Intrusion detection and prevention for IoT systems using Machine Learning

Nadia Chaabouni

THESIS PRESENTED

TO OBTAIN THE QUALIFICATION OF

DOCTOR OF

THE UNIVERSITY OF BORDEAUX

DOCTORAL SCHOOL OF MATHEMATICS AND COMPUTER SCIENCE

SPECIALIZATION COMPUTER SCIENCE

By CHAABOUNI Nadia

INTRUSION DETECTION AND PREVENTION FOR IOT SYSTEMS USING MACHINE LEARNING

Under the supervision of: MOSBAH Mohamed and ZEMMARI Akka

Defended on 13/07/2020

Members of the examination panel:

Mr. AHMED, Toufik	Professor, Bordeaux INP	President
Mr. DRIRA Khalil	Research Director at CNRS, LAAS Toulouse	Examiner
Mr. CUPPENS, Frédéric	Professor, IMT Atlantique	Recorder
Mr. GHAMRI-DOUDANE, Yacine	Professor, University of La Rochelle	Recorder
Mr. MOSBAH, Mohamed	Professor, Bordeaux INP	Supervisor
Mr. ZEMMARI, Akka	Associate Professor, University of Bordeaux	Supervisor
Mr. SAUVIGNAC, Cyrille	R&D Manager, Atos	Supervisor

Titre : Détection et prévention des intrusions pour les systèmes IoT en utilisant des techniques d'apprentissage

Résumé : Avec l'expansion de l'Internet des objets (IoT) et l'évolution des techniques d'attaque, la sécurité de l'IoT est devenue une préoccupation très importante. OneM2M est une initiative de standardisation mondiale pour l'IoT. Par conséquent, sa sécurité implique la sécurité de l'écosystème IoT. C'est pourquoi nous concentrons nos travaux sur la sécurité de ce standard. Dans cette thèse, nous proposons un système de détection et de prévention des intrusions (IDPS), basé sur les techniques d'apprentissage, pour les systèmes IoT utilisant oneM2M. Afin d'adopter les technologies émergentes et surtout avec ses résultats intéressants déjà éprouvés dans le domaine de la sécurité, les techniques d'apprentissage sont utilisées dans notre stratégie IDPS. Notre système oneM2M-IDPS détecte les menaces potentielles et y répond immédiatement. Il détecte et classifie les menaces sur trois niveaux d'apprentissage différents et réagit rapidement par des actions appropriées. OneM2M-IDPS ne traite pas seulement les menaces connues (attaques de sécurité et comportements anormaux), il est également capable de détecter les menaces inconnues (*zero-day*). De plus, l'IDPS est équipé d'un module d'apprentissage continu qui lui permet d'apprendre en permanence de nouveaux comportements afin d'être à jour.

Mots clés : Internet des objets (IoT), Techniques d'apprentissage, Détection d'intrusions, Prévention d'intrusions

---

Title: Intrusion detection and prevention for IoT systems using Machine Learning

Abstract: With the expansion of the Internet of Things (IoT) and the evolution of attack techniques, IoT security has become a more critical concern. OneM2M is a global standardization initiative for the IoT, therefore its security implies the security of the IoT ecosystem. Hence, we focus our work on the security of the oneM2M standard. In this thesis, we propose an Intrusion Detection and Prevention System (IDPS) based on Machine Learning (ML) for the oneM2M-based IoT systems. In order to adopt emerging technologies and especially with its interesting results already proven in the security domain, ML techniques are used in our IDPS strategy. Our oneM2M-IDPS detects potential threats and responds immediately. It detects and classifies threats on three different ML levels and reacts quickly with appropriate actions. OneM2M-IDPS not only handles known threats (security attacks and abnormal behaviors), it is also able to detect unknown/zero-day threats. In addition, the IDPS is equipped with a continuous learning module that allows it to continuously learn new behaviors in order to be up to date.

Keywords: Internet of Things (IoT), Machine Learning, Intrusion Detection, Intrusion Prevention

Acknowledgements

This PhD has been an enriching and tough journey, full of ups and downs, full of joy and tears and especially full of wealthy lessons. Fortunately, I was surrounded by many supportive people, who have given me strength to go through the difficult times. So, it is a great pleasure to thank them all for their positive impact during this quest.

Naturally, my first thanks come back to my supervisors Mohamed Mosbah, Akka Zemmari and Cyrille Sauvignac, for the chance they gave me to integrate their teams and work on a subject of remarkable importance with both scientific and industrial openness. I am grateful for their trust and their technical, emotional and financial support over these three years. Their advices at scientific, industrial and personal levels have resulted in the production of this work. Thanks for having shown my weaknesses and also congratulated me every time I achieve a milestone.

I would also like to thank the jury members of my defence. Many thanks to Mr. Toufik Ahmed, Professor at Bordeaux INP and Mr. Khalil Drira, Research Director at CNRS LAAS Toulouse, for agreeing to evaluate my work. I would also like to express my gratitude to Mr. Frédéric Cuppens, Professor at IMT Atlantique and Mr. Yacine Ghamri-Doudane, Professor at the University of La Rochelle, who agreed to read and review my thesis. I thank them in advance for all the attention they are willing to give to my work as well as their valuable feedback. It is really my utmost honor to have all these experts reviewing my work.

I would like to thank Atos for funding my research experience, with special gratitude to Mr. Dominique Parguel for his trust and support to make the PhD happen. In addition, I would like to express my sincere gratitude to all members of the Atos Innovation laboratory team for the friendly atmosphere in the team. I thank my colleagues Darius Matboo-Raftarhaghi, Vivien Achet, Arnaud Casteler and Nafissa Harrouz for all the debates, discussions, remarks and guidance, for their emotional support as well as the good mood that we share each day at the office. A special and deepest gratitude goes to my old colleagues Craig Josse and Charlie Huynh for their technical assistance and their wise recommendations. I would also like to thank all the people with whom I have the chance to discuss from the administration who helped with the administrative procedures.

Regarding my laboratory, the LaBRI, I am grateful for all its members that welcomed me. I would like to thank in particular Chahrazed Ksouri with whom I have had moments of complicity during this tough period. Thanks to all members of my office room as well as the members of the lab for the ups and downs that we have shared all together during this journey. I would also like to thank all the people from the administration and the system team who helped me many times and definitely contribute to the good mood at the LaBRI, thank you all.

Moreover, I would like to thank the "Association Nationale de la Recherche et de la Technologie" (ANRT) for CIFRE funding (N° 2017/0122).

Last but not least, I cannot finish without expressing all the gratitude I have for my loving family for their support and their love and comfort in the good and the bad, and for bringing joy to my soul even though we are always far away. My sincere thanks and love to my precious parents, my father Mounir, my mother Zineb and my beloved brothers Amine, Hamdi and Majdi. I am eternally indebted to them and to all my family members.

Furthermore, I would like to thank my friends. A special gratitude goes to Amel Raboudi with whom, despite the distance, I shared not only difficult and joyful moments but also reflections around the world of academic and industrial research. My sincere gratitude to all my dearest friends: (in alphabetical order) Amin, Amira, Hafsa, Houssem, Ibtihel, Ines, Salma, Yasmina and Yassine for their support and their true friendship and love, for their listening and their support in difficult times.

Finally, I am grateful to all my loved ones, all those who helped me during this PhD and whom I had the privilege of meeting during this journey.

Abstract

Pervasive growth of Internet of Things (IoT) is visible across the globe. Since the 2016 Dyn cyberattack caused by the infamous Mirai IoT botnet, IoT security has become a critical concern. The attack exposed the critical fault-lines among smart networks. The danger exposed by infested Internet-connected things not only affects the security of IoT, but also threatens the complete Internet ecosystem which can possibly exploit the vulnerable Things (smart devices) deployed as botnets. In the recent decade, security attack vectors have evolved bothways, in terms of complexity and diversity. Hence, to detect and prevent new attacks, it is important to enhance the security mechanisms with emerging technologies like Artificial Intelligence (AI) and Machine Learning (ML). Intrusion Detection Systems (IDS) are one of the strategies for using ML techniques in threat detection.

Moreover, with the increasing deployment of IoT in all areas such as healthcare, transportation, industrial automation, smart homes, etc., manufacturers tend to use proprietary solutions with customized hardware and software. Such a trend favours a fast time-to-market without sufficient embedded security and verification. OneM2M is a global standard initiative designed to satisfy the need for a common horizontal platform for the multi-industry Machine-to-Machine (M2M) / IoT applications. It aims to satisfy the need for a common M2M service layer that guarantees the communication between heterogeneous devices and applications. Various security mechanisms have been proposed in the oneM2M specifications to protect the IoT solutions. However, none of the specified techniques protect the IoT systems once the malicious user has gained access to the system and bypassed the first-line security measures. This represents a critical and unexplored topic for the oneM2M standard.

In this thesis, we decided therefore to propose an ML-based Intrusion Detection and Prevention System (IDPS) as a second line of security for IoT systems based on the oneM2M standard to detect and prevent intrusions. To the best of our knowledge, our proposal is the first IDPS for the oneM2M service layer. In order to embrace the emerging technologies and especially with its interesting results already proven in the security field, we have chosen to use ML techniques in our IDPS strategy. Our oneM2M-IDPS detects and responds immediately to potential threats as soon as they are carried out. It detects and classifies threats on three different ML levels and responds quickly with appropriate actions. OneM2M-IDPS not only deals with known threats (security attacks and abnormal behaviors), it is also able to detect zero-day/unknown threats. In addition, the IDPS is equipped with a continuous learning module that allows it to continuously learn new behaviours in order to be up to date. In order to achieve our goal, we have gone through several steps from the complete state of the art of IDPS with or without ML, to the definition of oneM2M attacks, to the creation of a dataset of these attacks, to the experimentation of different ML strategies in order to find the best ones for our proposed IDPS architecture.

Résumé

L'expansion de l'internet des objets (IoT) est visible dans le monde entier. Depuis la cyberattaque de Dyn en 2016, causée par le tristement célèbre *botnet Mirai IoT*, la sécurité de l'IoT est devenue une préoccupation plus importante. L'attaque a mis en évidence les failles critiques des réseaux intelligents. Le danger que représentent les objets contaminés connectés à Internet n'affecte pas seulement la sécurité de l'IoT, mais menace également l'ensemble de l'écosystème Internet qui peut éventuellement exploiter les objets vulnérables (dispositifs intelligents) déployés en tant que *botnets*. Au cours de la dernière décennie, les techniques d'attaques de sécurité ont évolué en termes de complexité et de diversité. Par conséquent, pour détecter et prévenir les nouvelles attaques, il est important de renforcer les mécanismes de sécurité avec des technologies émergentes comme l'intelligence artificielle et l'apprentissage automatique (*Machine Learning*). Les systèmes de détection d'intrusion (IDS) sont l'une des stratégies permettant d'utiliser les techniques d'apprentissage dans la détection des menaces.

