoS), User to Root (U2R), Probe (Probing) and Root to Local (R2L). The experimental results show that CoFed outperforms state-of-the-art ML/DL models in terms of accuracy and F1 score in both binary and multi-class classification while preserving the privacy of collaborators.

The main contributions of this chapter can be summarized as follows:

- We design a complete decentralized, efficient, and secure collaborative DDoS mitigation framework (CoFed) based on FL using blockchain and SDN.
- We propose a novel distributed architecture that allows multiple SDN domains to collaboratively build an efficient intrusion detection model that copes with new and sophisticated security threats while preserving the privacy of each SDN domain.
- We propose a secure aggregation scheme that uses Secure Multiparty Computation (SMPC) approach to securely aggregate local model updates from each collaborator.
- We propose a blockchain based scheme that makes use of Ethereum's smart contracts to maintain a collaboration that is fully decentralized, trustworthy, flexible, transparent, tamper-proof, and efficient.
- We evaluate the performance of CoFed in terms of detection rate, accuracy, and efficiency. We compare the performance of CoFed with the centralized ML/DL methods. The experimental results show that CoFed achieves privacy, efficiency, and security making it a promising framework to mitigate large scale attacks in SDN.

The rest of the chapter is organized as follows. Section 5.3 presents the background and related work. Section 5.4 presents an overview of CoFed. Section 5.5 describes the design and specification of CoFed. Section 5.6 presents the implementation and the performance evaluation of CoFed. Finally, Section 5.7 concludes the chapter.

5.3. Background And Related Work

Several schemes in the literature have integrated ML/DL techniques in IDSs to improve the efficiency of traditional IDSs. However, few of them consider the problem of the lack of balanced and up-to-date labeled datasets. In this section, we briefly describe attack categories considered in this work. Then, we overview the most prominent intrusion detection schemes as well as their security issues.

5.3.1. Categories of Attacks

In this work, we used the well-known public network security dataset NSL-KDD, which is an improvement of the classic KDD'99 dataset [**137**] that was created by MIT Lincoln Laboratory for DARPA Intrusion Detection Evaluation Program. KDD'99 records were collected over nine weeks during the International Knowledge Discovery and Data Mining Competition in a local area network (LAN) that simulates a typical United States Air Force LAN; it contains about five million connection records. However, KDD'99 dataset suffered from the huge number of redundant records which may lead, when training on this dataset, to a biased learning model. The NSL-KDD dataset is a revised and cleaned-up version of KDD'99 dataset in which all redundant records were removed. Therefore, a ML model, when trained on this dataset, will not be biased towards the redundant records. Moreover, the

Tableau 5.1. Sub-classes of each attack category of NSL-KDD dataset

Attack categories	**Sub-classes**
DDoS	back, land, pod, neptune, smurf, teardrop, mailbomb, apache2, processtable, udpstorm, worm
Probe	ipsweep, nmap, portsweep, satan, mscan, saint
U2R	buffer_overflow, perl, rootkit, sqlattack, ps, xterm
R2L	ftp_write, guess_passwd, imap, multihop, phf, spy, warezclient, warezmaster, sendmail, named, snmpgetattack, snmpguess, xlock, xsnoop, httptunnel

dataset contains a reasonable number of records in each of train and test sets. Table 5.1 shows the four major attack categories as well as the sub-classes of each attack category of the NSL-KDD dataset.

- Distributed Denial of Service (DDoS): is a type of attacks in which a large number of attackers try to flood a victim with massive number of illegitimate traffic in order to deny legitimate users' access to the victim's machine (*e.g.*, DNS amplification attacks).
- Probe (Probing): is a type of attacks in which the attacker tries to gather network's information in order to steal user's sensitive data (*e.g.*, port scanning attacks).
- User to Root Attack (U2R): is a type of attacks in which the attacker exploits the system's vulnerability to gain root privileges/access to that system. First, the attacker starts with an access to the system with a normal user account using some techniques like password sniffing or social engineering. Then, the attacker tries to gain access to the system as a root (*e.g.*, buffer overflow attacks).
- Remote to Local Attack (R2L): is a type of attacks in which the attacker exploits some vulnerabilities to gain local access to a remote machine (*e.g.*, guessing password attacks).

The NSL-KDD dataset contains 41 features per record where 31 are continuous, 7 are discrete and 3 are categorical. The features of the NSL-KDD dataset can be classified into four categories (see Tables 5.11 and 5.12 in the appendix): (1) Intrinsic: it represents the features that are extracted from the header of network traffic flow; (2) Content: it represents the features that are extracted from the payload of network traffic flow, it contains information about the content of the flow traffic; (3) Time-based traffic: it represents the time-based features that contain information about the number of connections made to the same host over a two seconds window; and (4) Host-based traffic: it represents the features

that contain information about the number of connections made to the same host over a series of connections.

5.3.2. Intrusion detection Schemes

Security threats continue to evolve and are becoming more devastating; several intrusion detection schemes have been proposed in the literature to cope with these attacks by integrating ML/DL techniques. In the following, we overview some of the most representative schemes and their limitations. In traditional IDSs, a set of rules (*i.e.*, attack signatures) are used to distinguish between normal and malicious behaviors; any activity that matches the predefined rules is classified as an attack. These IDSs are prone to zero-day attacks and need a continuous update to detect new attacks. The recent advent of ML and DL techniques have revolutionized the security field; several schemes have adopted these techniques to detect intrusions. Zhang et al. [**157**] proposed a new systematic framework that uses Random Forests (RF), sampling techniques, and a feature selection algorithm to detect network anomalies. In addition, they have employed a new unsupervised outlier detection scheme to improve the overall detection performance. The effectiveness of the proposed framework was evaluated using KDD'99 dataset. Yang et al. [**158**] proposed a combined network intrusion detection model that uses a deep belief network (DBN), a multi-restricted Boltzmann machine (RBM), a back propagation (BP) network, and support vector machine (SVM) to detect anomalies in wireless networks. Their detection model consists of four modules: (1) a data acquisition module, to select the appropriate dataset from the wireless network; (2) a data pre-treatment module, to transform the dataset (*i.e.*, digitization and normalization); (3) a data analysis module, to split train and test sets; and (4) a detection module, to perform the classification, handle anomalies and store them in a database.

Tao et al. [**159**] proposed an intrusion detection scheme (FWP-SVM-GA) based on genetic algorithms (GA) and SVM. To improve the training speed of SVM, they used a GA population search strategy. Then, they proposed a new fitness function in order to decrease the SVM error rate while maintaining a high true positive rate. Jan et al. [**160**] proposed a lightweight intrusion detection system for anomaly detection in Internet of Things (IoT). They used a packet arrival rate as input to the SVM classifier to detect intrusions in the IoT network. Farahnakian et al. [**161**] proposed a Deep Auto-Encoder (DAE) approach for intrusion detection systems. To avoid overfitting and local optima, they trained their proposed DAE in a greedy layer wise fashion. They evaluated the effectiveness of DAE using the KDD'99 dataset; they studied the performance of DAE in both binary and multi-class classification. Diro et al. [**162**] proposed a distributed deep learning (DL) attack detection system in IoT at the fog level. The fog nodes are responsible for training and hosting DL models. They compared the performance of their DL model against shallow ML models in

both binary and multi-class classification. Yin et al. [**163**] proposed the design and implementation of an intrusion detection system based on recurrent neural networks (RNN-IDS). The experimental results showed that RNN-IDS outperforms traditional ML models in terms of detection accuracy in both binary and multi-class classification.

In order to increase the detection rate of ML/DL models, several contributions have combined supervised and unsupervised approaches to build an efficient intrusion detection model. Teng et al. [**164**] proposed a self-adaptive and collaborative intrusion detection method based on decision trees (DTs) and SVM algorithms. They evaluated the effectiveness of the proposed framework using the KDD'99 dataset. The experimental results showed that the proposed method outperforms the use of only a single type of shallow ML models in terms of detection and recall rate. Majjed et al. [**165**] developed a self-taught learning (STL-IDS) framework by combining a sparse autoencoder (SAE) and SVM. The use of SAE in the STL framework helped to effectively represent the raw data while SVM was used for the classification task. STL-IDS reduced considerably the time of training. Wang et al. [**166**] proposed a novel intrusion detection system called hierarchical spatial-temporal feature-based intrusion detection system (HAST-IDS). HAST-IDS has two stages: (1) it uses deep convolutional neural networks (CNNs) to learn the low-level spatial features of network traffic; and (2) it uses the LSTM networks (short-term memory networks) to learn a high-level temporal feature. They evaluated the effectiveness of HAST-IDS using the standard DARPA and ISCX2012 datasets. Perez et al. [**167**] proposed multiple hybrid ML models (*i.e.*, supervised and unsupervised learning algorithms) for intrusion detection task. More specifically, they combined Neural Networks (NN) and SVM with radial kernel for supervised learning, and K-Means for unsupervised learning. Moreover, they used two other techniques (*i.e.*, Gradual Feature Reduction (GFR) and Principal Component Analysis) for feature selection. Gao et al. [**168**] proposed an adaptive ensemble learning model based on common ML algorithms (*i.e.*, Decision Trees (DT), SVM, logical regression (LR), k-nearest neighbors (KNN), Adaboost, random forests (RF), and deep neural networks) to increase the detection rate in traditional ML models.

Based on our analysis of existing ML/DL based IDSs works, we found that a number of these schemes [**161**,**162**,**166**] are computationally expensive. Also, most of them [**157**–**168**], are trained in some specific networks, on imbalanced and outdated datasets; this usually results in inefficient and inaccurate intrusion detection models when encountering zero-day attacks. The lack of new, balanced, and up-to-date datasets that contain sufficient data records of new attacks (*i.e.*, zero-day attacks) is still an ongoing challenge making these new attacks successful. The privacy nature of these datasets as well as the widespread emergence of adversarial attacks prevent major organizations (*e.g.*, network operators) from sharing their sensitive datasets with research communities to develop efficient and up-to-date ML/DL models.

More recently, attempts have been made to raise the issue of the lack of labeled data. Zhao et al. [**169**] proposed the use of transfer learning approach to detect new attacks by transferring the knowledge of intrusion detection from a source model trained on a private dataset to a target model. Wu et al. [**170**] proposed a knowledge-transfer based Convolution Neural Network (ConvNet) model, called TL-ConvNet, to detect unseen attacks based on a transfer learning approach. TL-ConvNet consists of: (1) a ConvNet architecture that holds the basic knowledge of network intrusions; and (2) a transfer learning approach that possesses the knowledge of the target attack model. However, these solutions may not be efficient to cope with zero-day attacks since they are mainly trained on some specific datasets of a specific user. There is no collaborative training in which users collaboratively build an efficient and up-to-date intrusion detection model that copes with the new emerging attacks. Moreover, these schemes [**169**, **170**] are not privacy preserving since the shared model may reveal the details of the user's private data during inference.

To address the shortcomings of these existing solutions [**157**–**170**], we propose a distributed, efficient, and secure collaborative DDoS mitigation framework; it allows multiple SDN domains to collaboratively train and build a shared global intrusion detection model without sharing their sensitive private data; thus, preserving the privacy of collaborators. Moreover, we introduce a novel Secure Multiparty Computation (SMPC) scheme to securely aggregate local model updates from each collaborator. Finally, we propose a blockchain based scheme to maintain a collaboration that is fully decentralized, trustworthy, flexible, and efficient. The use of new technologies (*i.e.*, FL, SMPC, SDN, blockchain and smart contracts) allows for a secure and efficient DDoS collaboration, making it a promising approach for major organizations to share their intrusion detection knowledge with other communities while preserving their privacy.

5.4. CoFed: An Overview

This section presents an overview of CoFed. More specifically, we explain how CoFed ensures: (1) a distributed collaborative learning between multiple SDN domains to build a joint global model by exchanging only encrypted model updates; and (2) a secure aggregation scheme to securely aggregate the locally computed model updates from each collaborator. Fig. 5.1 shows the system architecture of CoFed. CoFed allows multiple SDN domains to collaborate in a secure manner. First, the organization creates a collaboration smart contract in the blockchain. Then, it adds the authorized collaborators to the smart contract. Therefore, the organization enables the process of collaborative learning with only authenticated collaborators. The use of blockchain will ensure the integrity, transparency, availability, and the reliability of the collaboration system. Once done, the organization can start the training of a global model across these authenticated collaborators. First, the organization initializes

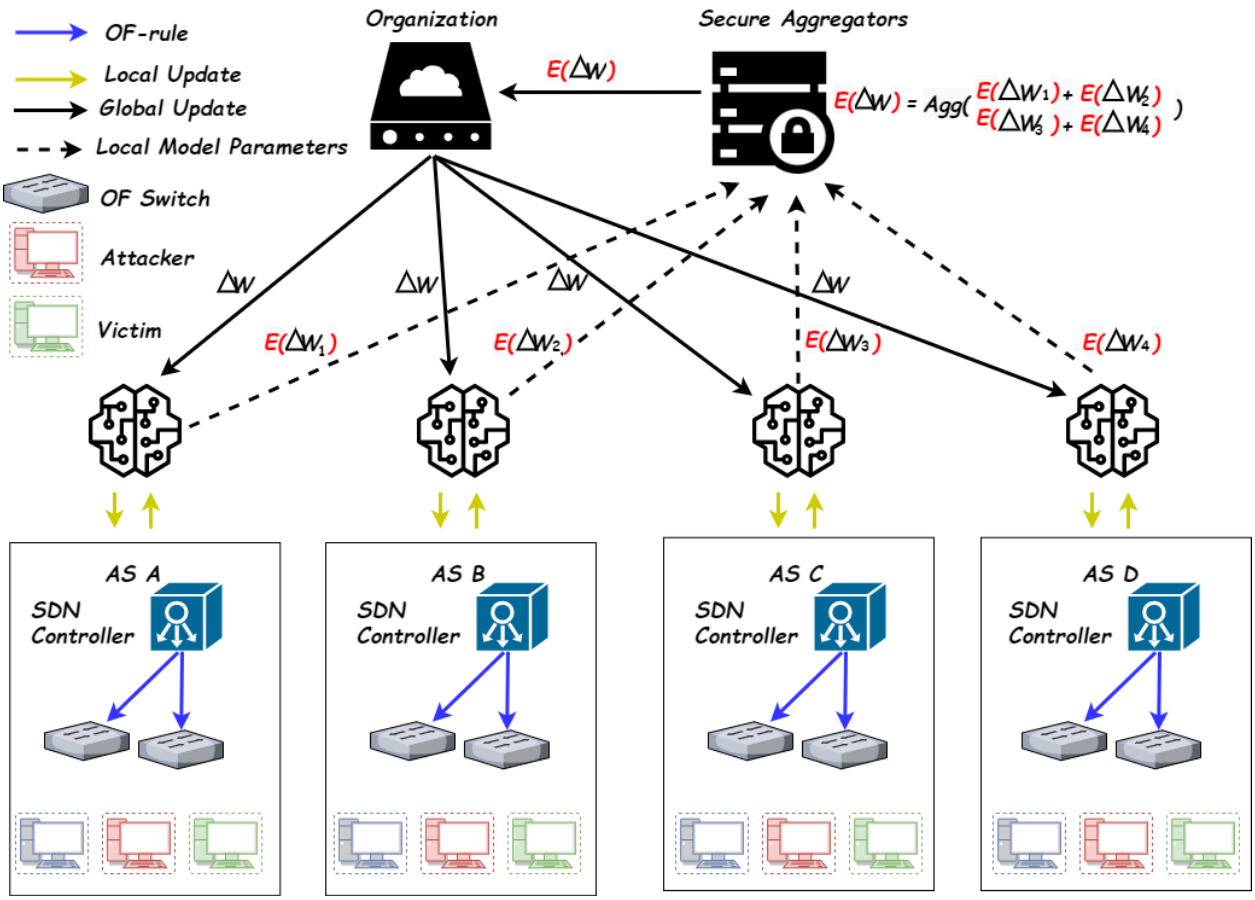


Figure. 5.1. CoFed's System Architecture

the global model with a set of parameters. Then, it sends these initial parameters to each collaborator (*i.e.*, ASs: A, B, C, D see Fig. 5.1). Each collaborator calculates in parallel local model updates based on its local training data, then sends the encrypted values of these updates to secure aggregators, these local updates are private to secure aggregators. These aggregators aggregate local encrypted updates sent from each collaborator and send the aggregated value to the organization. Last, the organization decrypts the aggregated values and sends the global model parameters to each collaborator. This process is repeated until a maximum round or a certain stop criterion is reached. In the following, we describe in more details our proposed framework. Then, we present the implementation as well as the evaluation of our proposed framework.

5.5. Design Of CoFed

5.5.1. CoFed'S Smart Contract

We consider an organization that would like to manage a privacy preserving collaborative learning process across multiple distributed SDN domains. First, the organization creates

and deploys CoFed's smart contract in Ethereum's blockchain [**8**]. Then, it adds, via CoFed's smart contract, FL collaborators (*i.e.*, SDN domains) into the collaboration system. It includes the collaborator's address, an initial credibility and other information (*e.g.*, collaborator notes). CoFed's smart contract provides the organization (*i.e.*, contract owner) with the flexibility to easily add/remove FL collaborators to/from the collaboration system and to manage the collaboration system in a fully decentralized, trustworthy, and transparent manner. CoFed's smart contract is programmed using solidity language [**89**]. For the sake of simplicity, the following is restricted to two important functions of CoFed's smart contract; where *Col* denotes an instance of a collaborator; *now* is the time passed in seconds since 1970; msg.sender is the address of the FL collaborator that sent the transaction to the smart contract; and the Externally Owned Account (EOA): is an account controlled by a pair of public and private keys.

addFLCollaborator(Col.EOA, Col.Infos): this function can only be called by the organization to add a FL collaborator; it takes as input the collaborator's Externally Owned Account (Col.EOA) and other information (Col.Infos) and adds the FL collaborator to the collaboration system as well as the timestamp of when the FL collaborator was added. Algorithm 12 illustrates the logic of this function.

Algorithm 12: addFLCollaborator

```
   Input  : col.EOA, col.Infos
   Output: null
 1 if msg.sender is not owner then
 2 |  throw;
 3 end
 4 if status is not true then
 5 |  throw;
 6 end
 7 if isFLCollaborator(col.EOA) == true then
 8 |  throw;
 9 else
10 |   length ←FLColsAdr.push(col.EOA)
   |   FLCols[col.EOA]←FLCol(col.EOA,col.Infos, credibility, now, length-1)
   |   emit FLColAdded(col.EOA, col.Infos)
   |   numberOfFLCols++
   |
11 end
```

removeFLCollaborator(col.EOA): this function can only be called by the organization to remove a FL collaborator from the collaboration system if needed; it takes as input the EOA of the FL collaborator (col.EOA) and removes him from the collaboration system. Algorithm 13 illustrates the logic of this function.

Algorithm 13: removeFLCollaborator

Input : col.EOA
Output: null

```
if msg.sender is not owner then
    throw;
end
if status is not true then
    throw;
end
if isFLCollaborator(col.EOA) == false then
    throw;
else
    rowToDelete ←FLCols[col.EOA].index
    keyToMove←FLColsAdr[length-1]
    FLColsAdr[rowToDelete]=keyToMove
    FLCols[keyToMove].index=rowToDelete
    FLColsAdr.length --
    emit FLColRemoved(col.EOA)
    numberOfFLCols --
end
```

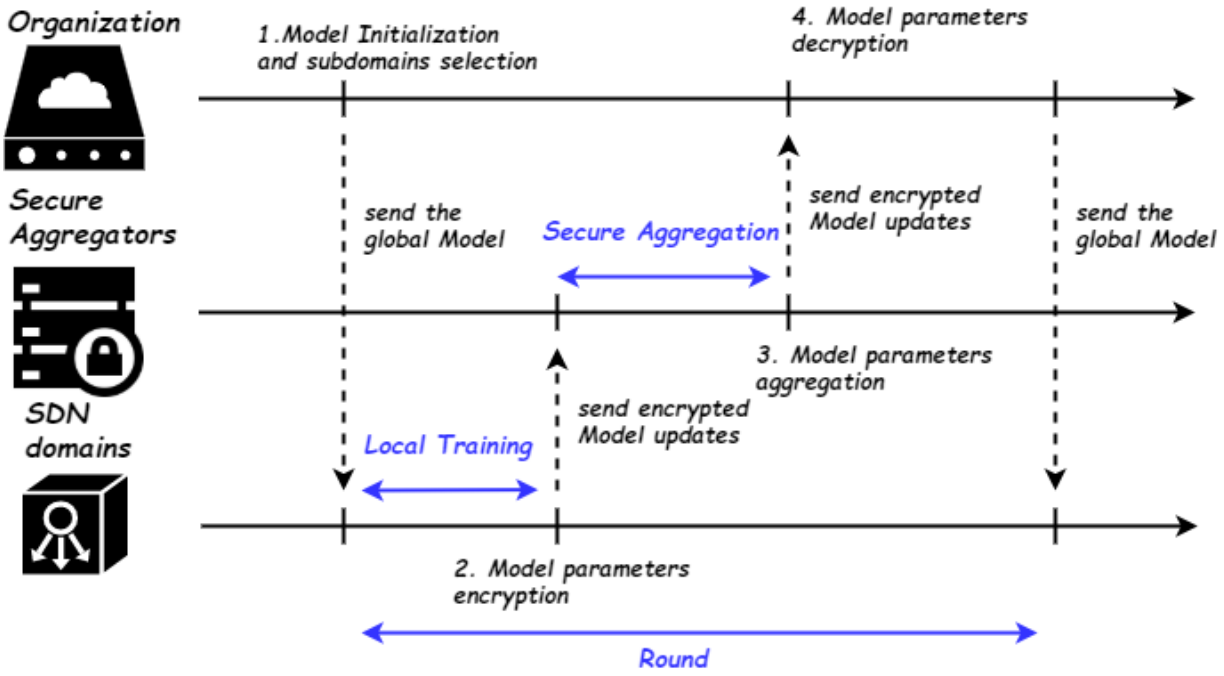


Figure. 5.2. Flowchart of CoFed

Tableau 5.2. Notations

Notations	Definition
W_r	The global model parameters at round r
W_r^k	The local model updates of collaborator k at round r
$W_r^{c,k}$	The encrypted local model updates of collaborator k at round r
$W_{r,s}^{c,k}$	The s^{th} shares of the encrypted model update of collaborator k at round r
η	The learning rate
B	The local batch size
E	The local epoch
K	The number of collaborators
n_k	The number of local training samples of collaborator k
r_{max}	The number of rounds
S_r	The number of secure aggregators at each round r

5.5.2. CoFed Framework

In this section, we describe the operation of CoFed; in particular, we describe how it enables a private learning between SDN domains by combining FL with a secure aggregation scheme. Table 5.2 shows the list of notations used to describe CoFed whereas Fig. 5.2 shows the flowchart that exhibits the interactions between our main components: the organization, secure aggregators and SDN domains (*i.e.*, collaborators): (1) the organization initiates the collaborative learning by initializing the global model parameters with a set of weights W_0. Then, it selects K authenticated collaborators. Afterwards, it sends the global model to each authenticated collaborator; (2) each collaborator k, $1 <= k <= K$ collaborators performs local updates and computes in parallel, based on the global model and its local training dataset, a new local model. Then, each collaborator encrypts its local model updates and split these updates into S_r shares; (3) each secure aggregator s, $1 <= s <= S_r$ receives one of the shares of each collaborator and updates the global model by aggregating the received encrypted local model updates and sends them to the organization; and (4) the organization decrypts the model parameters and starts new round. This process is repeated until a maximum round is reached. The use of SDN in this process allows each collaborator to obtain a global view of its local network to: (1) easily extract network traffic features to train the local model; and (2) easily deploy the global model to effectively and accurately cope with security threats (*e.g.*, zero-day attacks) once the global training across multiple SDN domains is done. We formalize the problem of collaborative learning as a problem of federated optimization as proposed by McMahan et al. [**142**]; in addition, we make use of a security layer to protect the locally-computed updates from reverse-engineering attacks. We define the optimization of the non-convex neural network objective function as follows:

$$\min_{w_r \in R^d} f(w_r) \quad \text{where} \quad f(w_r) = \frac{1}{K} \sum_{k=1}^{K} f_k(w_r) \tag{5.5.1}$$

where K is the number of collaborators and $f_k(w_r)$ is the local objective function for the k^{th} collaborator; f_k is parameterized by a high-dimensional vector $w_r \in \mathrm{R}^d$. Local training consists of finding parameters w_r, at each round r, that minimize the local loss function defined as follows:

$$\forall k, \qquad f_k(w_r) = \frac{1}{n_k} \sum_{j_k=1}^{n_k} f_{j_k}(w_r; x_{j_k}, y_{j_k}) \tag{5.5.2}$$

where n_k is the number of local samples (*i.e.*, (x_{j_k}, y_{j_k})) of the k^{th} collaborator.

For optimization, we use local stochastic gradient descent (SGD) with a fixed learning rate η on each collaborator. At the beginning of each round r, each collaborator independently computes the average gradient on its local data at the current model w_r using local batch or mini-batch for one or more local epochs e:

$$g_r^k = \nabla f_k(W_r; b_r) \tag{5.5.3}$$

Afterwards, each collaborator takes a step of local gradient descent on the current model using its local data as follows:

$$\forall k, \qquad W_r^k \leftarrow W_r - \eta \nabla f_k(W_r; b) \tag{5.5.4}$$

In order to protect these local model updates from reverse-engineering attacks, we use a secure multi-party computation scheme (SMPC). SMPC allows each collaborator k to split its secret (i.e., local model updates) into several shares given to secure aggregators, in such a way that each secure aggregator $s = 1..S_r$ has an encrypted value $w_{r,s}^{c,k}$ and learns nothing about the secret (*i.e.*, w_r^k). This approach is known as secret sharing, a well-know field in in modern cryptography, that provides better performance in terms of computation in comparison with other cryptography primitives (*i.e.*, Homomorphic encryption (HE)).