En outre, avec le déploiement croissant de l'IoT dans tous les domaines tels que la santé, les transports, l'automatisation industrielle, les maisons intelligentes, etc., les fabricants ont tendance à utiliser des solutions propriétaires avec des matériels et des logiciels personnalisés. Cette tendance favorise une commercialisation rapide, avec moins de vérification et moins de sécurité intégrée. OneM2M est une initiative de standardisation mondiale conçue pour répondre au besoin d'une plate-forme horizontale commune pour les applications multi-secteurs Machine-to-Machine (M2M) / IoT. Elle vise à satisfaire le besoin d'une couche de service M2M commune qui garantit la communication entre des dispositifs et des applications hétérogènes. Divers mécanismes de sécurité ont été proposés dans les spécifications oneM2M pour protéger les solutions IoT. Cependant, aucune des techniques spécifiées ne protège les systèmes IoT une fois que l'utilisateur malveillant a accédé au système et contourné les mesures de sécurité de première ligne. Ceci représente un sujet critique et inexploré pour le standard oneM2M.

Dans cette thèse, nous avons donc décidé de proposer un système de détection et de prévention des intrusions (IDPS) basé sur les techniques d'apprentissage comme deuxième ligne de sécurité pour les systèmes IoT basés sur le standard oneM2M pour détecter et prévenir les intrusions. À notre connaissance, notre proposition représente le premier IDPS pour la couche de service oneM2M. Afin d'adopter les technologies émergentes et surtout avec ses résultats intéressants déjà éprouvés dans le domaine de la sécurité, les techniques d'apprentissage sont utilisées dans notre stratégie IDPS. Notre oneM2M-IDPS détecte les menaces potentielles et y répond immédiatement. Il détecte et classifie les menaces sur trois niveaux d'apprentissage différents et réagit rapidement avec des actions appropriées. OneM2M-IDPS ne traite pas seulement les menaces connues (attaques de sécurité et comportements anormaux), il est également capable de détecter les menaces inconnues/*zero-day*. En outre, l'IDPS est équipé d'un module d'apprentissage

continu qui lui permet d'apprendre en permanence de nouveaux comportements afin d'être à jour. Afin de réaliser notre objectif, nous sommes passés par plusieurs étapes allant de l'état de l'art des IDPS munis ou non des techniques d'apprentissage, à la définition des attaques oneM2M, à la création d'une base de données de ces attaques, jusqu'à l'expérimentation de différentes stratégies d'apprentissage afin de trouver les meilleurs pour notre architecture IDPS.

Contributions

La principale contribution de cette thèse est la conception et la mise en œuvre d'un IDPS basé sur les techniques d'apprentissage pour protéger les objets qui utilisent le standard oneM2M. Pour atteindre cet objectif, un examen complet de l'écosystème de la sécurité de l'IoT est réalisé dans un article publié dans la revue *IEEE Communications Surveys & Tutorials* intitulé "**Network intrusion detection for IoT security based on learning techniques**" [CMZ+19]. L'article porte sur trois domaines importants : i) l'IoT, ii) les mécanismes de sécurité et iii) les techniques d'apprentissage automatique. Il guide le lecteur dans la découverte de l'intersection de ces trois domaines en fournissant tous les détails nécessaires pour comprendre la sécurité des systèmes IoT. Le document commence par présenter et catégoriser les menaces de sécurité contre l'IoT ainsi que les techniques de défense traditionnelles en mettant l'accent sur les types d'IDS. Ensuite, il énumère et examine les outils disponibles qui peuvent être utilisés pour développer et/ou évaluer les IDS déployés niveau réseau (NIDS): i) les corpus de données gratuits, ii) les renifleurs de réseau gratuits et open source et iii) les NIDS open source. En outre, le document traite les NIDS conçus pour l'IoT, leurs architectures, leurs déploiements et leurs applications dans les systèmes hétérogènes. Les techniques d'apprentissage sont présentées. Ensuite, les NIDS pour les systèmes IoT déployés via des techniques d'apprentissage sont détaillés, comparés et évalués. Nous avons examiné en détail l'état de l'art actuel, comparé et évalué les performances des systèmes NIDS basés sur les techniques d'apprentissage et déployés pour sécuriser les réseaux IoT. Enfin, le document s'est conclu par un résumé et une liste des orientations futures possibles.

Compte tenu de la nécessité d'un écosystème IoT standardisé, en particulier avec la croissance significative du nombre de dispositifs connectés, nos travaux se concentrent sur le standard oneM2M. Nous avons donc défini les attaques de sécurité qui menacent la couche de service oneM2M et proposé une abstraction pour les flux oneM2M appelée *GFlow*. Afin de mettre en œuvre un mécanisme de sécurité basé sur l'apprentissage, l'abstraction *GFlow* est utilisée pour la création d'un corpus de données de sécurité oneM2M. Ce corpus a été créé avec une plate-forme réelle et contient les attaques oneM2M ainsi que les *GFlows* légitimes. Grâce aux données générées, nous avons proposé le premier IDS générique pour la couche de service oneM2M basé sur l'apprentissage "à la frontière" ou ce qu'on appelle *edge machine learning*. Afin de choisir le meilleur algorithme d'apprentissage pour nos besoins, divers algorithmes sont

expérimentés pour des classifications binaires et multiples. Toutes ces études ont fait l'objet de la publication "**An Intrusion Detection System for the OneM2M Service Layer Based on Edge Machine Learning**" [CMZS19] dans la conférence internationale *Ad-Hoc Networks and Wireless (Adhoc-Now)*.

Un système de détection et de prévention des intrusions oneM2M (oneM2M-IDPS) basé sur l'apprentissage automatique était au centre de notre article "**A OneM2M Intrusion Detection and Prevention System based on Edge Machine Learning**" [CMZS20] publié dans *IEEE/IFIP Network Operations and Management Symposium (NOMS)*. Dans le cadre de ce travail, une architecture pour le oneM2M-IDPS est proposée. Chaque module de notre stratégie IDPS est détaillée en commençant par le module d'acquisition de données et d'extraction de caractéristiques. Ensuite, un nouvel IDS basé sur trois niveaux de détection est proposé. De plus, cet IDS est enrichi de deux nouveaux modules : i) un module de prévention pour agir contre les menaces détectées et ii) un module d'apprentissage continu pour fournir un IDPS à jour qui évolue avec l'évolution et l'émergence de nouvelles menaces.

Dans les travaux précédents, le module IDS utilisait des algorithmes d'apprentissage pour les trois différents niveaux de détection en mode classification binaire et multiple. Dans notre article de revue "**Anomaly and Novelty Detection and Prevention for IoT Security: A Continuous Machine Learning Approach**", soumis à la fin de la thèse, nous détectons les anomalies dans le standard oneM2M d'une part, et les nouveautés en termes de menaces d'autre part, dans une plateforme de détection et de prévention des intrusions. En ce qui concerne la détection des anomalies, les algorithmes de classification à une classe (OCC) sont examinés pour détecter les comportements qui diffèrent de la norme. De plus, une nouvelle méthode de détermination de seuil basée sur l'algorithme de l'arbre de décision est proposée. Toutefois, pour la détection des nouveautés, l'approche OCC est étendue pour i) détecter les nouvelles menaces non connues et ii) classer celles qui sont déjà connues auparavant. En outre, le flux de travail du module de prévention est spécifié et les actions de prévention sont précisées. De plus, les différents scénarios qui activent le module d'apprentissage continu, de manière autonome ou à la demande de l'Homme, sont décrits en détails. Ils permettent la mise à jour permanente du système IDPS.

Organisation de la thèse

La thèse est organisée en six chapitres qui sont eux-mêmes divisés en plusieurs sections. Le chapitre 1 présente l'écosystème de la sécurité des objets connectés. La section 1.1 commence par la présentation et la classification des catégories de menaces de sécurité de l'IoT ainsi que des techniques de défense traditionnelles afin de fournir au lecteur le contexte de sécurité nécessaire pour une meilleure compréhension de ce manuscrit. Dans la section 1.2, notre motivation pour ce travail ainsi que les problé-

matiques auxquelles nous essayons de répondre sont détaillées. Dans la section 1.3, les différentes contributions apportées aux problématiques étudiées durant la thèse sont présentées.

Dans le chapitre 2, la littérature traitant les IDPS, déployés au niveau réseau dans les systèmes IoT, est présentée et discutée. La section 2.1, commence par l'état de l'art des IDPS pour l'IoT qui ne sont pas basés sur des techniques d'apprentissage. Ces travaux se concentrent davantage sur l'architecture de déploiement et la stratégie de détection. Après avoir détaillé chaque travail séparément, nous les discutons et les comparons sur la base de leurs architectures (si elle est distribuée, centralisée ou hybride), de leurs méthodologies de détection (basées sur les signatures, les anomalies ou hybrides), de leurs stratégies de validation (simulation ou émulation) ainsi que des menaces traitées. Une importance a également été accordée aux forces et faiblesses de chacune des propositions. La section 2.2 concerne les IDPS basés sur les techniques d'apprentissage. Comme dans la première partie, nous fournissons une brève description de chaque travail, puis nous comparons et discutons leurs déploiements, leurs méthodologies, les corpus de données qu'ils ont utilisés, les menaces qu'ils ont traitées et les algorithmes d'apprentissage qu'ils ont employés.

Dans le chapitre 3, un aperçu du standard oneM2M (section 3.1) est fourni: son architecture et ses mécanismes de sécurité définis dans ses spécifications. Ensuite, dans la section 3.2, nous nous concentrons sur les menaces oneM2M liées à la disponibilité des services puisque oneM2M concerne par définition les services pour les systèmes M2M et IoT. Une taxonomie et une mise en œuvre des menaces oneM2M sont proposées. Ce chapitre se termine par la section 3.3 qui détaille la création d'un corpus de données de sécurité oneM2M. Elle commence par l'état de l'art des corpus de données libres utilisés pour la création des IDS IoT déployés au niveau réseau. Comme il n'existe pas de corpus de données pour le standard oneM2M, nous créons nos propres données en respectant la taxonomie des menaces oneM2M présentée dans la section précédente. Ce jeu de données est basé sur une nouvelle proposition d'abstraction pour les flux oneM2M. Cette abstraction est le fondement de notre IDPS.