First, we set a fixed finite field *i.e.*, from $0, 1, ..., Q-1$ for a prime Q in which all computations will take place. Each collaborator k encodes its local model updates w_r^k as integers, by scaling them with a fixed precision constant and computes the modulo of the result with the prime Q. Afterwards, each collaborator k splits its encoded value (*i.e.*, secret) into S_r shares. Each secure aggregator s, $1 <= s <= S_r$, receives one of the shares that represents the encrypted value of the locally computed model update $w_{r,s}^{k,c}$. Then, each secure aggregator s aggregates these encrypted gradients $w_{r,s}^{k,c}$ and applies the following update:

$$\forall s, \qquad W_{r+1,s}^c \leftarrow W_{r,s}^c - \eta \frac{1}{K} \sum_{k=1}^{K} g_{r,s}^k \tag{5.5.5}$$

where $\frac{1}{K}\sum_{k=1}^{K} g_{r,s}^{k}=\nabla f(W_{r,s}^{c}; b_{r,s}^{c})$. Since, for each collaborator k:

$$\forall k, \qquad W_{r+1}^{c,k} \leftarrow W_{r}^{c} - \eta g_{r}^{k} \tag{5.5.6}$$

The final model update for each secure aggregator s becomes as follows:

$$\forall s, \qquad W_{r+1,s}^{c} \leftarrow \frac{1}{K}\sum_{k=1}^{K} W_{r+1,s}^{c,k} \tag{5.5.7}$$

Finally, the organization reconstructs the secret by summing the final model updates from each aggregator s. Then, it decrypts the model parameters and sends the new global model to each authenticated collaborator. This process is repeated until a maximum round r_{max} is reached. Algorithm 14 shows the pseudo-code of the operation of CoFed.

Algorithm 14: CoFed Training

Initial Phase: the organization selects K authenticated collaborators (indexed by k) and initializes the global model parameters with a set of W_0. Then, it sends the global model to each of the authenticated collaborators.
S_r ←subset of secure aggregators at round r
for r ←*1 to* r_{max} **do**
  for *each Secure aggregator s in* S_r **do**
    $W_{r+1,s}^{c}$ ← SecureAggregation(r,s)
  end
  W_{r+1}^{c} ←$\sum_{s=1}^{S_r} W_{r+1,s}^{c}$
  W_{r+1} ←decrypt(decode(W_{r+1}^{c}))
  Send W_{r+1} to each collaborator
end
SecureAggregation(r,s): //Each secure aggregator executes this function
for *each collaborator k* ***in parallel*** **do**
  $W_{r+1,s}^{c,k}$ ← CollaboratorUpdate(k)
end
$W_{r+1,s}^{c}$= $\frac{1}{K}\sum_{k=1}^{K} W_{r+1,s}^{c,k}$
return $W_{r+1,s}^{c}$ to organization
CollaboratorUpdate(k): // Each authenticated collaborator executes this function
B ←Split local datasets into local batches of size B
for *each Local epoch e in E* **do**
  for *batch b in B* **do**
    W_{r}^{k} ←$W_r - \eta\nabla f_k(W_r; b)$
  end
end
$W_{r}^{c,k}$ ←encrypt(encode(W_{r}^{k}))
Split $W_{r}^{c,k}$ into S_r shares
return $W_{r,s}^{k,c}$ to each secure aggregator s

5.6. Implementation

In this section, we present the evaluation of CoFed. First, we introduce the experimental environment. Then, we present the experimental results. Finally, we evaluate the performance of CoFed.

5.6.1. Experimental environment

The implementation of CoFed is done using Pysyft, a generic framework for privacy preserving deep learning, built on top of PyTorch [**171**]. The deployment of CoFed smart contract is done using truffle framework [**92**], a decentralized application development framework. First, we have coded CoFed smart contract using solidity, a high-level language programming. Afterwards, we have compiled CoFed smart contract into Ethereum Virtual Machine byte code. Once compiled, EVM byte code and Application Binary Interface (ABI) were generated. First, we have deployed CoFed smart contract on a private blockchain using Ganache simulator [**93**], an Ethereum simulator used for testing smart contracts. Afterwards, we have deployed it on Ethereum official test network Ropsten [**94**], an open blockchain platform. We run our experiments on Google Colaboratory [**138**] using the Tesla T4 GPU on a PC with Intel Core i7-8750H-2.2 GHz, 16GB RAM, and GTX 1050 GPU.

5.6.2. Experimental Results

The performance of CoFed is evaluated using NSL-KDD dataset; Table 5.3 shows some of the samples of this dataset. The 41 values of each sample corresponds to features described in Tables 5.11 and 5.12. Our aim is to build a global model that categorizes each sample as one of the five classes: normal, DoS, probe, R2L, or U2R. The NSL-KDD dataset covers the train set (*i.e.*, $NSL-KDDTrain^{+}$) and the test set (*i.e.*, $NSL-KDDTest^{+}$) and contains a reasonable number of records in each set. The training dataset contains 22 types of attacks while the test dataset contains 37 types of attacks; these additional new attacks in the test dataset make the detection task more realistic and allow to test if the trained global model can generalize the training data. Table 5.4 shows the total number of records in the train and test datasets and Figs. 5.3(a) and 5.3(b) show, respectively, the NSL-KDD training and testing data class distributions. The NSL-KDD dataset contains 41 features, three of them are categorical/non-numeric (*i.e.*, 'protocol_type', 'service' and 'flag'); 'protocol_type' has three values (*i.e.*, 'tcp', 'udp', and 'icmp') whereas 'service' and 'flag' have, respectively, 70 and 11 distinct values. We encoded 'protocol_type' feature into a numeric form using one hot encoding (*i.e.*, (1,0,0) for 'tcp', (0,1,0) for 'udp', and (0,0,1) for 'icmp'). Similarly, we encoded the two other categorical features (*i.e.*, 'service' and 'flag') into numeric features. Thus, the 41 initial dimensional features are mapped into 122-dimensional features after this transformation. The data distributions of input features of the NSL-KDD dataset vary

Tableau 5.3. Samples of NSL-KDD dataset

Sample content	type
0, tcp, ftp_data, SF, 491, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 0, 0, 0, 0, 1, 0, 0, 150, 25, 0.17, 0.03, 0.17, 0, 0, 0, 0.05, 0	**normal**
0, tcp, private, REJ, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 229, 10, 0, 0, 1, 1, 0.04, 0.06, 0, 255, 10, 0.04, 0.06, 0, 0, 0, 0, 1, 1, neptune	**DoS**
1, tcp, telnet, RSTO, 0, 15, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 8, 0.00, 0.12, 1.00, 0.50, 1.00, 0.00, 0.75, 29, 86, 0.31, 0.17, 0.03, 0.02, 0.00, 0.00, 0.83, 0.71, mscan	**Probe**
0, tcp, telnet, SF, 129, 174, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1.00, 0, 0, 255, 255, 1, 0, 0, 0, 0.01, 0.01, 0.02, 0.02, guess_passwd	**R2L**
8, tcp, ftp, SF, 220, 688, 0, 0, 0, 4, 0, 1, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 53, 27, 0.51, 0.08, 0.02, 0, 0, 0, 0, 0, buffer_overflow	**U2R**

Tableau 5.4. Details of NSL-KDD dataset

Dataset	Records	Normal	DoS	Probe	R2L	U2R
NSL-KDDTrain	125973	67343	45927	11656	995	52
NSL-KDDTest	22544	9711	7558	2421	2754	200

widely. Some of the features (*e.g.*, 'src_bytes [0,1.37*10^9]' and 'dst_bytes [0,1.3*10^9]') of the NSL-KDD dataset have larger values than others (*e.g.*, 'num_failed_logins [0,5]' and 'num_compromised [0,7479]') while some other features (*e.g.*, 'duration [0,42908]') have large ranges between the maximum and the minimum values; this will impact the results as the global model will miss out important information in the features that have minimum values such as 'num_failed_logins'. Hence, we re-scale the values of the features values according to Eq. (5.6.1) using a standardization technique to have a mean of 0 and a standard deviation of 1 (unit variance). As the last step in this preprocessing phase, the output feature ('Label'), that contains name of attack, is changed to numerical values (see Table 5.5) to consider both binary and multi-class classification.

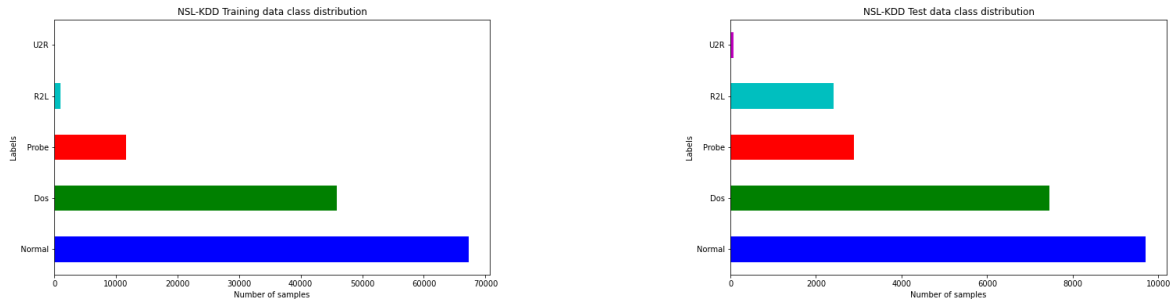
$$X'_i = \frac{X_i - Mean(X_i)}{stdev(X_i)} \tag{5.6.1}$$

where X_i denotes the input feature (*e.g.*, duration), $Mean(X_i)$ and $stdev(X_i)$ denote, respectively, the mean and standard deviation values for each input feature.

Our aim in this work is to test the effectiveness and feasibility of CoFed; thus, we constructed a global deep neural network with input layer of 122 neurons that corresponds to the dimension of the input sample of NSL-KDD dataset, two hidden layers with LeakyReLU [**172**] (Leaky Rectified Linear Unit) as activation function and an output layer of 5 dimensions in the case of multi-class classification and 2 dimensions in the case of binary classification

Tableau 5.5. NSL-KDD labels

Label	Multi-class	Binary
Normal	0	0
DoS	1	1
Probe	2	1
R2L	3	1
U2R	4	1



((a)) **((b))**

Figure. 5.3. NSL-KDD training and testing data class distribution

(see Table 5.5). Dropout [**173**] is used as regularization technique to prevent overfitting, cross entropy loss function $\mathcal{L}$ as in Eq. (5.6.2) is used as loss function; Adam (Adaptive Moment Estimation) [**174**], an algorithm for first-order gradient-based optimization, is used as optimizer to minimize the loss function $\mathcal{L}$.

$$\mathcal{L} = -\frac{1}{N}\sum_{j=1}^{N} y_j * log(\hat{y}_j) \tag{5.6.2}$$

where y_j is the ground truth vector for j^{th} class, $\hat{y}_j$ is the predicted value for j^{th} class, and N is the number of samples in the batch.

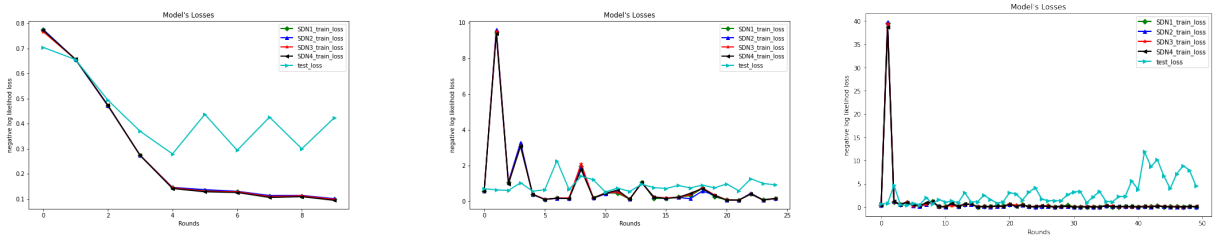
To test CoFed, we defined two different scenarios while considering both multi-class classification and binary classification in each scenario: (1) **Scenario 1:** we divide the NSL-KDD train set between collaborators in such a way that each collaborator will have a subset of each attack type; and (2) **Scenario 2:** we divide the NSL-KDD train set between collaborators in such a way that each collaborator will have a type of attack; in this second scenario, we aim to test the effectiveness of our proposed solution when the distribution of classes is not uniform *i.e.*, imbalanced dataset (see Fig. 5.3). In both scenarios, we randomly initialize global model parameters. Then, we send the global model to each of the four collaborators (*i.e.*, SDN1, SDN2, SDN3, and SDN4 see Fig. 5.1). Each collaborator trains the model locally using his local data. Then, we test the global shared model on

Tableau 5.6. Used scenarios

Scenarios	Type	Rounds	Local epochs
1	Binary	10,25,50	1-5
	Multi-class	10,25,50	1-5
2	Binary	10,25,50	1-5
	Multi-class	10,25,50	1-5