Dans le chapitre 4, nous présentons les différents défis et objectifs respectés et garantis avec notre proposition oneM2M-IDPS : l'interopérabilité, l'autonomie, l'extensibilité et le respect des contraintes de ressources, la modularité dans la conception, l'adaptabilité et l'extensibilité et enfin la réaction active et en temps réel (section 4.1). De plus, la section 4.2 détaille la stratégie ainsi que l'architecture de l'IDPS et la conception de chacun de ses quatre modules : i) le module d'acquisition de données et d'extraction de caractéristiques, ii) le module IDS, iii) le module IPS ainsi que iv) le module d'apprentissage continu et d'annotation humaine.

Le chapitre 5 se focalise sur l'aspect d'apprentissage automatique de notre IDPS. Notre module de détection est mis en œuvre avec ses trois niveaux d'apprentissage en utilisant le corpus de données oneM2M décrit dans le chapitre 3. Différents algorithmes

d'apprentissage et d'apprentissage profond *Deep Learning* sont expérimentés pour chaque niveau de détection afin de choisir les plus appropriés et les plus efficaces. Ce chapitre commence par la section 5.1 qui explique notre choix d'adoption des techniques d'apprentissage pour les niveaux de détection. Ensuite, nous présentons les métriques sur lesquelles nous nous sommes appuyés pour évaluer l'efficacité de chaque algorithme dans notre contexte de détection d'intrusion oneM2M, ainsi que l'environnement expérimental. De plus, deux approches principales sont étudiées: i) la détection basée sur des algorithmes d'apprentissage supervisés (section 5.2) et ii) la détection basée sur une approche de classification à une classe (section 5.3). Pour chacun des deux, les algorithmes expérimentés (définitions et outils) sont présentés, les différentes étapes de la mise en place des expériences sont expliquées, puis les résultats sont exposés, comparés et discutés. En ce qui concerne la première approche, l'apprentissage supervisé est choisi vu que les expériences menées dans la section correspondante s'inscrivent dans le contexte de la classification où nous disposons de données annotées et labélisées. Dans cette section, les trois niveaux de détection de notre IDS sont expérimentés, puis l'effet de la taille et de l'équilibre du jeu des données d'entraînement sur le processus de détection est étudié. Dans la dernière section de ce chapitre, nous examinons l'approche de classification à classe unique pour le premier et le deuxième niveau de détection. Dans le premier niveau, nous mettons en évidence la détection de tout comportement différent de la normale (connu pour la détection d'anomalies) ainsi que les techniques permettant de déterminer le seuil de normalité. Dans le deuxième niveau, l'approche de classification à une classe est adaptée pour permettre la détection des menaces inconnues ainsi que la multi-classification des menaces déjà connues (détection de nouveauté et multi-classification).

Le chapitre 6 conclut la thèse avec un résumé des principales contributions et une proposition détaillée de quelques pistes de recherche qui peuvent être suivies pour traiter davantage les problématiques exposées.

Acronyms and Abbreviations

6LoWPAN IPv6 over Low-power Wireless Personal Area Network

ACP Access Control Policy

ADA Amplify Discovery Application entity

ADN Application Dedicated Node

AEnc Auto-Encoder

AE Application Entity

AF Application entity Flooding

AIS Artificial Immune System

AI Artificial Intelligence

AOAMC Amplify One Application entity Multiple Containers

AOAOC Amplify One Application entity One Container

API Application Programming Interface

ASN Application Service Node

CF Container Flooding

CIF ContentInstance Flooding

CSE Common Services Entity

CsF Containers Flooding

DDoS Distributed Denial of Service

DL Deep Learning

DoS Denial of Service

DR Detection rate

DT Decision Tree

ELM Extreme Learning Machine

ESFCM Semi-supervised Fuzzy C-Means

FPR False Positive Rate

GUI Graphical User Interface

IDPS Intrusion Detection and Prevention System

IDS Intrusion Detection System

IN Infrastructure Node

IoT Internet of Things

IPS Intrusion Prevention System

KNN K Nearest Neighbors

LDA Linear Discriminant Analysis

LR Logistic Regression

M2M Machine-To-Machine

MITM Man-In-The-Middle

MLP Multi Layer Perceptron

ML Machine Learning

MN Middle Node

N_OP Number of OPerations

N_TH Number of THreads

NB Naive Bayes

NIDS Network Intrusion Detection System

NSE Network Services Entity

OC-SVM One-Class SVM

OCC One-Class Classification

OPF Optimum Path Forest

Op Operation

OS-ELM Online Sequential Extreme Learning Machine

O Originator

PCA Principal Component Analysis

R2L Remote-to-Local

RBF Radial Basis Function

RF Random Forest

RPL Routing Protocol for Low-Power

R Resource

SAE Sparse Auto-Encoder

SF Subscription Flooding

SMO Sequential Minimal Optimization

SVM Support Vector Machines

TPR True Positive Rate

U2R User-To-Root

VAE Variational Auto-Encoder

VF Various Flooding

List of Figures

List of Tables

Contents

Chapter 1

Introduction and IoT Security Overview

Contents

Internet of Things (IoT) is considered as the third industrial revolution [Rif14]. It is defined as "the interconnection, via the Internet, of computing devices embedded in everyday objects, enabling them to send and receive data" [Gra14]. IoT market is growing at a breathtaking pace, starting with 2 billion objects in the year 2006 to projected 200 billion by 2020 [IDC15]. IoT sensors/devices often collect and process spatial and temporal information for specific events and environment tackling various challenges [GBMP13], [STJ14]. The IoT objects or Things have become smarter, treatment is more intelligent and communications have turned instructive. Therefore, IoT is used in almost all fields: domestic, education, entertainment, energy distribution, finances, healthcare, smart-cities, tourism and even transportation [VF14]. Consequently, indus-

try, academia and individuals are trying to integrate the flow of fast commercialization with seldom attention to the safety and the security of IoT devices and networks. Such a neglect can possibly endanger the IoT users and in turn disrupt the vibrant ecosystem. For example, Smart-homes can be remotely controlled by cyber-criminals and Smart vehicles can be hijacked and remotely controlled to create panic among citizens.

The danger exposed by these Internet-connected Things does not only affect the security of IoT systems, but also the complete eco-system including web-sites, applications, social networks and servers, via controlled smart device as robot networks (botnet). In other words, compromising a single component and/or communication channel in IoT-based systems can paralyze the part or complete Internet network. In 2016, the Dyn cyberattack [Kep16] harvested connected devices installed within smart-homes and conscripted them into "botnets" (also referred to as a "zombie army") via a malware called Mirai. In addition to IoT systems vulnerabilities, attack vectors are evolving in terms of complexity and diversity. Consequently, more attention should be paid to the analysis of these attacks, their detection as well as the infection prevention and recovery of systems after the attacks.

1.1 Overview of Threats and Security Mechanisms in IoT

Since security of pervasive IoT systems is critical, it is important to identify IoT threats and specify existing defense strategies. In this section, we introduce and categorize the IoT security threat categories as well as traditional defense techniques in order to provide the reader with the security necessary background for a better understanding of the dissertation.

IoT security threatens the security of traditional systems; However, IoT security mechanisms are different from those used in traditional systems, mainly for the following reasons:

- IoT systems are constrained in terms of computational capability, memory capacity, battery life and network bandwidth. Hence, it is not possible to deploy existing traditional security solutions which are often resource intensive.

- IoT systems are heavily distributed and heterogeneous systems. Thus, centralized traditional solutions may not be suitable. Moreover, the distributed aspect of IoT adds more difficulties and constraints in their protection.

- IoT systems are deployed in a physical environment which is unpredictable. Thus, physical attacks have joined the list of traditional security threats.

- IoT systems are connected to the Internet since each device can be accessed with its IP address. There is therefore an additional range of Internet-related threats.

- IoT systems are composed of a large number of constrained objects that generate huge amount of data. So it is easy to flood and attack these small devices on the one side, and the limited bandwidth of the networks on the other side.

- IoT systems cover a large number of heterogeneous protocols and technologies in the same system. Hence, the proposed IoT security solutions must take into consideration the large panel of these protocols and technologies in the same proposal

Two main topics are covered in the rest of this section: the categorization of IoT threats and traditional defence mechanisms.

1.1.1 Categorization of IoT threats

Since the IoT systems are varied and are facing multiple challenges, IoT threats can be categorized into two types. The first type is about categorization depending on the layers of the IoT systems' architecture, while the second one deals with IoT threats categorization based on their design challenges.

1.1.1.1 IoT threats categorization by layers

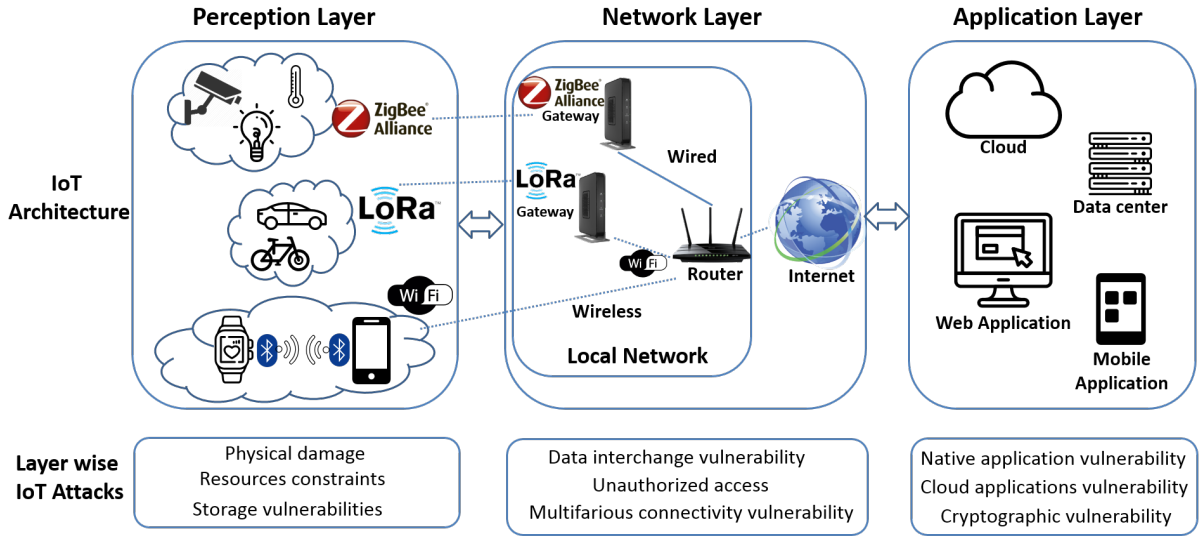

Figure 1.1: IoT architecture & layer wise attacks

IoT systems relate physical environment to the virtual one. A standard representation of IoT architecture is shown in Figure 1.1. IoT consists of three main layers [KU14] which are perception (physical) layer, network (transport) layer and application layer.