Tableau 5.7. Confusion matrix

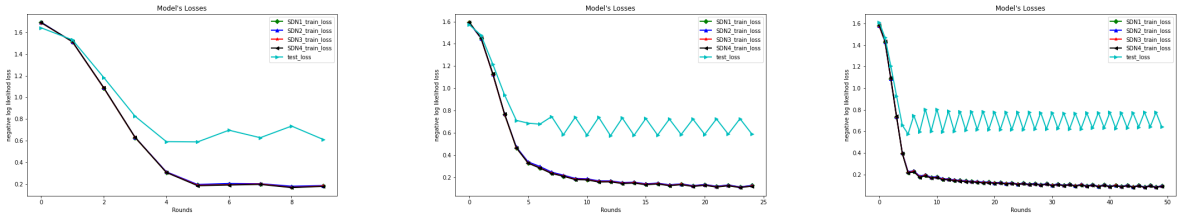
	Classified as illegitimate	Classified as legitimate
illegitimate flows	TP (True Positives)	FN (False Negatives)
legitimate flows	FP (False Positives)	TN (True Negatives)



((a)) ((b)) ((c))

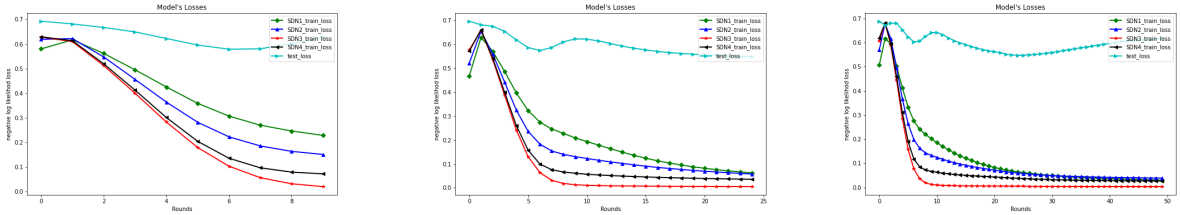
Figure. 5.4. Model loss for scenario 1 binary classification for: a) 10 rounds; b) 25 rounds; and c) 50 rounds.

the test set (*i.e.*, $NSL - KDDTest^{+}$). In each scenario, we varied the number of rounds from 10 to 50 and the number of local epochs from 1 to 5 in both multi-class and binary classification. The scenarios, we evaluated, are summarized in Table 5.6. Figs. 5.4, 5.5, 5.6, and 5.7 show the learning curves of the tested models over rounds; they show the negative log likelihood loss values for scenario 1 and scenario 2, respectively, during training and testing phases for both binary and multi-class classification. Figs. 5.4(a), 5.4(b) and 5.4(c) show the loss of models of scenario 1 in binary classification for 10 rounds, 25 rounds, and 50 rounds, respectively. Figs. 5.5(a), 5.5(b) and 5.5(c) show the model loss for scenario 1 in multi-class classification for 10 rounds, 25 rounds, and 50 rounds, respectively. We observe that the loss, in both scenarios, decreases until it reaches the minimum; the loss of each collaborator's model is almost zero which indicates that models are capable of learning from each other without sharing their local data; the test loss also decreases to a point of stability. Similarly, Figs. 5.6 and 5.7 show the model loss for scenario 2 in binary classification and multi-class classification, respectively. We observe that the training and testing losses decrease to a point of stability. In the following section, we study the performance of CoFed, and we compare our results with state-of-the-art centralised ML/DL models.



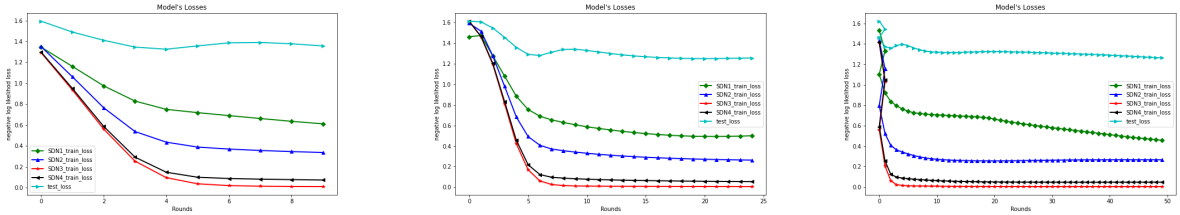
((a)) ((b)) ((c))

Figure. 5.5. Model loss for scenario 1 Multi-class classification for: a) 10 rounds; b) 25 rounds; and c) 50 rounds.



((a)) ((b)) ((c))

Figure. 5.6. Model loss for scenario 2 binary classification for: a) 10 rounds; b) 25 rounds; and c) 50 rounds.

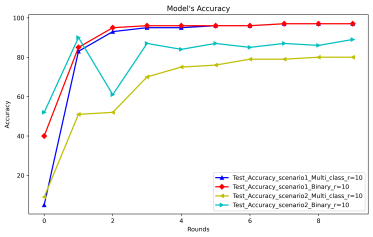


((a)) ((b)) ((c))

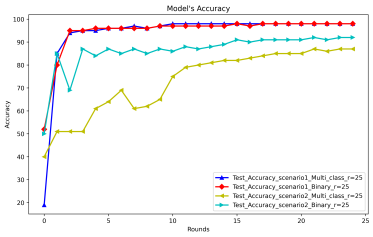
Figure. 5.7. Model loss for scenario 2 Multi-class classification for: a) 10 rounds; b) 25 rounds; and c) 50 rounds.

5.6.3. Performance Evaluation

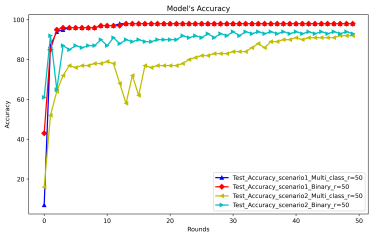
We evaluate the performance of CoFed in terms of Accuracy, Precision, Recall called detection rate (DR) or TPR (True Positive Rate), and F1-score. The performance of CoFed is measured using the area under the ROC Curve (AUC). The ROC curve shows TPR according FPR. Finally, a confusion matrix is used to show the complete performance of CoFed (see Table 5.7). Accuracy is defined in Eq. 5.6.3 as the percentage of the number of correct predictions over the total number of predictions.



((a))

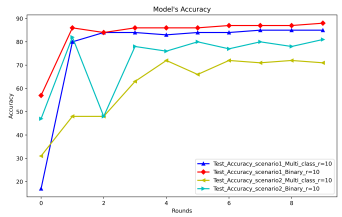


((b))

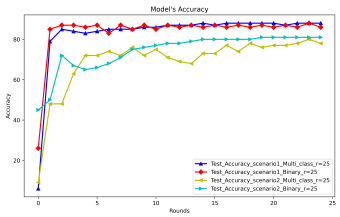


((c))

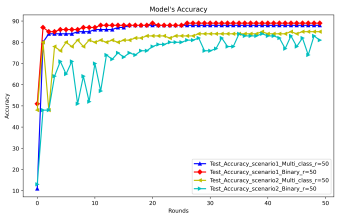
Figure. 5.8. Accuracies on the $NSL-KDDTrain^{+}$ dataset for: a) 10 rounds; b) 25 rounds; and c) 50 rounds.



((a))



((b))



((c))

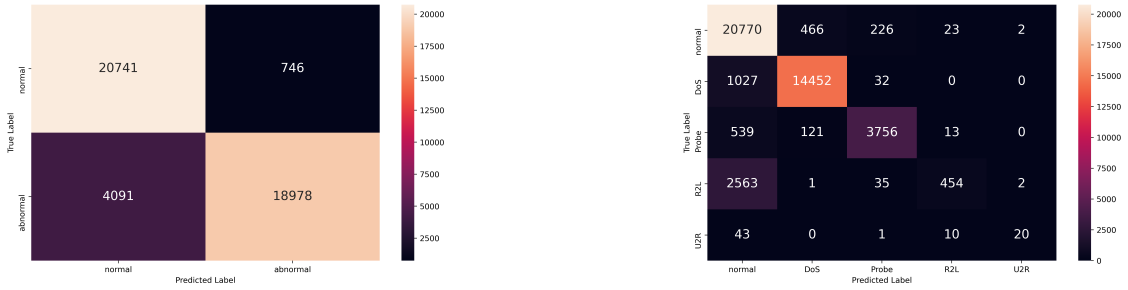
Figure. 5.9. Accuracies on the $NSL-KDDTest^{+}$ dataset for: a) 10 rounds; b) 25 rounds; and c) 50 rounds.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.6.3}$$

When the class distribution is imbalanced, Accuracy may not be a good indicator. Therefore, we need to evaluate specific class performance metrics. Precision, one of such metrics, is defined in Eq. 5.6.4; it is the percentage of the number of correct predictions of intrusions over the total number of predicted intrusions.

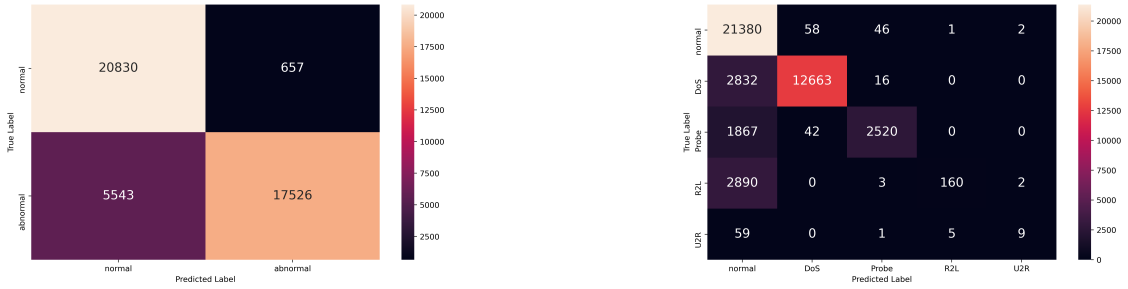
$$Precision = \frac{TP}{TP + FP} \tag{5.6.4}$$

Recall, called detection rate (DR) or TPR (True Positive Rate) is another important metric, that is defined in Eq. 5.6.5; it is the percentage of the number of correct predictions



((a)) ((b))

Figure. 5.10. Confusion matrices for scenario 1 on $NSL-KDDTest^{+}$ in: a) binary classification; and b) multi-class classification.



((a)) ((b))

Figure. 5.11. Confusion matrices for scenario 2 on $NSL-KDDTest^{+}$ in: a) binary classification; and b) multi-class classification.

of intrusions over the total number of presented intrusions.

$$Recall = DR = TPR = \frac{TP}{TP + FN} \tag{5.6.5}$$

It is often convenient to combine precision and recall into a single metric called F1 score. F1 is the harmonic mean of the precision and recall; it is defined as follows:

$$F1 = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{5.6.6}$$

False positive rate (FPR) is defined in Eq. 5.6.7; it is the percentage of the number of intrusions incorrectly classified as normal input samples over the total number of negative samples (*i.e.*, normal data).

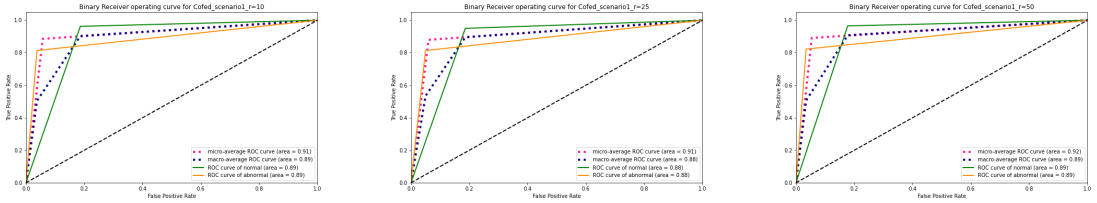
$$FPR = \frac{FP}{TN + FP} \tag{5.6.7}$$

Tableau 5.8. Performance metrics for different scenarios on $NSL-KDDTest^{+}$ dataset

Scenarios	Accuracy	Precision	Recall	F1	Time
Sc1_binary_r=10	**0.89**	**0.90**	**0.89**	**0.89**	**60,48**
Sc1_binary_r=25	0.89	0.90	0.89	0.89	120,02
Sc1_binary_r=50	0.89	0.90	0.89	0.89	290,5
Sc1_multi_r=10	0.86	0.86	0.86	0.84	69,39
Sc1_multi_r=25	0.88	0.88	0.88	0.86	172.34
Sc1_multi_r=50	**0.89**	**0.89**	**0.89**	**0.87**	**349.99**
Sc2_binary_r=10	0.80	0.86	0.80	0.80	185.23
Sc2_binary_r=25	0.82	0.85	0.82	0.81	371.65
Sc2_binary_r=50	**0.86**	**0.88**	**0.86**	**0.86**	**626.79**
Sc2_multi_r=10	0.72	0.73	0.72	0.66	64.93
Sc2_multi_r=25	0.81	0.86	0.81	0.78	344.75
Sc2_multi_r=50	**0.83**	**0.86**	**0.83**	**0.80**	**639.56**

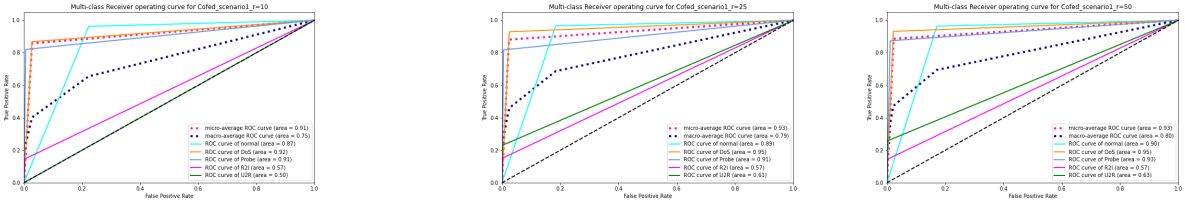
where TP (True Positives) represent intrusions that are correctly identified as intrusions, FN (False Negatives) represent intrusions that are classified as legitimate samples, FP (False Positives) represent legitimate samples that are identified as intrusions, and TN (True Negatives) represent legitimate samples that are classified as legitimate. A good model is a model that has high accuracy, detection rate and F1 score with a low false positive rate. We evaluate the performance of CoFed in both binary and multi-class classification using $NSL-KDDTrain^{+}$ and $NSL-KDDTest^{+}$ datasets. When training the global model, we tried to minimize the loss (see Figs. 5.4, 5.5, 5.6, and 5.7) and maximize the accuracy.