First, the perception layer is the hardware layer. It is composed of the different sensors and actuators that send and receive data using different communication standards such as Bluetooth, RFID and 6LowPAN. Second, the network layer is the one which ensures the effective routing/transmission of data/information. It uses communication protocols like WiFi, 3G, 4G, 5G, GSM, IPv6, etc. Third, the application layer, also called the software layer, is the top layer that provides systems with the business logic and offers the user interfaces to the end users (traffic monitoring, smart classroom, etc.).

Each layer may present multiple vulnerabilities [BSP+11] as illustrated in Figure 1.1. Since devices are placed at different physical locations, they may be exposed to environmental hazards [IJL16] like rain/snow/wind or malicious attacks or unintentional damage. Also, stored data may be stolen via physical access. Sensors are tiny Things so, they suffer from resource constraint issues [IJL16] (computational resources, memory or energy, etc.). While data/commands are exchanged (network layer), they can

face different network vulnerabilities such as data interchange vulnerabilities [IJL16] (data transfer can be shut down because of network floods or malicious gateway access), unauthorized access [IJL16] (impersonation attack, communication interception, password guessing attacks, etc.) and multifarious connectivity vulnerabilities [IJL16] (data integrity violation, bad Quality-Of-Service (QoS), etc.). Moreover, the application layer is mainly exposed to software problems [IJL16] such as account enumeration, insecure account credentials and lack of account suspension after a limited number of password guessing. Cloud applications [MPB+13, MPVT17] can be attacked by viruses, trojan horses, worms, etc.. Since IoT is based on low computational capability devices, transport encryption is sometimes neglected or used in a weak version. Therefore, communications are easily traceable and easily discovered (Cipher text-only attack, Man-In-The-Middle).

1.1.1.2 IoT threats categorization by challenges

To understand IoT security attacks first, we introduce some IoT attack technical terms, then we present IoT challenges-based categorization.

1.1.1.2.1 Technical terms of the attacks

- **Spoofing** [AM14, CDL16] or impersonation attack sneaks authentication credentials to gain unauthorized service access. Credentials can be stolen directly from a device, via eavesdropping the communication channel or by phishing. Spoofing can be categorized into: i) IP address spoofing; ii) ARP spoofing; and iii) DNS server spoofing. IP address spoofing refers to the falsification of content in the source IP header to mask sender's identity or to launch a reflected distributed denial of service (DDoS) attack. ARP spoofing attacks typically address resolution protocol (ARP). The spoofing attack resolves IP addresses to Media Access Control (MAC) addresses. When an attacker sends spoofed ARP messages across the Local

Area Network (LAN), attacker's MAC address will be linked with the IP address of a legitimate member of the network. Consequently, malicious parties can steal data, modify data in-transit or even stop traffic on a LAN. DNS server spoofing modifies a DNS server (a system that associates to each domain name an IP address) reroutes a specific domain name to unauthorized IP address of the infected server.

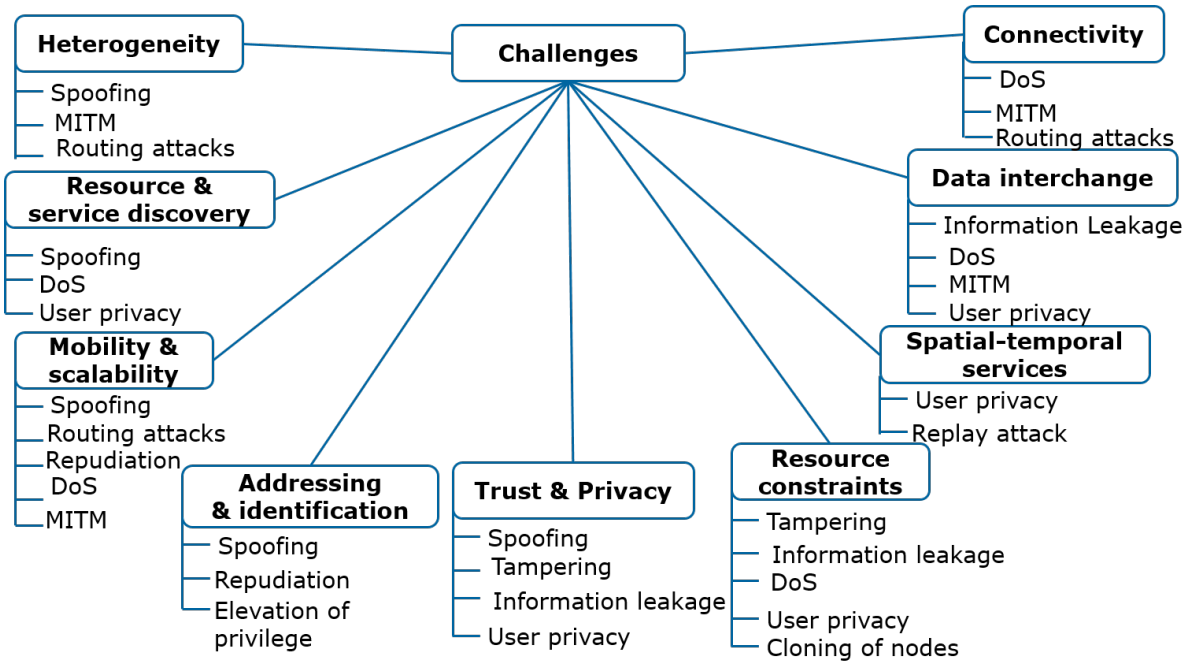

Figure 1.2: IoT threats categorization by design challenges

- **Routing attacks** [KW03] target routing protocols where the exchanged routing information are spoofed, altered or replayed to generate fictitious routing behaviors (i.e., false network traffic attraction). Sinkhole attack [BS17] concerns a malicious node attracting huge traffic by presenting imaginary path as an optimal routing path. Regarding selective forwarding attack [WRV13], it is a data forwarding misbehavior where an attacker selectively forwards malicious packets while discarding genuine and important packets. Furthermore, blackhole attack [BS17] aims to disrupt normal data flow within a network. Initially, the attack fudges one or more faulty nodes as the best route(s); then, starts to drop data pack-

ets routed through the faulty path. On the other hand, wormhole attack [BS17] needs at least two faulty nodes strategically located in the network and connected via wired or wireless link. These malicious nodes tunnel the packets faster than normal track. They claim their route, to the rest of the nodes in the network, as the shortest way to transfer their information. Moreover, replay attack [AM14] considers the re-transmission or the delaying of valid data to gain unauthorized access within already established session.

- **Tampering** attack [AM14] is categorized into: i) Device tampering; and ii) Data tampering. The device tampering can be easily performed especially when an IoT device spends most of the time unattended. It can be easily stolen without being noticed and so used maliciously. The device can be stolen as hardware or just as software. The data tampering involves malicious modification of data. This can be data stored in databases or data transiting between two devices.

- **Repudiation** [Mic05] is about devices doing a malicious action and then denying performing it. It is the case when a device sends a virus on the network without leaving any trace to identify it. This attack particularly threatens systems that do not have the ability to track and record prohibited transactions.

- **Information disclosure** [AM14] deals with unauthorized information access. An attacker achieves the same by attaching snooping devices, by snooping the network channel or by getting physical access to a device; e.g Probe [AOF+10, BHDA14] is when attackers try to gather information about a target node and its vulnerabilities by scanning connections (Port scanning, etc.). With information disclosure comes sensitive information leakage such as side channel attack [AIZ+17].

- **DDoS** [KU14], the Distributed Denial of Service attack, is performed by multiple compromised nodes together from different geographic locations. Besides, **DoS** attack implicates a malicious attacker that attempts to consume network resources,

target CPU time and/or bandwidth of legitimate users by flooding the system with rogue and amplified traffic. To conduct an efficient DDoS attack, botnets are used. They are networks of infected/controlled internet-connected devices. As mentioned in [KKSG16], DDoS attacks are the most frequent attacks especially in IoT / Fog networks related to social IoT such as smart cities, etc. DDoS attacks can be categorized into the following types [ZFS17, PJ14]:

1. Flooding attacks are based on bombarding a victim's system with a large number of packets, mainly UDP or ICMP packets [Cis12], which causes compromise of network bandwidth. Flooding attack can be easily launched using botnets.
2. Amplification attacks can be established by exploiting reflection mechanism and spoofing the IP sources. Attackers send packets to reflector servers with a source IP address set to their victim's IP therefore indirectly overwhelming the victim with the response packets. To resume, hackers exploit vulnerabilities in different protocols to turn small queries into huge number of requests to slowdown and/or crash the victim's server(s). For examples, there are smurf, fraggle attacks [PJ14] and DNS, SSDP amplification [Hen16, Ros14] distributed attacks.
3. Protocol exploit attacks are built on malicious exploit of different protocols. As examples, there are SYN flood [Hen16], TCP reset [Hen16] and water torture attack [Lir16, Hol16].
4. Malformed attacks are based on malformed network packets such as using the same IP address for source and destination addresses [PJ14, Hen16].
5. Logical/software attacks are related to application protocols. An example is Ping of Death [KU14] where an attacker sends simple fragmented ICMP ECHO request packet, larger than maximum IP packet size so that the victim fails to reassemble it. In Teardrop [Cis12] attack, the adversary sends two fragments that do not reassemble by the offset value of the packet.

- **Elevation of privilege** [Mic05, AM14] is about obtaining or elevating privileges to access a device/service while not having a legal right. Such an attack can lead to dangerous situation especially when the attacker becomes a trusted part system. User-To-Root (U2R) and Remote-to-Local (R2L) [HB99, AOF+10] are two examples of elevation of privilege attack. U2R is about gaining root privileges (superuser) on a node when the attacker initially has only a normal user account. R2L occurs when an attacker does not have an account on the victim node, hence exploits vulnerabilities to gain local access as a user via password guessing or breaking.

- **MITM** [AM14, CDL16], Man-In-The-Middle attack which represents interception of communication between two systems by an adversary to eavesdrop a conversation. MITM attacks are classified as ARP Cache poisoning, DNS spoofing, session hijacking, ICMP redirect, port stealing, etc.

- **User privacy** [AM14, ADMC17] is like information disclosure. Besides, a hacker does not necessarily need to have access to unauthorized information to learn about a user. This can be done by analyzing metadata and traffic.

- **Cloning Nodes** [GR14, SJS15, BJCF+18] concerns reintroducing a clone of a node in the network or a component in a system after capturing the credentials and the characteristics of the original one. Such an attack enables the malicious user to control the system, insert false information, disable functions, etc. Once an object is under the control of the attacker without the knowledge of its owner (botnet), the entire network can be infected.

1.1.1.2.2 IoT threats categorization by design challenges Because of different challenges related to IoT systems' design, developers as well as industries should pay attention to many potential threats. Many research papers surveyed IoT security challenges and research opportunities such as Zhang et al. [ZCW+14], Mahmoud et al. [MYAZ15] and Hossain et al. [HFH15]; they detail IoT security challenges such as

Object identification, Authentication and IoT privacy, etc. in IoT networks. A categorization based on IoT systems' design challenges is presented in Figure 1.2 and detailed below.

- **Heterogeneity and Interoperability**: the backend IoT solutions consider the use of sensors, actuators and gateways provided by different vendors and may have different versions. To do so, the use of a dispositive managing interoperability between heterogeneous devices is needed. Such a component can be bombarded with fake requests that can lead to DoS attacks. In such a heterogeneous environment, spoofing, routing attacks as well as MITM are more likely to occur compared to the homogeneous systems. It is easier for a malicious node to impersonate a genuine Thing, gain unauthorized access to data and/or relay communication between two nodes message injection. As we can see from Figure 1.1, IoT might be considered as the term which references a world of large variety of heterogeneous protocols and standards [GMS15]. Their consideration makes IoT security solutions more and more complex. Al-Fuqaha et al. provide a good survey about these technologies in [AFGM+15].

- **Connectivity**: in IoT, connectivity between different components of the system is required whether physical or in terms of availability of services. For the first case, data from peripheral devices (sensors for example) have to be connected to an IP network with bridging devices which may be the cause for routing attacks as well as MITM attacks. For the connectivity in terms of services, changes in the availability of services should be notified to the respective devices so that the latter do not flood the system unknowingly with repetitive and non-available requests. Such a flood can lead to a DoS attack. Moreover, the QoS in IoT networks can be crucial specially in emergency situations. Thus, robust packet routing and a good QoS in data delivery should be ensured even in highly dynamic topologies [AT18].

- **Mobility and Scalability**: devices of IoT systems can be in continuous mobility in the field area; hence, they can change bridges they are connected to. This

often causes disruption of discontinuity and/or connections to unauthorized services. Attacks like repudiation, MITM, DoS, sinkhole and wormhole become potentially possible. To mitigate such risks, security solutions not only consider mobile devices, but also network components such as the switches and the routers [AATD18].

- **Addressing and Identification**: field devices in IoT applications use usually low power radios for short distance connection (less than 1 kilometer). For that, coordinator nodes allocate local addresses that do not follow a common standard, to peer devices [BTC15]. Consequently, these addresses remain hidden behind the FAN gateway/bridge; hence, malicious behaviors become untraceable [BTC15]. As a result, isolation of malicious node(s) and detection of spoofing and repudiation attacks are difficult. Further, the node(s) can attempt to access unauthorized privileges without being screened from the outside network (elevation of privilege).

- **Spatio-temporal services**: events in IoT can be characterized by the amplitude of spatio-temporal impulse [BTC15]. As a result, data from IoT devices of same systems should have reasonable temporal behavior and spatial geolocation. However, these spatio-temporal tags must be protected from malicious users to avoid replay attacks [BTC15]. Also, the user's location data must not be revealed to unauthorized users.

- **Resource constraints**: most peripherals IoT devices are tiny, which means that they are resource constrained in terms of computing power, onboard memory, network bandwidth and energy availability. Tampering, information leakage and node cloning are possible attacks since the smart devices and sensors are resource constrained. The constrained resources limit the deployment of cryptographic solutions hence, lightweight solutions are foremost concern. For example, in [AAT18], authors overcome such limitations and propose a novel error correc-

tion and detection technique entitled "Low Complexity Parity Check (LCPC)", to improve the quality of futuristic IoT networks.

- **Data Interchange**: before data interchange begins, it must be encrypted at the source IoT nodes. The encryption mechanisms, depend on the type of hardware, their computational capability and storage capacity. Inappropriate selection leads to security vulnerabilities such as information leakage (i.e., keys are being shared between multiple devices when encrypted packets are decrypted and repacked at multiple points in the communication chain). Additionally, nodes that encrypt data can be attacked via denial of service or resource exhaustion attacks. For this reason, end to end encryption is desirable.

- **Resource and service discovery**: in IoT systems, mechanisms of resource and service discovery should be deployed to enable autonomy and self-discovery of the devices. These mechanisms should be protected with two way authentication to avoid spoofing and restrict the malware component from flooding the system with feigned requests to thwart DoS attacks.

- **Trust and privacy**: IoT smart sensor devices manage private / sensitive user informations (e.g. user habits, patients data, civil protection data etc.); hence, confidentiality and data protection are extremely important. In fact, trust and privacy [SRGCP15, YWY+17] are fundamental issues for IoT based networks. Users, Things and devices are required to authenticate via reliable services for mitigating spoofing, tampering and information leakage attacks. Trust and privacy are getting more attention with smartphones equipped for example with Android OS [SAU17, FBL+15, GLT+18, SLSGZ19, BBJJ+17, GAL+20, BHL+17].

1.1.2 Traditional defense mechanisms

After detailing and categorizing IoT threats, in the following we discuss attack mitigation techniques which protects existing IoT systems and networks. Over time,

conventional IT security solutions have covered servers, networks and cloud storage. Most of these solutions can be deployed for security of IoT systems. Defense mechanisms can be separate, or combined depending on the treated threats [CBO17]. In this section, traditional mechanisms that can be used to protect IoT systems are described.

First, **filter packets** [Tan03], with firewalls and proxies for example, represent an important defense against IP spoofing attacks (and consequently DDoS attacks). Two types of filtering are possible: i) ingress filtering; and ii) egress filtering. Ingress filtering on incoming packets is about blocking packets from outside the network with a source address inside the network to protect against outside spoofing attacks. However, egress filtering on outgoing packets is about blocking packets within the network with a source address that is not inside to prevent an internal hacker from attacking external machines.

Second, **adopt encryption** with cryptographic protocols, data storage encryption or virtual private networks (VPNs). Using cryptographic network protocols (i.e., Transport Layer Security (TLS), Secure Shell (SSH), HTTP Secure (HTTPS), etc.) leads to the encryption of data/code/updates before sending and authenticating them. The defense is based on digital signatures/certificates (pair of public and private keys) to ensure, in one hand, that data/code/update was sent by the legitimate device/service and was never modified. On the other hand, it guarantees that data/code/updates are encrypted and cannot be read or used by unauthorized individual. Cryptographic network protocols [FMA+18] can be used to protect things against IP spoofing, tampering, repudiation, MITM, user privacy compromising attacks and node cloning. Moreover, encrypting data storage helps prevent information disclosure and maintains user privacy. Regarding VPN, it is a secure communication tunnel between two or more devices. It encrypts the communication by creating a virtual private link over the existing unsecure network. Encryption is a good solution to preserve confidentiality and privacy. However, IoT networks are vulnerable since

the resource limits the devices. Therefore, the use of light cryptographic solutions such as proposal from Al-Turjman et al. [ATA18a] is an interesting approach. They propose a platform based on confidential cloud-assisted wireless sensor networks (WSN)[1]. It preserves confidentiality, integrity and access privileges (CIA). The proposed agile framework ensures integrity of collected sensor data with elliptic curve cryptography.

Third, **employ robust password authentication schemes**. This involves limiting access to data by assigning appropriate privileges to resources. The use of One-Time Password (OTP) can be an interesting solution. Spoofing, tampering, information disclosure, elevation of privileges and MITM can be avoided by the above mechanisms. For IoT networks, authentication strategies need to be lightweight such as in [BJMYZ15, BJMYZ13, ATEE+17, ATA18b].

Fourth, **audit and log activities** on web servers, database servers and application servers. Due to these traces, anomalies can be detected. More specifically, log key events such as transaction, login/logout, access to file system or failed resource access attempt(s) can detect anomalous behavior. A good practice to protect these files is to back up them, regularly analyze them for detection of suspicious activity and relocate system log files from their default locations. Further, secure the log files by using restricted ACLs (Access Control List: a list of permissions attached to an object) and ecrypt the transaction log. These techniques prevent IoT systems from repudiation and privilege elevation attacks.

Fifth, **detect intrusions using IDS (Intrusion Detection System)**. An **IDS** [PKRM12] is a combination of software and hardware which monitors network or systems to identify malicious activities and provide immediate alerts. IDSs have been adopted [ANMH16] since 1970 [HHS+17]. They are generally categorized according

[1] A WSN is wireless geographically distributed network of sensors to monitor physical or environmental conditions.

to i) deployment; and ii) detection methodology.
IDS deployment is categorized as i) HIDSs; and ii) NIDSs. Host-based Intrusion Detection Systems (HIDSs) are installed on a host machine (i.e., a device or a Thing). They monitor and analyze activities related to system application files and operation system. HIDSs are preferred against insider intrusion deterrence and prevention. Network-based Intrusion Detection Systems (NIDSs) capture and analyze packet flow in the network. In other words, they are scanning sniffed packets. NIDSs are strong against external intrusion attacks. Since our interest is towards security of resource constrained IoT systems, the rest of the chapter will focus on NIDSs solutions.

In the following, we discuss scenario after the intrusion has happened. A good detection system is the one which identifies the compromised situation and minimizes the loss by quickly identifying the attack(s).

There is a variety of IDSs. In [KPSV13], detection methodologies are categorized into i) misuse detection; ii) anomaly detection; iii) specification detection; and iv) hybrid detection.

- Misuse detection or signature detection (knowledge based) is a set of predefined rules (such as bytes sequence in network traffic or known malicious instructions sequence used by a malware) that are loaded and matched with events. When a suspicious event is detected, an alert is triggered. This type of IDS is efficient for known attacks; unfortunately it cannot detect zero-day [BHDA14] / unknown / unseen attacks [CMO12] due to lack of signatures. Cyber security solutions prefer signature based detection as it is simple to implement and effective for identifying known attacks (high detection rate with low false alarm rate).