Figs. 5.8 and 5.9 show Accuracy on $NSL-KDDTrain^{+}$ and $NSL-KDDTest^{+}$ datasets, respectively, for 10 rounds, 25 rounds, and 50 rounds. We observe that the global model achieves an accuracy of 97%, 98%, and 99% on the $NSL-KDDTrain^{+}$ dataset for 10 rounds, 25 rounds, and 50 rounds, respectively; it also achieves an accuracy of 85%, 88%, and 89% on the $NSL-KDDTest^{+}$ dataset for 10 rounds, 25 rounds, and 50 rounds, respectively. Additionally, we used the precision, recall, and F1 score metrics to evaluate CoFed. Table 5.8 shows detailed performance of the global model of each evaluated scenario on the $NSL-KDDTest^{+}$ dataset. In addition, Figs. 5.10 and 5.11 show the confusion matrices on the $NSL-KDDTest^{+}$ dataset for scenario 1 and scenario 2, respectively. In scenario 1, CoFed achieves 89%, 90%, 89%, 89% in accuracy, precision, recall, and F1 score, respectively, for only 10 rounds of training (60,48 seconds) in binary classification. For multi-class classification, CoFed achieves 89%, 89%, 89%, 87% in accuracy, precision, recall, and F1 score, respectively, for 50 rounds of training (349,99 seconds). In scenario 2, CoFed achieves 86%, 88%, 86%, 86% in accuracy, precision, recall, and F1 score, respectively, for 50 rounds of training (629,79 seconds) in binary classification. For multi-class classification, CoFed achieves 83%, 86%, 83%, 80% in accuracy, precision, recall, and F1 score, respectively, for 50 rounds of training (639,56 seconds). The ROC curves show the True Positive Rate according



((a)) ((b)) ((c))

Figure. 5.12. ROC Curves of scenario 1 Binary classification on the $NSL-KDDTest^{+}$ dataset for: a) 10 rounds; b) 25 rounds; and c) 50 rounds.



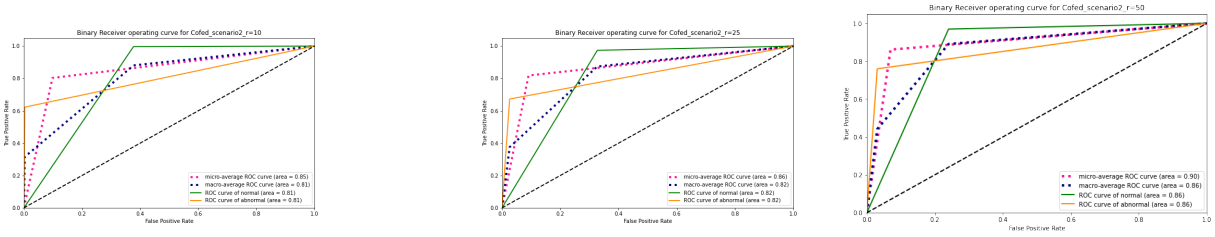
((a)) ((b)) ((c))

Figure. 5.13. ROC Curves of scenario 1 Multi-class classification on the $NSL-KDDTest^{+}$ dataset for: a) 10 rounds; b) 25 rounds; and c) 50 rounds.

to the False Positive Rate. The area under the ROC Curve (AUC) is a measurement metric that measures the degree of separability between the classes. The higher the AUC (*i.e.*, value near to 1), the better the model is in distinguishing between normal and abnormal samples. Figs. 5.12 and 5.13 show the ROC curves of scenario 1 in binary and multi-class classification, respectively, on the $NSL-KDDTest^{+}$ dataset. Figs. 5.14 and 5.15 show ROC curves of scenario 2 in binary and multi-class classification, respectively, on the $NSL-KDDTest^{+}$ dataset. In scenario 1, we obtain an AUC of 0.91, 0,91, and 0.92 in 10 rounds, 25 rounds, and 50 rounds, respectively, in binary classification. For multi-class classification, we obtain an AUC of 0.91, 0,93, and 0.93 in 10 rounds, 25 rounds, and 50 rounds, respectively. In scenario 2, we obtain an AUC of 0.85, 0,86, and 0.90 in 10 rounds, 25 rounds, and 50 rounds, respectively, in binary classification. For multi-class classification, we obtain an AUC of 0.83, 0,88, and 0.89 in 10 rounds, 25 rounds, and 50 rounds, respectively. The experiment results show that CoFed achieves promising results in both binary and multi-class classification in both scenarios while preserving the privacy of each collaborator. In the following, we compare our results with baseline ML/DL models in centralized training architecture.

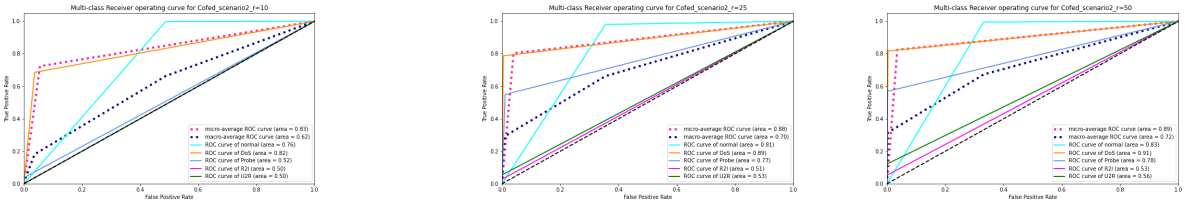
5.6.4. Comparative Analysis

In this section, we compare the results obtained by CoFed with the centralized ML schemes in both binary and multi-class classification using the $NSL-KDDTest^{+}$ dataset.



((a)) ((b)) ((c))

Figure. 5.14. ROC Curves of scenario 2 Binary classification on the $NSL-KDDTest^{+}$ dataset for: a) 10 rounds; b) 25 rounds; and c) 50 rounds.



((a)) ((b)) ((c))

Figure. 5.15. ROC Curves of scenario 2 Multi-class classification on the $NSL-KDDTest^{+}$ dataset for: a) 10 rounds; b) 25 rounds; and c) 50 rounds.

Tableau 5.9. Performance metrics of CoFed and the centralized ML/DL models in binary classification on $NSL-KDDTest^{+}$ dataset

Methods	**Accuracy**	**F1 score**	**Time (second)**
DNN	0.75	0.74	-
RNN-IDS	0.83	-	5516
ResNet50	0.79	0.79	-
GoogleNet	0.77	0.76	-
CharCNN-IDS	0.85	0.86	-
STL-IDS	0.84	0.85	673,031
CoFed	**0.89**	**0,89**	**60,48**

Tableau 5.10. Performance metrics of CoFed and the centralized ML/DL models in multi-class classification on $NSL-KDDTest^{+}$ dataset

Methods	**Accuracy**	**F1 score**	**Time (second)**
DNN	0.75	0.74	-
RNN-IDS	0.81	-	11444
ResNet50	0.79	-	-
GoogleNet	0.77	-	-
CharCNN-IDS	0.79	-	-
STL-IDS	0.8	0.79	61362.384
CoFed	**0.89**	**0,89**	**349,99**

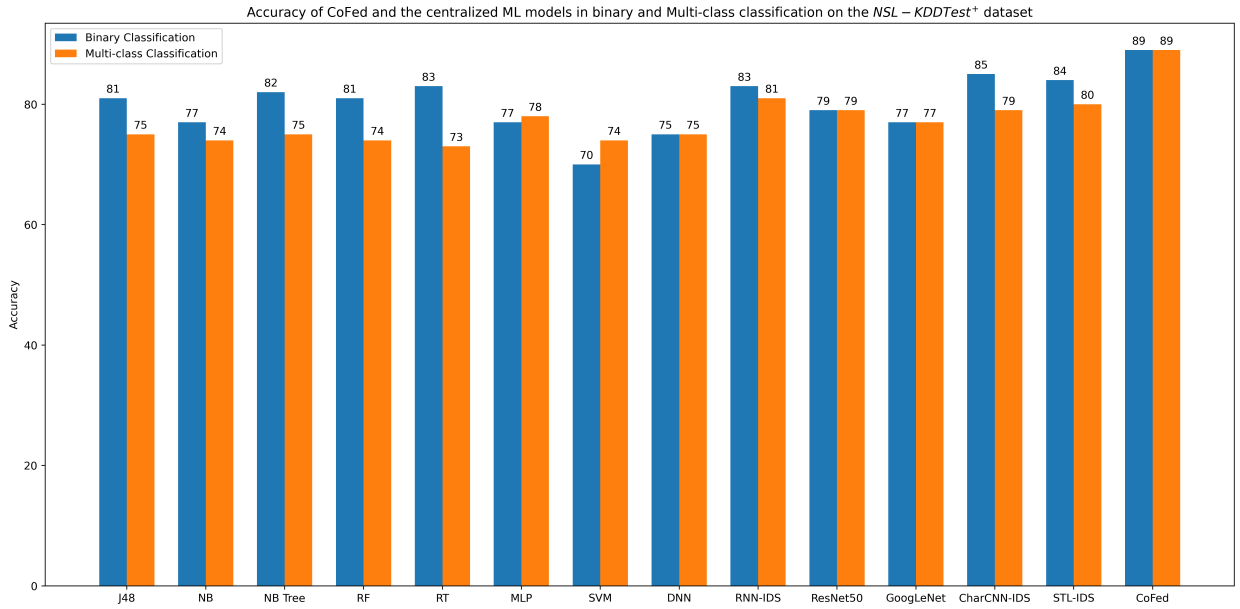


Figure. 5.16. Accuracy of CoFed and the centralized ML/DL models in binary and multi-class classification on the $NSL-KDDTest^{+}$ dataset.

Tavallaee et al. [**139**] measured the accuracy of the following shallow ML models: J48, Naive Bayes (NB), NB Tree, Random Forest (RF), Multi-layer Perceptron (MLP), Random Tree (RT), and Support Vector Machine (SVM) algorithms. We also compare our proposal with the following recent DL-based intrusion detection approaches in the literature: Deep Neural Networks (DNN) [**175**], recurrent neural networks (RNN-IDS) [**163**], convolutional neural networks (ResNet50, GoogLeNet) [**140**], Character-level Intrusion based on convolutional neural networks (CharCNN-IDS) [**176**], and combined Sparse Autoencoder with SVM (STL-IDS) [**165**]. Fig. 5.16 shows the accuracy of CoFed and the centralized ML/DL models in binary and multi-class classification using the $NSL-KDDTest^{+}$ dataset. We observe that in both binary and multi-class classification, CoFed achieves the highest accuracy of 89%. Tables 5.9 and 5.10 show the values of the metrics (*i.e.*, F1 score and training time) of CoFed and centralized ML/DL models in binary and multi-class classification, respectively, using the $NSL-KDDTest^{+}$ dataset. We observe that, in binary classification, CoFed achieves the highest accuracy of 89% and the highest F1 score of 89% with only 60,48 seconds of training time. We also observe that, in multi-class classification, CoFed achieves the highest accuracy of 89% and the highest F1 score of 89% with only 349,99 seconds of training time. The experimental results show that CoFed not only preserves the privacy but also has better detection accuracy and F1 score with smaller training time than centralized ML/DL methods. This makes CoFed a promising cyber-security framework that mitigates the new emerging attacks while preserving the privacy of collaborators.

5.7. conclusion

In this chapter, we proposed a global cyber-security framework, called CoFed, where multiple SDN domains (*i.e.*, collaborators) collaboratively train and build an efficient global intrusion detection model without sharing their private data. The experimental results on the well-known public network security dataset NSL-KDD showed that CoFed achieved efficiency and high accuracy in detecting sophisticated security threats in both binary and multi-class classification, making it a promising framework to mitigate large scale attacks in SDN environment.