- Anomaly detection (behavior based) [PP07] compares a normal recorded behavior with current input. Initially, normal network and system behavior are modeled. In case of deviation from normal behavior, the detector considers it as an attack.

Anomaly is identified with statistical data analysis, mining and algorithmic learning approaches. Anomaly detectors are successful in preventing unknown attacks. However, they tend to generate high false positive rate since previously unseen (yet legitimate) behaviors may be classified as anomalous. Anomaly IDSs make attacks more difficult to succeed since normal profile activities are customized for each system, each application and each network. It is difficult to know exactly which activities can be undetected.

- Specification detection has the same logic as anomaly detection. It defines anomaly as deviation from normal behavior. This approach is based on manually developed input specifications to capture legitimate (rather than those previously seen) behavior and its deviations. However, specifications require the user to give input. This method reduces high false alarm rate as compared to anomaly detectors.

- Hybrid detection is a combination of previous methods, especially signature and anomaly based detection. Hybrid detector improves accuracy by reducing false positive events. Most of the existing anomaly detection systems are in reality hybrid ones. They start with an anomaly detection, then try to relate it with the correspond signature.

Sixth, **prevent intrusions with IPS (Intrusion Prevention System)**. An **IPS** is an IDS which responds to a potential threat by attempting to prevent it from succeeding. An IPS responds immediately and stops malicious traffic to pass before it responds by either dropping sessions, resetting sessions, blocking packets, or proxying traffic. However, an IDS responds after detecting passed attacks. There are many types of IPS [PKRM12] mainly in-line detection, layer seven switches, deceptive systems, application firewalls and hybrid switches. To get more details about IPS types, please refer to Patil et al. paper [PKRM12].

1.2 Motivation and Problem Statement

With the increasing deployment of IoT systems and the fact that their security affects IoT systems themselves as well as other systems connected to the Internet, we focus our research on IoT security.

The traditional mechanisms presented in Section 1.1.2 can be used to protect IoT systems. However, some of them like encryption and authentication are insufficient and inadequate [RWV13] to protect IoT. IDSs are more suitable for this case of systems. They can be considered as the last line of defense when other tools are broken. Another advantage of IDS is that they are variated and adaptable depending on needs. They can be enhanced with learning logic such as Machine Learning (ML) and Artificial Intelligence (AI) techniques in addition to other advanced technologies. IPSs are also important for IoT systems in order to act after the detection of a threat (using an IDS) and thus prevent the infection and the shutdown of these systems and their connections. Moreover, day by day, researchers are devoting more and more effort to ML-based IDSs, particularly for their ability to detect unknown/zero day threats as well as their low false alarm rate. Therefore, we focus in this thesis on the study and the implementation of an Intrusion Detection and Prevention System (IDPS) based on ML for the IoT ecosystem in order to immediately detect and respond to potential threats as soon as they occur. Such an IDPS must respect the different challenges imposed by the IoT context presented in Section 1.1.1.2.2 especially heterogeneity, interoperability, connectivity and resource constraints.

With the unbridled growth in the number of IoT devices and their deployment in all areas, for all industries in all sectors, the need to move from the use of proprietary solutions with custom hardware and software to a multi-industry standard solution is more necessary than ever. We are therefore focusing our security framework towards the oneM2M standard [One19] which is an international partnership project launched in 2012 by eight of the world's leading ICT standards bodies. OneM2M standard enables the communication between heterogeneous devices and applications by defin-

ing a common M2M Service Layer for the multi-industry M2M applications. OneM2M standardizes the IoT ecosystem with respect to the exiting worldwide networks and standards. Therefore, securing the IoT ecosystem can be done through securing the oneM2M standard. Indeed oneM2M provides a large panel of security mechanisms to protect the service layer itself as well as the communication between the oneM2M architecture layers. However, none of the specified techniques protect the IoT systems once the malicious user has gained access to the system and bypassed the first-line security measures. This represents a critical and unexplored topic for the oneM2M standard. We decided therefore to make our ML-based IDPS a second line of security for IoT systems based on the oneM2M standard to detect and prevent intrusions. To the best of our knowledge, the ML-based IDPS proposal detailed and discussed in this thesis represents the first one for the oneM2M service layer.

1.3 Contributions

The main contribution of this work is the design and the implementation of an IDPS based on ML to protect the oneM2M standard. To achieve this goal, we began by conducting a comprehensive review of the IoT security ecosystem in a paper published in the journal *IEEE Communications Surveys & Tutorials* entitled "**Network intrusion detection for IoT security based on learning techniques**" [CMZ+19]. This paper targeted three important areas: i) IoT, ii) security mechanisms and iii) machine learning techniques. It guides the reader in discovering the intersection of the three areas by providing all the necessary details to understand the security in IoT. The paper started by presenting and categorizing the IoT security threats as well as the traditional defense techniques with a focus on IDSs types. Then, it listed and discussed available tools that can be used to develop and/or evaluate NIDS: i) free datasets, ii) free and open source network sniffers and iii) open source NIDSs. Moreover, the paper discussed IoT powered NIDS, their architectures, deployments and implications for the heterogeneous systems. Learning techniques via ML classifiers were introduced. Then, NIDSs for IoT systems

deployed via learning techniques were reviewed, compared and evaluated. We comprehensively discussed existing state of the art, compared and evaluated performance of ML based IoT systems deployed to secure the networks. Finally, the paper concluded with a summary and a list of possible future directions.

As discussed in Section 1.2, considering the need for a standardized IoT ecosystem especially with the significant growth in the number of connected devices, we focus our work on the oneM2M standard. Therefore, we defined the security attacks that threaten the oneM2M service layer and proposed an abstraction for the oneM2M flows named *GFlow*. In order to implement a security mechanism based on ML, we used the *GFlow* abstraction for the creation of a oneM2M security dataset. This dataset was created with a real world platform and contains the oneM2M attacks as well as legitimate *GFlows*. Thanks to the generated dataset, we proposed the first generic IDS for the oneM2M service layer based on edge ML. In order to choose the best ML algorithm for our needs, we experimented various shallow algorithms in a binary and multiple classification. All these studies have been the subject of the publication "**An Intrusion Detection System for the OneM2M Service Layer Based on Edge Machine Learning**" [CMZS19] in the *International Conference on Ad-Hoc Networks and Wireless (Adhoc-Now)*.

A oneM2M IDPS based on ML was the focus of our paper "**A OneM2M Intrusion Detection and Prevention System based on Edge Machine Learning**" [CMZS20] published in *IEEE/IFIP Network Operations and Management Symposium (NOMS)*. In this work, we proposed an architecture for the oneM2M-IDPS. We detailed each module of our IDPS strategy. First, we described the data acquisition and features extraction module. Then, we proposed a new IDS based on three levels of ML detection. Furthermore, we enriched the IDS with two new modules: i) a prevention module to act against the detected threats and ii) a continuous learning module to provide an up-to-date IDPS that evolves with the evolution and emergence of new threats.

In the previous works, the IDS module used shallow or deep ML algorithms for the three different levels of detection in a binary and a multi-classification mode. In our journal paper "**Anomaly and Novelty Detection and Prevention for IoT Security: A**

Continuous Machine Learning Approach", submitted at the end of the PhD, we detect anomalies in the oneM2M standard on the one hand, and detect novelties in terms of threats on the other hand, in an intrusion detection and prevention platform. Regarding anomaly detection, we examined one-class classification algorithms (OCC) to detect behaviors that differ from the norm and we propose a new threshold determination method based on the decision tree algorithm. However, for the novelty detection, we extended the OCC's approach to i) detect new, unseen threats and ii) classify the already known ones. In addition, we designed the prevention workflow and we specified the prevention actions. Furthermore, we described in details the different scenarios that, autonomously or at human request, activate the continuous learning module which enables the continuous update of the IDPS system.

1.4 Organization of the dissertation

The rest of the dissertation is organized in five chapters which are further divided into multiple sections. An overview of these chapters is provided below:

In Chapter 2, we present and discuss the literature proposals dealing with the network IDPS in IoT systems. In Section 2.1, we start with the state of the art of IDPS for IoT that are not based on machine learning techniques. These works focus more on the deployment architecture and the detection strategy. After detailing each work separately, we discuss and compare them on the basis of their architectures (if it is distributed, centralized or hybrid), their detection methodologies (signature-based, anomaly-based or hybrid), their validation strategies (simulation or emulation) as well as their treated threats. Importance was also given to the strengths and weaknesses of each one of proposals. In Section 2.2, we dived into the IDPS based on ML techniques. As in the first part, we provide a brief description for each work, then compare and discuss their deployments, their methodologies, the datasets they used, the threats they dealt with, and the ML algorithms they employed.

In Chapter 3, we give an overview of the oneM2M standard (Section 3.1): its archi-

tecture and its security mechanisms defined in its specifications. Then, in Section 3.2, we concentrate on the oneM2M threats related to the service availability since oneM2M is, first and foremost, about services for M2M and IoT systems. We propose the taxonomy and implementation of the oneM2M threats. We finish this chapter with details about the creation of the oneM2M security dataset. We start Section 3.3 by presenting the state of the art of free datasets used for the IoT NIDS creation. As no datasets exist for the oneM2M standard, we create our own dataset with respect to the taxonomy of oneM2M threats presented in the section before. The dataset is based on a new abstraction proposal for the oneM2M flows. This abstraction is the foundation of our IDPS.

In Chapter 4, we present the different challenges and aims respected and guaranteed with our oneM2M-IDPS proposal namely interoperability, autonomy, scalability and resource constraints respect, modularity in design, adaptability and extensibility and finally the active and real-time reaction (Section 4.1). Moreover, in Section 4.2, we detail its strategy as well as the architecture and the design of each of its four modules: i) the data acquisition and features extraction module, ii) the IDS module, iii) the IPS module and iv) the continuous learning module.