5.8. Appendices

Tableau 5.11. List of features in the NSL-KDD dataset

Feature Type	Feature Type	Type	Description
Intrinsic	1. Duration	Continuous	Duration (number of seconds) of the connection
	2. Protocol-type	Categorical	Type of the protocol, *e.g.*, tcp, udp, etc.
	3.Service	Categorical	Network service on the destination, *e.g.*, http, etc.
	4. Src-bytes	Continuous	Number of data bytes from source to destination.
	5. Dst-bytes	Continuous	Number of data bytes from destination to source.
	6. Flag	Categorical	Normal or error status of the connection.
	7. Land	Discrete	1 if connection is from/to the same host/port; 0 otherwise.
	8. Wrong-fragment	Continuous	Number of "wrong" fragments.
	9. Urgent	Continuous	Number of urgent packets.
Content	10. Hot	Continuous	Number of "hot" indicators (*e.g.*, directory accesses).
	11. Num-failed-logins	Continuous	Number of failed login attempts.
	12. Logged-in	Discrete	1 if successfully logged in; 0 otherwise.
	13. Num-compromised	Continuous	Number of "compromised" conditions.
	14. Root-shell	Discrete	1 if root shell is obtained; 0 otherwise.
	15. Su-attempted	Discrete	1 if "su root" command attempted; 0 otherwise.
	16. Num-root	Continuous	Number of "root" accesses.
	17. Num-file-creations	Continuous	Number of file creation operations.
	18. Num-shells	Continuous	Number of shell prompts.
	19. Num-access-files	Continuous	Number of operations on access control files.
	20. Num-outbound-cmds	Continuous	Number of outbound commands in an ftp session.
	21. Is-hot-login	Discrete	1 if the login belongs to the "hot" list; 0 otherwise.
	22. Is-guest-login	Discrete	1 if the login is a "guest" login; 0 otherwise .

Tableau 5.12. List of features in the NSL-KDD dataset

Feature Type	Feature Type	Type	Description
Time-based traffic features	23. Count	Continuous	Number of connections to the same host as the current connection in the past two seconds.
	24. Srv-count	Continuous	Number of connections to the same service as the current connection in the past two seconds.
	25. Serror-rate	Continuous	% of connections that have "SYN" errors (to the same-host).
	26. Rerror-rate	Continuous	% of connections that have "REJ" errors (to the same-host).
	27. Same-srv-rate	Continuous	% of connections to the same service.
	28. Diff-srv-rate	Continuous	% of connections to different service.
	29. Srv-serror-rate	Continuous	% of connections that have "SYN" errors (to the same-service).
	30. Srv-rerror-srate	Continuous	% of connections that have "REJ" errors (to the same-service).
	31. Srv-diff-host-rate	Continuous	% of connections to different hosts.
Host-based traffic features	32. Dst-host-count	Continuous	The Feature Count for destination host.
	33. Dst-host-srv-count	Continuous	The Feature Srv-count for destination host.
	34. Dst-host-same-srv-rate	Continuous	The Feature Same-srv-rate for destination host.
	35. Dst-host-diff-srv-rate	Continuous	The Feature Diff-srv-rate for destination host.
	36.Dst-host-same-src-port-rate	Continuous	The Feature Same-src-port-rate for destination host.
	37. Dst-host-srv-diff-host-rate	Continuous	The Feature Diff-host-rate for destination host.
	38. Dst-host-serror-rate	Continuous	The Feature Serror-rate for destination host.
	39. Dst-host-srv-serror-rate	Continuous	The Feature Srv-serror-rate for destination host.
	40. Dst-host-rerror-rate	Continuous	The Feature Rerror-rate for destination host.
	41. Dst-host-srv-rerror-rate	Continuous	The Feature Srv-serror-rate for destination host.

Chapitre 6

Conclusion

Le projet de cette thèse s'est focalisé sur la mise en place des solutions permettant: (1) la mitigation des attaques par déni de service distribué sur deux niveaux (intra-domaine et inter-domaine); et la (2) le renforcement des aspects sécurité dans les réseaux programmables. Les différentes études que nous avons effectuées ont donné lieu à une compréhension plus complète du projet. Cela nous a permis de bien déterminer les limitations des solutions existants et de démontrer les points forts apportés par nos contributions [**177**–**186**].

6.1. Contributions

Nous avons proposé une première contribution pour détecter et prévenir les attaques d'amplification DNS en se reposant sur le SDN et en considérant tous les cas possibles d'attaques. Nous avons proposé un mécanisme proactif pour filtrer les requêtes et les réponses illégitimes de l'ensemble du trafic DNS. Nous avons proposé un nouveau mécanisme de collecte des statistiques des flux de données afin d'extraire le vecteur représentant l'attaque. Nous avons proposé un mécanisme de détection en temps réel pour détecter automatiquement cette attaque en utilisant des algorithmes d'apprentissage automatique. Nous avons démontré l'efficacité de cette contribution dans un environnement SDN. Nous avons évalué les performances de notre première contribution en termes d'efficacité et de précision. Les résultats des simulations ont montré que notre solution permet d'atténuer efficacement les attaques d'amplification DNS avec une précision élevée et moins de charges sur le plan de contrôle. Puis, nous avons étendu ce système de détection et de mitigation au niveau de l'inter-domaine en utilisant un système de collaboration efficace, flexible, transparent, fiable, et sécurisé. Nous avons implémenté et évalué cette contribution dans un environnement SDN et blockchain afin de prouver sa flexibilité et son efficacité à détecter et mitiger ces attaques. Les résultats des simulations ont montré que notre solution permet d'atténuer efficacement les attaques DDoS avec une précision élevée, un faible taux de faux positifs, et un coût réduit de déploiement.

Par la suite, nous avons étendu la mitigation intra-domaine afin de mitiger d'autres vecteurs d'attaques (R2L, Probe, U2R, etc.). Nous avons amélioré les performances de détection en utilisant des algorithmes d'apprentissage d'ensemble (Ensemble Learning (EL)). Nous avons évalué les performances de notre troisième contribution en utilisant des données d'attaques réelles provenant de deux ensembles de données publiques, à savoir le NSL-KDD et l'UNSW-NB15. Les résultats des simulations ont montré que notre solution permet de mitiger efficacement les attaques avec une précision élevée et un faible taux de faux positifs. En fin, nous avons étendu notre mitigation inter-domaine en considérant des techniques d'apprentissage fédéré. Nous avons proposé une nouvelle architecture distribuée qui permet à plusieurs domaines SDN de collaborer de manière sécurisée. Par la suite, nous avons utilisé un nouveau mécanisme de calcul multipartite sécurisé (Secure multi-party computation(SMPC)) pour agréger de manière sécurisée les mises à jour des modèles de chaque participant. Nous avons intégré les contrats intelligents d'Ethereum pour maintenir la collaboration d'une manière totalement décentralisée, fiable, flexible et efficace. Les résultats des simulations ont montré que notre solution permet de mitiger efficacement les attaques tout en préservant la vie privée des collaborateurs.

6.2. Prochains travaux

Dans nos travaux, nous avons fixé un délai d'inactivité pour les règles OF. Pour des valeurs élevées, les règles OF restent dans la table OF pendant une longue période, ce qui peut épuiser la mémoire des commutateurs OF (TCAM), tandis que des valeurs trop faibles peuvent conduire à l'abandon de réponses légitimes. Les résultats obtenus sont très encourageants pour la suite des travaux. Pour les travaux futurs, nous allons concevoir un nouvel algorithme d'optimisation pour définir dynamiquement ces délais d'attente. Cette optimisation peut être basée sur la capacité de la table OF, le taux de trafic et la charge entre plan de données et du plan de contrôle. Aussi, nous allons utiliser l'apprentissage automatique contradictoire/Réseaux antagonistes génératifs (Adversarial machine learning/Generative adversarial networks) pour renforcer les aspects sécuritaires des systèmes de détection d'intrusion. Par la suite, nous allons etudier l'apport de l'apprentissage non supervisé, ou les données ne sont pas étiquetées, dans la détection d'intrusion. Nous développerons de nouveaux algorithmes d'apprentissage non supervisé afin d'extraire des classes présentant des caractéristiques communes de trafic légitime et par la suite bloquer le trafic de l'attaque tout en préservant un faible taux de faux positifs. D'autres part, nous proposerons de nouvelles mécanismes de collaborations entre un domaine sous l'attaque (domaine d'un client (Customer)) et un domaine (AS domaine) qui peut déployer des services de sécurité à travers le SDN dans ce domaine attaqué. Cette collaboration doit être sécurisée et fiable pour les deux parties.

Références bibliographiques

[1] "Floodlight Controller." [Online]. Available: https://floodlight.atlassian.net/wiki/spaces/HOME/overview

[2] "Opendaylight (ODL) Controller." [Online]. Available: https://www.opendaylight.org/

[3] "Ryu Controller." [Online]. Available: https://ryu.readthedocs.io/en/latest/library.html

[4] "Domaine Name System." [Online]. Available: https://datatracker.ietf.org/doc/html/rfc1035

[5] "Network Time Protocol." [Online]. Available: https://datatracker.ietf.org/doc/html/rfc5905

[6] B. Schneier, "Lessons from the dyn ddos attack." [Online]. Available: https://www.schneier.com/blog/archives/2016/11/lessons_from_th_5.html

[7] "DNS expertise." [Online]. Available: http://dns.measurement-factory.com/surveys/sum1.html

[8] "Ethereum: A secure decentralised generalised transaction ledge." [Online]. Available: https://ethereum.org/

[9] L. Horwitz, "The future of IoT miniguide: The burgeoning iot market continues." [Online]. Available: https://www.cisco.com/c/en/us/solutions/internet-of-things/future-of-iot.html

[10] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'14, 2014.

[11] "NSL-KDD dataset." [Online]. Available: https://www.unb.ca/cic/datasets/nsl.html

[12] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems," in *2015 Military Communications and Information Systems Conference (MilCIS)*, 2015.

[13] C. Rossow, "Amplification hell: Revisiting network protocols for ddos abuse," in *In Proceedings of the 2014 Network and Distributed System Security Symposium, NDSS*, 2014.

[14] S. Sharwood, "Github wobbles under ddos attack." [Online]. Available: https://www.itsecurityguru.org/2015/08/26/github-wobbles-under-ddos-attack/

[15] S. Scott-Hayward, S. Natarajan, and S. Sezer, "A survey of security in software defined networks," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 623–654, Firstquarter 2016.

[16] T. Huang, F. R. Yu, C. Zhang, J. Liu, J. Zhang, and Y. Liu, "A survey on large-scale software defined networking (sdn) testbeds: Approaches and challenges," *IEEE Communications Surveys Tutorials*, vol. 19, no. 2, pp. 891–917, Secondquarter 2017.

[17] D. B. Rawat and S. R. Reddy, "Software defined networking architecture, security and energy efficiency: A survey," *IEEE Communications Surveys Tutorials*, vol. 19, no. 1, pp. 325–346, Firstquarter 2017.

[18] E. Kaljic, A. Maric, P. Njemcevic, and M. Hadzialic, "A survey on data plane flexibility and programmability in software-defined networking," *IEEE Access*, vol. 7, pp. 47 804–47 840, 2019.

[19] R. Mohammadi, R. Javidan, and M. Conti, "Slicots: An sdn-based lightweight countermeasure for tcp syn flooding attacks," *IEEE Transactions on Network and Service Management*, vol. 14, no. 2, pp. 487–497, June 2017.

[20] Z. Wang, H. Hu, and G. Cheng, "Design and implementation of an sdn-enabled dns security framework," *China Communications*, vol. 16, no. 2, pp. 233–245, Feb 2019.

[21] B. Rashidi, C. Fung, and E. Bertino, "A collaborative ddos defence framework using network function virtualization," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 10, pp. 2483–2497, Oct 2017.

[22] B. Rashidi, C. Fung, and M. Rahman, "A scalable and flexible ddos mitigation system using network function virtualization," in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, April 2018, pp. 1–6.

[23] A. Jakaria, B. Rashidi, M. A. Rahman, C. Fung, and W. Yang, "Dynamic ddos defense resource allocation using network function virtualization," in *Proceedings of the ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*, ser. SDN-NFVSec '17. New York, NY, USA: ACM, 2017, pp. 37–42. [Online]. Available: http://doi.acm.org/10.1145/3040992.3041000

[24] T. Dargahi, A. Caponi, M. Ambrosin, G. Bianchi, and M. Conti, "A survey on the security of stateful sdn data planes," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1701–1725, thirdquarter 2017.

[25] A. Abdou, P. C. van Oorschot, and T. Wan, "Comparative analysis of control plane security of sdn and conventional networks," *IEEE Communications Surveys Tutorials*, vol. 20, no. 4, pp. 3542–3559, Fourthquarter 2018.

[26] I. Farris, T. Taleb, Y. Khettab, and J. Song, "A survey on emerging sdn and nfv security mechanisms for iot systems," *IEEE Communications Surveys Tutorials*, vol. 21, no. 1, pp. 812–837, Firstquarter 2019.

[27] J. Zheng, Q. Li, G. Gu, J. Cao, D. K. Y. Yau, and J. Wu, "Realtime ddos defense using cots sdn switches via adaptive correlation analysis," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 7, pp. 1838–1853, July 2018.

[28] L. Fawcett, S. Scott-Hayward, M. Broadbent, A. Wright, and N. Race, "Tennison: A distributed sdn framework for scalable network security," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 12, pp. 2805–2818, Dec 2018.

[29] J. Steadman and S. Scott-Hayward, "Dnsxd: Detecting data exfiltration over dns," in *2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Nov 2018, pp. 1–6.

[30] S. Deng, X. Gao, Z. Lu, and X. Gao, "Packet injection attack and its defense in software-defined networks," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 3, pp. 695–705, March 2018.