In Chapter 5, we concentrate on the ML aspect of our IDPS. We implement our detection module with its three ML levels using the oneM2M dataset described in Chapter 3. We experiment with different ML and Deep Learning (DL) algorithms for each detection level in order to choose the most appropriate and efficient ones. We start this chapter (Section 5.1) by explaining our choice to adopt ML and DL techniques for the detection levels. Then, we present the metrics we relied on to evaluate the effectiveness of each algorithm in our oneM2M intrusion detection context, as well as the experimental environment. Furthermore, we study two main approaches: i) detection based on supervised ML algorithms (Section 5.2) and ii) detection based on one-class classification approach (Section 5.3). For each one we introduce the experimented algorithms (definitions, tools and frameworks), we explain the different stages of setting up the experiments and then report, compare and discuss the results. Regarding the

first approach, we chose supervised ML because the experiments conducted in the corresponding section fall within the context of classification where we know already the classes into which the model should categorize the inputs. This section experiments supervised ML for the three levels of our IDS, then study the effect of size and balance of training dataset on the detection process. In the last section of this chapter, we examine the one-class classification approach for both the first and second ML detection levels. In the first level, we highlight the detection of any behaviour different from normal (known for anomaly detection) as well as the techniques for determining the threshold of normality. In the second level, the one-class classification approach will be used to allow the detection of unknown threats as well as the multi-classification of already known threats (multi-class novelty detection).

In Chapter 6, we conclude the dissertation by summarizing the key contributions and discussing some directions for future work that can be followed to continue researching the exposed problems.

Chapter 2

State of the Art of Network Intrusion Detection Systems for IoT

Contents

Introduction

As discussed in Chapter 1, our goal is to build an IDPS based on ML for the oneM2M standard. To our knowledge, there is no work in the state of the art that proposes an IDS or an IPS for the oneM2M service layer. Thus, we focus on the literature proposals dealing with the network IDPS as these are the works closest to our problematic. It is

important to mention that most of the studies deal with intrusion detection and that few of them propose a prevention system; we will therefore use the term Network Intrusion Detection System (NIDS) more often to express even those with a prevention strategy. This chapter is divided into two main sections: the first one treats the state of the art of NIDSs for IoT that are not based on machine learning techniques. We will focus more on the architectures and the strategies of the proposed systems. Works are detailed and compared with a special focus on the advantages and disadvantages of each one. The second part will deal with the literature of NIDSs that use learning techniques in their detection process.

2.1 Network Intrusion Detection Systems

To get a better idea about IoT NIDS architectures and deployments, we discuss, in the following, relevant works from the state of the art of NIDS IoT security. We focus on their detection mechanisms, architectures and validation strategies. The reviewed IoT NIDS papers provide a comprehensive overview of the evolution of the domain from the first proposed solution [RWV13] to the present days. We start with detailed description of the authors' solutions. Then we summarize them in a comparative Table 2.1 with the advantages and disadvantages of each solution.

2.1.1 State of the art of NIDS for IoT

This section details each IoT NIDS proposal with a special focus on architectures, detection methodologies and treated threats.

Raza et al. [RWV13, AO17] designed and implemented SVELTE, the first IoT IDS. It is a real time intrusion detection system based on a hybrid signature accompanied by an anomaly based detection technique. The work meets the requirements of IPv6-connected IoT and concentrates on routing attacks such as spoofing and sinkhole. SVELTE considers IoT challenges and deploys lightweight IDS modules in resource con-

strained nodes and resource-intensive IDS modules at the Border Router (BR). It integrates three main modules: i) 6Mapper (6LoWPAN[1] Mapper) which gathers information about the RPL[2] network and reconstructs the network in the 6BR[3], ii) Intrusion Detection component that analyzes mapped data and iii) a mini-firewall (whitelist firewall for the IP-connected IoT that uses RPL as a routing protocol in 6LoWPAN networks) which is distributed and designed to offload nodes by filtering unwanted traffic before it enters the resource constrained network. SVELTE was implemented in the Contiki OS. It detects sinkhole attacks with 90% true positive rate (TPR) in a small lossy network and almost 100% TPR for a lossless network configuration. Unfortunately, DoS attacks can affect the solution [KCK+13] as well. Since IDS nodes use the network to transmit attack information, once DoS affects the network, detection of the attack fails.

Kasinathan et al. [KPSV13, KCK+13] studied DoS detection for 6LoWPAN developed as a part of ebbits, a EU FP7 project[4]. The supported architecture presented in Figure 2.1 is based on Suricata IDS[5]. The proposed architecture is considered centralized despite the distributed IDS probes. In fact, IDS probes which are external modules, sniff the network in promiscuous mode then send data to the main NIDS (based on Suricata) via wired connection. When the latter matches traffic with an attack signature, an alert is launched to the DoS protection manager. The protection manager analyzes the attempt with additional data collected from other ebbits managers. Moreover, it reduces the false alarm (incorrect detection genuine data: false positives plus false negatives) rate. This solution overcomes SVELTE limitations since the IDS mechanism does not depend on the network architecture so it cannot be affected by DoS attacks against the IoT network. The suggested framework DEMO in [KCK+13], is scalable and real-word applicable for most IoT systems. This work was evaluated using a penetration testing (PenTest) system called Scapy [KCK+13] which is more light-weight then

[1] IPv6 over Low-power Wireless Personal Area Network
[2] Routing Protocol for Low-Power
[3] 6LoWPAN Border Router
[4] The ebbits project is a European research project which deals with architecture, technologies and processes to enable mainstream enterprise incorporate IoT eco-system.
[5] Suricata [Fou17] is an open-source, multi-threaded signature-based NIDS.

Metasploit [noa17b]. The IDS adapts existing open source technologies. It starts with Suricata which is an open-source IDS and modifies it with IEEE 802.15.4 and 6LoWPAN decoders. Further, an additional detection module; FAM consists on frequency agility manager which analyzes channel occupancy states in real time to allow the network to become aware of the interference level and monitors attacks on Prelude which is a security incident and event management system (SIEM) to monitor the attack events or alerts. The Suricata engine triggers alerts according to the programmed rules; this solution could therefore detect different attacks depending on the rules developed.

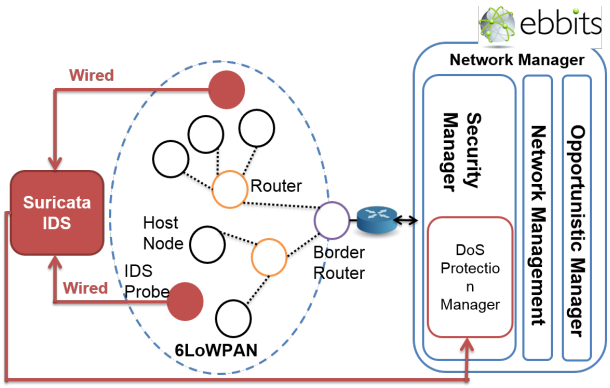

Figure 2.1: DEMO architecture

Jun and Chi [JC14] proposed an IDS for IoT systems based on Complex Event Processing (CEP) technology which is an emerging and efficient technology to filter and process real-time events[6]. It is a good solution for large volumes of messages with low latency and therefore, such a technology can be adapted to IoT needs. Jun and Chi evaluated on-line performance rather than off-line. The architecture of this solution is schematized in Figure 2.2. The system starts with collecting data (network traffic and event usage) from IoT devices, extracts events from sensed data, then performs security events detection using Event Processing Repository EPR[7] and CEP engine[8]. Finally, actions are performed by the action engine. Jun and Chin implemented their event-

[6]CEP technologies merge multiple data sources to interpret real time actions from complicated events or patterns.
[7]EPR is a repository of Event Processing Model statement EPM (a collection of events correlation).
[8]CEP engine analyzes a mass of events, identifies the most important ones, and produces actions.

processing IDS architecture using Esper (CEP engine for complex event processing and event series analysis). Their approach is CPU intensive, but consumes less memory. Indeed, it proved better real-time performance. For example, for 800k of data, CEP-based IDS consumes 62% of CPU, 730MB of memory and 422 millisecond processing time. However, traditional IDS like utilizes 57% of CPU, 1064MB of memory and 8688ms for processing 800k of data. It is interesting to note that, the framework is designed but not evaluated for any kind of attack detection.

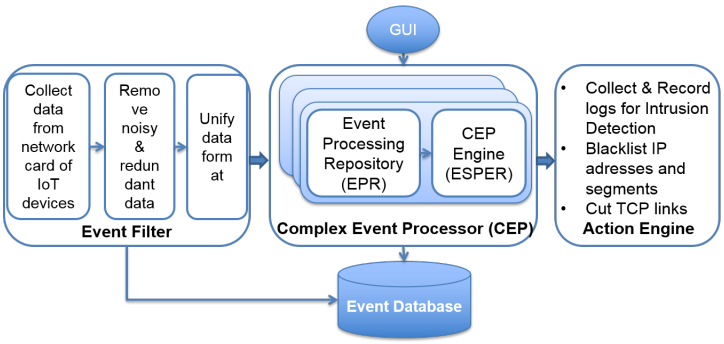

Figure 2.2: CEP-based IDS architecture for IoT

Cervantes et al. [CPNS15] detect the dangerous sinkhole attack on the routing services in IoT. They proposed Intrusion detection of SiNkhole attacks on 6LoWPAN for InterneT of ThIngs (INTI). It combines watchdog [WOM14], reputation [PTPB10] and trust strategies for detection of attackers. First, as a hierarchical structure, the nodes (grouped or separated) are classified as leaders. Then, the nodes can change role over the time based on network requirements. Each node monitors a number of transmissions performed by a superior node. If an attack is detected, an alert message is broadcast and a cooperative isolation of the malicious node is performed. Cervantes et al. gave importance for node mobility and network self-repair, which are limitations with Raza et al. [RWV13] approach. Their simulation results show sinkhole detection rate (DR) of 92% on 50 fixed nodes scenario and of 75% for 50 mobile nodes. Also, authors reported low false positives and false negatives compared to SVELTE.

Surendar and Umamakeswari [SU16] set up an Intrusion Detection and Response IoT System (InDReS) with 6LowPAN. InDReS uses constraint based specification tech-

niques to detect sinkhole attacks in RPL networks. In InDReS, sensor nodes are grouped into clusters under the supervision of an observer node. Observer nodes count the dropped packets of their adjacent nodes and assign a score to each of them using Dempster Shafer theory [Dem08, Sha76] to detect malicious node. The latter will be announced so that all the nodes co-operate to isolate it. Finally, the network reconstructs itself. Authors' strategy improves efficiency of some critical QoS metrics over the existing INTI scheme which is limited by the energy consumption and the packed drop ratio. Authors simulated their proposal on NS2 simulator.