[31] "Openvswicth." [Online]. Available: https://www.openvswitch.org/

[32] "Mininet." [Online]. Available: http://mininet.org

[33] D. Huistra, "Detecting reflection attacks in dns flows." [Online]. Available: https://pdfs.semanticscholar.org/4ad8/24537f212f70e25e4cbab55498f5a8e43942.pdf

[34] T. Rozekrans, M. Mekking, and J. de Koning, "Defending against dns reflection amplification attacks," Feb 2013.

[35] C. Sun, B. Liu, and L. Shi, "Efficient and low-cost hardware defense against dns amplification attacks," in *IEEE GLOBECOM 2008 - 2008 IEEE Global Telecommunications Conference*, Nov 2008, pp. 1–5.

[36] F. Guo, J. Chen, and T.-c. Chiueh, "Spoof detection for preventing dos attacks against dns servers," in *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems*, ser. ICDCS '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 37–. [Online]. Available: https://doi.org/10.1109/ICDCS.2006.78

[37] G. Kambourakis, T. Moschos, D. Geneiatakis, and S. Gritzalis, "A fair solution to dns amplification attacks," in *Proceedings of the Second International Workshop on Digital Forensics and Incident Analysis*, ser. WDFIA '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 38–47. [Online]. Available: https://doi.org/10.1109/WDFIA.2007.2

[38] Z. A. El Houda, L. Khoukhi, and A. Hafid, "Chainsecure - a scalable and proactive solution for protecting blockchain applications using SDN," in *2018 IEEE Global Communications Conference (GLOBECOM)*, Dec 2018, pp. 1–6.

[39] Z. A. E. Houda, A. Hafid, and L. Khoukhi, "Brainchain - a machine learning approach for protecting blockchain applications using sdn," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.

[40] R. Braga, E. Mota, and A. Passito, "Lightweight ddos flooding attack detection using nox/openflow," in *IEEE Local Computer Network Conference*, Oct 2010, pp. 408–415.

[41] S. A. Mehdi, J. Khalid, and S. A. Khayam, "Revisiting traffic anomaly detection using software defined networking," in *Recent Advances in Intrusion Detection*, R. Sommer, D. Balzarotti, and G. Maier, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 161–180.

[42] R. Wang, Z. Jia, and L. Ju, "An entropy-based distributed ddos detection mechanism in software-defined networking," in *2015 IEEE Trustcom/BigDataSE/ISPA*, vol. 1, Aug 2015, pp. 310–317.

[43] S. Lim, J. Ha, H. Kim, Y. Kim, and S. Yang, "A sdn-oriented ddos blocking scheme for botnet-based attacks," in *2014 Sixth International Conference on Ubiquitous and Future Networks (ICUFN)*, July 2014, pp. 63–68.

[44] A. Zaalouk, R. Khondoker, R. Marx, and K. Bayarou, "Orchsec: An orchestrator-based architecture for enhancing network-security using network monitoring and sdn control functions," in *2014 IEEE Network Operations and Management Symposium (NOMS)*, May 2014, pp. 1–9.

[45] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, "Combining openflow and sflow for an effective and scalable anomaly detection and mitigation mechanism on sdn environments," *Comput. Netw.*, vol. 62, pp. 122–136, Apr. 2014. [Online]. Available: http://dx.doi.org/10.1016/j.bjp.2013.10.014

[46] J. B. Hong and D. S. Kim, "Assessing the effectiveness of moving target defenses using security models," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 2, pp. 163–177, March 2016.

[47] "sflow-rt." [Online]. Available: http://www.sflow-rt.com

[48] "Rest API." [Online]. Available: http://www.sflowrt.com/reference.php

[49] "Opendaylight." [Online]. Available: https://www.opendaylight.org/

[50] "Floodlight." [Online]. Available: http://www.projectfloodlight.org/

[51] C. E. Shannon, "Prediction and entropy of printed english." Bell system technical journal, 1950.

[52] C. Briat, "Convergence and equivalence results for the jensen's inequality—application to time-delay and sampled-data systems," *IEEE Transactions on Automatic Control*, vol. 56, no. 7, pp. 1660–1665, 2011.

[53] P. D. Hoff, "A First course in Bayesian statistical methods," *Springer*, 2009.

[54] "Openflow switch specification." [Online]. Available: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.3.0.pdf

[55] "Scapy." [Online]. Available: http://www.secdev.org/projects/scapy

[56] "Nodejs." [Online]. Available: https://nodejs.org/en/

[57] "Tcpdump." [Online]. Available: https://www.tcpdump.org/

[58] "Wireshark."

[59] "Iperf." [Online]. Available: https://iperf.fr/

[60] "DDoS attacks against US banks." [Online]. Available: https://www.computerworld.com/article/2493861/ddos-attacks-against-u-s--banks-peaked-at-60-gbps.html

[61] J. Zheng, Q. Li, G. Gu, J. Cao, D. K. Y. Yau, and J. Wu, "Realtime ddos defense using cots sdn switches via adaptive correlation analysis," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 7, pp. 1838–1853, July 2018.

[62] S. Simpson, S. N. Shirazi, A. Marnerides, S. Jouet, D. Pezaros, and D. Hutchison, "An inter-domain collaboration scheme to remedy ddos attacks in computer networks," *IEEE Transactions on Network and Service Management*, vol. 15, no. 3, pp. 879–893, Sep. 2018.

[63] P. Phaal, "Sflow." [Online]. Available: https://www.ietf.org/rfc/rfc3176.txt

[64] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2009. [Online]. Available: http://www.bitcoin.org/bitcoin.pdf

[65] D. Hopwood, S. Bowe, T. Hornby, and N. Wilcox, "Zcash protocol specification," *Zerocoin Electric Coin Company , Oakland, CA, USA, Tech*, 2016.

[66] "Blockchain for Financial Services." [Online]. Available: https://www.ibm.com/blockchain/financial-services

[67] B. Houtan, A. S. Hafid, and D. Makrakis, "A survey on blockchain-based self-sovereign patient identity in healthcare," *IEEE Access*, vol. 8, pp. 90 478–90 494, 2020.

[68] H. Moudoud, S. Cherkaoui, and L. Khoukhi, "An iot blockchain architecture using oracles and smart contracts: the use-case of a food supply chain," in *2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2019.

[69] S. Wang, Y. Zhang, and Y. Zhang, "A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems," *IEEE Access*, vol. 6, pp. 38 437–38 450, 2018.

[70] M. Pärssinen, M. Kotila, R. Cuevas Rumin, A. Phansalkar, and J. Manner, "Is blockchain ready to revolutionize online advertising?" *IEEE Access*, vol. 6, pp. 54 884–54 899, 2018.

[71] Etherscan, "The Ethereum Block Explorer: Ropsten Testnet." [Online]. Available: https://ropsten.etherscan.io

[72] P. Ferguson and D. Senie, "Network ingress filtering: Defeating denial of service attacks which employ ip source address spoofing (bcp 38)." [Online]. Available: http://tools.ietf.org/html/rfc2827

[73] R. Braga, E. Mota, and A. Passito, "Lightweight ddos flooding attack detection using nox/openflow," in *IEEE Local Computer Network Conference*, Oct 2010, pp. 408–415.

[74] S. A. Mehdi, J. Khalid, and S. A. Khayam, "Revisiting traffic anomaly detection using software defined networking," in *Recent Advances in Intrusion Detection*, R. Sommer, D. Balzarotti, and G. Maier, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 161–180.

[75] M. Yu, L. Jose, and R. Miao, "Software defined traffic measurement with opensketch," in *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, ser. nsdi'13. Berkeley, CA, USA: USENIX Association, 2013, pp. 29–42. [Online]. Available: http://dl.acm.org/citation.cfm?id=2482626.2482631

[76] R. Wang, Z. Jia, and L. Ju, "An entropy-based distributed ddos detection mechanism in software-defined networking," in *2015 IEEE Trustcom/BigDataSE/ISPA*, vol. 1, Aug 2015, pp. 310–317.

[77] S. Lim, J. Ha, H. Kim, Y. Kim, and S. Yang, "A sdn-oriented ddos blocking scheme for botnet-based attacks," in *2014 Sixth International Conference on Ubiquitous and Future Networks (ICUFN)*, July 2014, pp. 63–68.

[78] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, "Combining openflow and sflow for an effective and scalable anomaly detection and mitigation mechanism on sdn environments," *Comput. Netw.*, vol. 62, pp. 122–136, Apr. 2014. [Online]. Available: http://dx.doi.org/10.1016/j.bjp.2013.10.014

[79] F. Guo, J. Chen, and T.-c. Chiueh, "Spoof detection for preventing dos attacks against dns servers," in *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems*, ser. ICDCS '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 37–. [Online]. Available: https://doi.org/10.1109/ICDCS.2006.78

[80] K.Nishizuka, L.Xia, J.Xia, D.Zhang, L.Fang, C.Gray, and R. Compton, "Interorganization cooperative DDOS protection mechanism." [Online]. Available: https://tools.ietf.org/html/draft-nishizuka-dots-inter-domain-mechanism-02

[81] J. Steinberger, B. Kuhnert, A. Sperotto, H. Baier, and A. Pras, "Collaborative ddos defense using flow-based security event information," in *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, April 2016, pp. 516–522.

[82] "Total cost of ownership (tco) for PKI - white paper." [Online]. Available: https://www.msctrustgate.com/v1/pdf/TCO%20for%20PKI.pdf

[83] K. Giotis, M. Apostolaki, and V. Maglaris, "A reputation-based collaborative schema for the mitigation of distributed attacks in sdn domains," in *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, April 2016, pp. 495–501.

[84] B. Rashidi, C. Fung, and E. Bertino, "A collaborative ddos defence framework using network function virtualization," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 10, pp. 2483–2497, Oct 2017.

[85] Y. Chen, K. Hwang, and W. Ku., "Collaborative detection of ddos attacks over multiple network domains," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 12, pp. 1649–1662, Dec 2007.

[86] B. Rodrigues, T. Bocek, A. Lareida, D. Hausheer, S. Rafati, and B. Stiller, "A blockchain-based architecture for collaborative ddos mitigation with smart contracts," in *Security of Networks and Services in an All-Connected World*, D. Tuncer, R. Koch, R. Badonnel, and B. Stiller, Eds. Cham: Springer International Publishing, 2017.

[87] M. Steichen, S. Hommes, and R. State, "Chainguard- a firewall for blockchain applications using sdn with openflow," in *2017 Principles, Systems and Applications of IP Telecommunications (IPTComm)*, Sep. 2017, pp. 1–8.

[88] "Go Ethereum." [Online]. Available: https://geth.ethereum.org/

[89] "Solidity." [Online]. Available: https://solidity.readthedocs.io/en/develop/

[90] "Virtualbox." [Online]. Available: https://www.virtualbox.org/

[91] "Hping3." [Online]. Available: https://tools.kali.org/information-gathering/hping3

[92] "Truffle." [Online]. Available: https://truffleframework.com/

[93] "Ganache." [Online]. Available: https://truffleframework.com/docs/ganache/overview

[94] "Ropsten." [Online]. Available: https://ropsten.etherscan.io/

[95] "Eosio: The most powerful infrastructure for decentralized applications." [Online]. Available: https://eos.io/

[96] S. Morgan, "Global ransomware damage costs predicted to reach $20 billion (usd) by 2021." [Online]. Available: https://cybersecurityventures.com/

[97] B. V. Dasarathy and B. V. Sheela, "A composite classifier system design: Concepts and methodology," *Proceedings of the IEEE*, vol. 67, no. 5, pp. 708–713, 1979.

[98] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 10, pp. 993–1001, 1990.

[99] R. E. Schapire, "The strength of weak learnability," in *Mach Learn 5.* Springer International Publishing, 1990, p. 197–227.

[100] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608005800231

[101] H. Drucker and P. Simard, "Boosting performance in neural networks," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 7, no. 4, p. 705–719, 1993.

[102] L. Breiman, "Bagging predictors," *Mach Learn 24*, p. 123–140, 1996.

[103] P. V. R. Ferreira, R. Paffenroth, A. M. Wyglinski, T. M. Hackett, S. G. Bilén, R. C. Reinhart, and D. J. Mortensen, "Multiobjective reinforcement learning for cognitive satellite communications using deep neural network ensembles," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 5, pp. 1030–1041, 2018.

[104] T. Hussain, S. M. Siniscalchi, H. L. S. Wang, Y. Tsao, V. M. Salerno, and W. H. Liao, "Ensemble hierarchical extreme learning machine for speech dereverberation," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 12, no. 4, pp. 744–758, 2020.

[105] Y. Bian, Y. Wang, Y. Yao, and H. Chen, "Ensemble pruning based on objection maximization with a general distributed framework," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 9, pp. 3766–3774, 2020.

[106] J. Yang and F. Wang, "Auto-ensemble: An adaptive learning rate scheduling based deep learning model ensembling," *IEEE Access*, vol. 8, pp. 217 499–217 509, 2020.

[107] Z. Zhu, Z. Wang, D. Li, Y. Zhu, and W. Du, "Geometric structural ensemble learning for imbalanced problems," *IEEE Transactions on Cybernetics*, vol. 50, no. 4, pp. 1617–1629, 2020.

[108] R. Polikar, "Ensemble based systems in decision making," *IEEE Circuits and Systems Magazine*, vol. 6, no. 3, pp. 21–45, 2006.

[109] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 Military Communications and Information Systems Conference (MilCIS)*, 2015, pp. 1–6.

[110] N. Moustafa, "Unsw-nb15." [Online]. Available: www.unsw.adfa.edu.au/australian-centre-for-cyber-security/cybersecurity/ADFA-NB15-Datasets

[111] S. Z. Lin, Y. Shi, and Z. Xue, "Character-level intrusion detection based on convolutional neural networks," in *2018 International Joint Conference on Neural Networks (IJCNN)*, 2018, pp. 1–8.

[112] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," in *2016 International Conference on Wireless Networks and Mobile Communications (WINCOM)*, 2016, pp. 258–263.

[113] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21 954–21 961, 2017.

[114] K. Wu, Z. Chen, and W. Li, "A novel intrusion detection model for a massive network using convolutional neural networks," *IEEE Access*, vol. 6, pp. 50 850–50 859, 2018.

[115] M. A. Ambusaidi, X. He, P. Nanda, and Z. Tan, "Building an intrusion detection system using a filter-based feature selection algorithm," *IEEE Transactions on Computers*, vol. 65, no. 10, pp. 2986–2998, 2016.

[116] R. A. R. Ashfaq, X.-Z. Wang, J. Z. Huang, H. Abbas, and Y.-L. He, "Fuzziness based semi-supervised learning approach for intrusion detection system," *Information Sciences*, vol. 378, pp. 484–497, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025516302547

[117] K. Singh, S. C. Guntuku, A. Thakur, and C. Hota, "Big data analytics framework for peer-to-peer botnet detection using random forests," *Information Sciences*, vol. 278, pp. 488–497, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025514003570

[118] M. Al-Qatf, Y. Lasheng, M. Al-Habib, and K. Al-Sabahi, "Deep learning approach combining sparse autoencoder with svm for network intrusion detection," *IEEE Access*, vol. 6, pp. 52 843–52 856, 2018.

[119] S. A. Ludwig, "Intrusion detection of multiple attack classes using a deep neural net ensemble," in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2017, pp. 1–7.

[120] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, 2018.

[121] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, and M. Zhu, "Hast-ids: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection," *IEEE Access*, vol. 6, pp. 1792–1806, 2018.

[122] X. Gao, C. Shan, C. Hu, Z. Niu, and Z. Liu, "An adaptive ensemble machine learning model for intrusion detection," *IEEE Access*, vol. 7, pp. 82 512–82 521, 2019.

[123] I. Ahmad, M. Basheri, M. J. Iqbal, and A. Rahim, "Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection," *IEEE Access*, vol. 6, pp. 33 789–33 795, 2018.

[124] C. D. McDermott and A. Petrovski, "Investigation of computational intelligence techniques for intrusion detection in wireless sensor networks." *International journal of computer networks and communications*, vol. 9, pp. 45–56, 2017. [Online]. Available: http://hdl.handle.net/10059/2526

[125] N. Moustafa, G. Misra, and J. Slay, "Generalized outlier gaussian mixture technique based on automated association features for simulating and detecting web application attacks," *IEEE Transactions on Sustainable Computing*, pp. 1–1, 2018.

[126] N. Moustafa, E. Adi, B. Turnbull, and J. Hu, "A new threat intelligence scheme for safeguarding industry 4.0 systems," *IEEE Access*, vol. 6, pp. 32 910–32 924, 2018.

[127] N. Ravi, S. M. Shalinie, and D. Danyson Jose Theres, "Balance: Link flooding attack detection and mitigation via hybrid-sdn," *IEEE Transactions on Network and Service Management*, vol. 17, no. 3, pp. 1715–1729, 2020.

[128] A. Ali and M. M. Yousaf, "Novel three-tier intrusion detection and prevention system in software defined network," *IEEE Access*, vol. 8, pp. 109 662–109 676, 2020.

[129] Z. Chen, F. Jiang, Y. Cheng, X. Gu, W. Liu, and J. Peng, "Xgboost classifier for ddos attack detection and analysis in sdn-based cloud," in *2018 IEEE International Conference on Big Data and Smart Computing (BigComp)*, 2018, pp. 251–256.

[130] D. He, S. Chan, X. Ni, and M. Guizani, "Software-defined-networking-enabled traffic anomaly detection and mitigation," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1890–1898, 2017.

[131] M. Bjorklund, J. Schoenwaelder, and A. Bierman, "Network Configuration Protocol (NETCONF)," Internet Requests for Comments, RFC Editor, RFC 6241, June 2011. [Online]. Available: https://tools.ietf.org/html/rfc6241

[132] "Traffic monitoring using sflow." [Online]. Available: https://sflow.org/sFlowOverview.pdf

[133] J. H. Friedman, "Greedy function approximation: A gradient boostingmachine." *The Annals of Statistics*, vol. 29, no. 5, pp. 1189 – 1232, 2001. [Online]. Available: https://doi.org/10.1214/aos/1013203451

[134] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, no. null, p. 281–305, Feb. 2012.

[135] "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, no. 85, p. 2825–2830, 2011. [Online]. Available: https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf

[136] "Openflow switch." [Online]. Available: https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf

[137] "Kdd cup 1999 dataset." [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/

[138] "Google colaboratory." [Online]. Available: https://colab.research.google.com/

[139] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009, pp. 1–6.

[140] Z. Li, Z. Qin, K. Huang, X. Yang, and S. Ye, "Intrusion detection using convolutional neural networks for representation learning," in *Neural Information Processing*. Springer International Publishing, 2017, pp. 858–866.

[141] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[142] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*, 2017.

[143] "Kaspersky." [Online]. Available: https://usa.kaspersky.com/

[144] "Mcafee." [Online]. Available: https://www.mcafee.com/

[145] B. Brik, A. Ksentini, and M. Bouaziz, "Federated learning for uavs-enabled wireless networks: Use cases, challenges, and open problems," *IEEE Access*, vol. 8, pp. 53 841–53 849, 2020.

[146] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys Tutorials*, pp. 1–1, 2020.

[147] D. Preuveneers, V. Rimmer, I. Tsingenopoulos, J. Spooren, W. Joosen, and E. Ilie-Zudor, "Chained anomaly detection models for federated learning: An intrusion detection case study," *Applied Sciences*, vol. 8, p. 2663, 2018.

[148] S. Savazzi, M. Nicoli, and V. Rampa, "Federated learning with cooperating devices: A consensus approach for massive iot networks," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4641–4654, 2020.

[149] I. Martinez, S. Francis, and A. S. Hafid, "Record and reward federated learning contributions with blockchain," in *2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, 2019, pp. 50–57.

[150] L. U. Khan, S. R. Pandey, N. H. Tran, W. Saad, Z. Han, M. N. H. Nguyen, and C. S. Hong, "Federated learning for edge networks: Resource optimization and incentive mechanism," *IEEE Communications Magazine*, vol. 58, no. 10, pp. 88–93, 2020.

[151] M. M. Moghaddam, M. H. Manshaei, W. Saad, and M. Goudarzi, "On data center demand response: A cloud federation approach," *IEEE Access*, vol. 7, pp. 101 829–101 843, 2019.

[152] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah, "Distributed federated learning for ultra-reliable low-latency vehicular communications," *IEEE Transactions on Communications*, vol. 68, no. 2, pp. 1146–1159, 2020.

[153] Y. Wang, Z. Su, N. Zhang, and A. Benslimane, "Learning in the air: Secure federated learning for uav-assisted crowdsensing," *IEEE Transactions on Network Science and Engineering*, pp. 1–1, 2020.

[154] M. Pourvahab and G. Ekbatanifard, "Digital forensics architecture for evidence collection and provenance preservation in iaas cloud environment using sdn and blockchain technology," *IEEE Access*, vol. 7, pp. 153 349–153 364, 2019.

[155] Z. A. E. Houda, A. Hafid, and L. Khoukhi, "Blockchain meets ami: Towards secure advanced metering infrastructures," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.

[156] "Pysyft." [Online]. Available: https://github.com/OpenMined/PySyft

[157] J. Zhang, M. Zulkernine, and A. Haque, "Random-forests-based network intrusion detection systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 5, pp. 649–659, 2008.

[158] H. Yang, G. Qin, and L. Ye, "Combined wireless network intrusion detection model based on deep learning," *IEEE Access*, vol. 7, pp. 82 624–82 632, 2019.

[159] P. Tao, Z. Sun, and Z. Sun, "An improved intrusion detection algorithm based on ga and svm," *IEEE Access*, vol. 6, pp. 13 624–13 631, 2018.

[160] S. U. Jan, S. Ahmed, V. Shakhov, and I. Koo, "Toward a lightweight intrusion detection system for the internet of things," *IEEE Access*, vol. 7, pp. 42 450–42 471, 2019.

[161] F. Farahnakian and J. Heikkonen, "A deep auto-encoder based approach for intrusion detection system," in *2018 20th International Conference on Advanced Communication Technology (ICACT)*, 2018, pp. 178–183.

[162] A. A. Diro and N. Chilamkurti, "Distributed attack detection scheme using deep learning approach for internet of things," *Future Generation Computer Systems*, vol. 82, pp. 761 – 768, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X17308488

[163] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21 954–21 961, 2017.

[164] S. Teng, N. Wu, H. Zhu, L. Teng, and W. Zhang, "Svm-dt-based adaptive and collaborative intrusion detection," *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 1, pp. 108–118, 2018.

[165] M. Al-Qatf, Y. Lasheng, M. Al-Habib, and K. Al-Sabahi, "Deep learning approach combining sparse autoencoder with svm for network intrusion detection," *IEEE Access*, vol. 6, pp. 52 843–52 856, 2018.

[166] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, and M. Zhu, "Hast-ids: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection," *IEEE Access*, vol. 6, pp. 1792–1806, 2018.

[167] D. Perez, M. A. Astor, D. P. Abreu, and E. Scalise, "Intrusion detection in computer networks using hybrid machine learning techniques," in *2017 XLIII Latin American Computer Conference (CLEI)*, 2017, pp. 1–10.

[168] X. Gao, C. Shan, C. Hu, Z. Niu, and Z. Liu, "An adaptive ensemble machine learning model for intrusion detection," *IEEE Access*, vol. 7, pp. 82 512–82 521, 2019.

[169] J. Zhao, S. Shetty, J. Pan, C. Kamhoua, and K. Kwiat, "Transfer learning for detecting unknown network attacks," *EURASIP Journal on Information Security*, vol. 2019, 12 2019.

[170] P. Wu, H. Guo, and R. Buckland, "A transfer learning approach for network intrusion detection," in *2019 IEEE 4th International Conference on Big Data Analytics (ICBDA)*, 2019, pp. 281–285.

[171] "Pytorch framework." [Online]. Available: https://pytorch.org/

[172] A. L. Maas, "Rectifier nonlinearities improve neural network acoustic models," 2013.

[173] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014.

[174] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2015.

[175] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," in *2016 International Conference on Wireless Networks and Mobile Communications (WINCOM)*, 2016, pp. 258–263.

[176] S. Z. Lin, Y. Shi, and Z. Xue, "Character-level intrusion detection based on convolutional neural networks," in *2018 International Joint Conference on Neural Networks (IJCNN)*, 2018, pp. 1–8.

[177] Z. Abou El Houda, A. S. Hafid, and L. Khoukhi, "Cochain-sc: An intra- and inter-domain ddos mitigation scheme based on blockchain using sdn and smart contract," *IEEE Access*, vol. 7, pp. 98 893–98 907, 2019.

[178] Z. Abou El Houda, L. Khoukhi, and A. Senhaji Hafid, "Bringing intelligence to software defined networks: Mitigating ddos attacks," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2523–2535, 2020.

[179] Z. Abou El Houda, A. S. Hafid, and L. Khoukhi, "Sdnboost: An sdn based boosting ensemble learning framework for advanced attack mitigation," *IEEE Transactions on Cognitive Communications and Networking*, 2021.

[180] ——, "Cofed: A privacy preserving collaborative ddos mitigation framework based on federated learning using sdn and blockchain," *IEEE Transactions on Network Science and Engineering*, 2021.

[181] Z. A. El Houda, A. Hafid, and L. Khoukhi, "Co-iot: A collaborative ddos mitigation scheme in iot environment based on blockchain using sdn," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6.

[182] Z. A. El Houda, L. Khoukhi, and A. Hafid, "Chainsecure - a scalable and proactive solution for protecting blockchain applications using sdn," in *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1–6.

[183] Z. A. E. Houda, A. Hafid, and L. Khoukhi, "Blockchain meets ami: Towards secure advanced metering infrastructures," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.

[184] ——, "Brainchain - a machine learning approach for protecting blockchain applications using sdn," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.

[185] Z. A. El Houda, A. S. Hafid, and L. Khoukhi, "Blockchain-based reverse auction for v2v charging in smart grid environment," in *ICC 2021 - IEEE International Conference on Communications*, 2021, pp. 1–6.

[186] Z. A. El Houda, L. Khoukhi, and A. Hafid, "A novel machine learning framework for advanced attack detection using sdnn," in *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 1–6.