Fu et al. [FYC+17] presented uniform intrusion detection system while considering the following two important points: i) varied heterogeneity of IoT networks; ii) IoT sensor and smart device resource constraint. Authors claimed that, their solution is the first one that benefits from automata theory to model and detect the intrusions of IoT networks. Based on an extension of Labelled Transition Systems [Tre96], they provided a uniform description of the IoT systems network traffic flows then compared the real-time action flows with Standard Protocol Libraries to detect and report: jam-attack, false-attack and reply-attack. The intrusion detection and prevention strategy uses an Event Monitor (which collects the network traffic and transmits data into digital files to the IDS Event Analyzer) and an Event Database (which implements three databases stored on the cloud: i) Standard Protocol Library[9]; ii) Abnormal Action Library[10]; and iii) Normal Action Library[11].). The intrusion detection is maintained by the IDS Event Analyzer which is composed of the following three basic models:

- Network Structure Learning Model: considers packet data as input, builds a general view of the network topology, distinguishes IoT devices IDs and sends them to the Action Flows Abstraction Model.

- Action Flows Abstraction Model: classifies the collected real-time packets from IoT

[9]Standard Protocol Library describes the standard protocols through Glued-IOLTS [FK14].
[10]Abnormal Action Library recognizes anomaly actions flows for the system.
[11]Normal Action Library contains possible action flows created from the Standard Protocol Libraries using the techniques of Fuzzing [TDB12] and Robustness Testing [LLL+10]

into message sequences then translates these messages to abstract action flows with the help of Standard Protocol Library.

- Intrusion Detection Model: compares the result of Action Flows Abstraction Model with the Abnormal Action Library. If it matches, the action flow is marked intrusive; otherwise, an anomaly detection method will be applied. For the latter, if the input transition sequence does not match entries of the Normal Action Library then an expert manual verification is needed (to avoid the false positives). If it is finally marked as safe, then the record is added to the Normal Library, otherwise, it is added to the Abnormal Action Library.

Regarding the Response Unit, it reports three types of attacks (jam-attack, false-attack and reply-attack) to a management station. Fu et al. experimented their solution on IoT experiment environment but unfortunately, they did not present detection rates.

Midi et al. [MRMB17] proposed Kalis which is claimed to be "the first approach to intrusion detection for IoT that does not target an individual protocol or application, and adapts the detection strategy to the specific network features". Kalis is a "network-based, hybrid signature/anomaly-based, hybrid centralized/distributed, online IDS that adapts to different environments". It can be deployed as a standalone tool on a separate external device (to overcome the fact that most IoT devices does not support software changes). It is an automatic knowledge-driven IDS which means that it chooses automatically detection techniques depending on collected network's features. Precisely, each attack could be done only in some IoT systems and not in others depending on the system features. For example, it is not possible to have replication attack in a single hop system. Kalis identifies even the presence, or not, of prevention techniques such as the use of cryptographic functions. Hence, it is effective and efficient regarding resource consumption. Kalis was implemented using Java on an Odroid xu3 development board and evaluated with real-world IoT devices. The system includes "small WSN[12] of six TelosB nodes, a Nest Thermostat, an August SmartLock, a Lifx smart

[12]Wireless Sensor Network

lightbulb, an Arlo security system, and an Amazon Dash Button". To sniff intermediate hops of data packets, Kalis was located near the middle portion of the WSN. Midi et al. replayed actual traces of network traffic of the prototype and added additional packets with 50 different symptom instances for each attack. The DR of Kalis is 91% with 100% accuracy, 0.19% CPU usage and 13978.62 KB memory consumption. Here, it is important to note that the traditional IDS use have around 48% DR with 75% accuracy, 0.22% CPU usage and 23961.06 KB RAM.

Granjal and Pedroso proposal [GP18] deals with the combination of detection and reaction to various attacks (especially DoS attacks) in Internet-integrated CoAP[13] WSN. Unlike most of the IDSs for WSN, their framework is applicable to IoT systems based on 6LoWPAN protocol. The IDPS considers different requirements like cross-layered detection, detection of external and internal threats, quick reaction and attackers blocking, the extensibility of the system, its reconfigurability as well as the interoperability between communication technologies. Authors proposed a hybrid, collaborative detection design where both WSN devices and 6LoWPAN border router collaborate in the protection process. Resource consuming tasks are executed at the border routers (6LBR) level. 6LoWPAN packets are analyzed on the basis of the IP source address and the number of messages received over a particular time frame. When an undesired message is detected, it is quickly dropped by the attacked device and in parallel a notification is sent to the 6LBR which enables various types of security measures like denying further messages originated at a particular address and warning other sensors to also start blocking the attacker. Hence, with respect of the security policy, the 6LBR decided if an action must be triggered. The proposed framework has been evaluated with the Contiki OS and examined mainly its impact on the memory of sensing devices, the energy consumption, the computational effort and the delay in reaction against attacks.

Aldaej [Ald19] paper is one of the studies that focuses more on prevention strategy than on intrusion detection. The author proposes a prevention scheme based on the results of an existing IDS in order to actively defend and prevent the intrusions from

[13]CoAP: Constrained Application Protocol

bandwidth attacks and more specifically DDoS. After each defined time threshold, the IDS nodes in the network send the collected information to a centralized IDS where log files are manipulated using forensic analysis and a report is generated. An Active Profile Database (APD) that provides a statistical analysis of each malicious node is updated continuously and used as input to the IPS algorithm. Depending on the malicious magnitude of the nodes and the predefined threshold, the prevention module updates and organizes its blacklist table. Aldaej's proposal contains also a reaction module that is invoked each time the blacklist is updated in order to react and maintain the performance on the IoT network. The most severe action is to isolate the malicious nodes.

2.1.2 Comparison and Discussion

In the following, a comparison of previously reviewed proposals for IoT NIDS is given (Table 2.1). As it can be noticed, most of the works deploy distributed architectures [RWV13, CPNS15, SU16, FYC+17]. This **type of deployment** is more suitable for IoT systems than centralized strategies [KCK+13, JC14] since the distribution of devices is an important IoT characteristic. However, centralized IDS detect better security attacks that involve a group of devices operating silently (without directly shutting down the network) than the distributed ones. For example, DDoS attacks are difficult to detect in a distributed deployment. For such attacks, a hybrid architecture such as in [MRMB17, GP18] is more appropriate. Hence, a distributed network analysis with a centralized general inspection is guaranteed (called also hierarchical strategy). Moreover, deployment of NIDS on the IoT system itself or in a separate external device is important. Kasinathan et al. [KCK+13] and Midi et al. [MRMB17] are the only ones who proposed their NIDS as a standalone tool. The adoption of such a strategy overcomes the lack of resources in IoT devices as well as the problem of proprietary IoT software that can not be changed. This protects the IoT system against overload. However, employing additional infrastructure adds complexity in case of network maintenance and system protection.

References	IDS deployment	Detection Methodology	Validation Strategy	Treated Threats	Advantages	Disadvantages
Raza et al. [RWV13, AO17]	Distributed	Hybrid (signature and anomaly based)	Simulation	Routing attacks like spoofing and sinkhole, selective forwarding and information alteration	• Resource constraints challenge is taken into consideration • Distributed mini-firewall for the IP-connected IoT devices is integrated • Flexible and can be extended to detect more attacks	DoS attack can affect SVELTE
Kasinathan et al. [KPSV13, KCK+13]	Centralized	Signature-based	Emulation	DoS attack	• False alarms reduction • IDS is deployed on additional infrastructure • Scalable and real-word applicable	Detected attacks depend on declared rules
Jun and Chi [JC14]	Centralized	Signature-based	—	—	• Real-time detection • Better real-time performance • Low memory consumption • IoT Massive data are taken into consideration	• CPU intensive • Detected attacks depend on declared rules
Cervantes et al. [CPNS15]	Distributed	Hybrid (trust and reputation strategy)	Simulation	Sinkhole attack	• INTI takes into consideration node mobility and network self-repair • Less false positive and false negative rate than SVELTE	IDS placements change over the time which can consume more resources.
Surendar and Umamakeswari [SU16]	Distributed	Specification based	Simulation	Sinkhole attack	• Resource constraints challenge is taken into consideration • Low average energy consumption • Low packet drop ratio • Instant network response against detected attacks	Cannot detect unknown attacks
Fu et al. [FYC+17]	Distributed	Hybrid (signature and anomaly based)	Emulation	Jam-attack, false attack and reply attack	• Heterogeneity of IoT networks is taken into consideration • Resource constraints challenge is taken into consideration • Low false positive rate	• The state based algorithm may cause "state space explosion" • Human intervention is needed for false positive alarms • DoS attack can affect the solution
Midi et al. [MRMB17]	Hybrid (centralized and distributed)	Hybrid (signature and anomaly based)	Emulation	DoS, routing and conventional network attacks	• Real-time detection • Lightweight in terms of CPU and RAM requirements • Dynamic self-adapting IDS • Automatic knowledge-driven IDS • Different IoT communication protocols and applications are taken into consideration • Deployable on border router or as a standalone tool	• High level perspective may not suitable for constrained compute objects • Kalis proposes compile time deployment which may not be feasible for resource constrained sensors which may even be resource constrained in comparison to WSN nodes
Granjal and Pedroso [GP18]	Hybrid (centralized and distributed)	Signature-based	Simulation	Attacks against 6LoWPAN and CoAP, as well as DoS	• Cross-layered detection • Detection of external and internal threats • Quick reaction and attackers blocking • Extensibility and Reconfigurability of the system • Interoperability between communication technologies	Detected attacks depend on declared rules
Aldaej [Ald19]	Centralized	Anomaly-based	Simulation	DDoS	• Magnitude of DDoS attack is taken into consideration • Prevention and Action module could isolate malicious nodes	Cannot detect unknown attacks

Table 2.1: Comparison of NIDS for IoT

Regarding **detection methodology**, both signature and anomaly detection are deployed. Each method has its advantages and drawbacks. Signature-based detection is efficient for known attacks; however, it cannot detect unknown attacks since the signature database must be updated which is time consuming. When the size of signature database increases, NIDS is required to compare the input with all the existing signatures. Anomaly-based methodology detects unknown / unseen attacks; however, it suffers from high false alarms. Consequently, hybrid detection such as [FYC+17, MRMB17] have been deployed as practical solutions.

Regarding the **validation strategy**, two important parameters are identified: simulation and emulation. Simulation models the behavior of the target system in a different environment. It provides the basic behavior of a system; it may not necessarily adhere to the rules of the original system. Emulation duplicates the exact same target behavior of the original system operating in a different environment. Therefore, emulation is more close to real life situation when compared with simulation [KCK+13, FYC+17, MRMB17]. Simulation is acceptable in IoT since the implementation of an IoT system requires a large number of physical devices to get closer to reality which is not an easy task for experimental research. The second point to discuss about validation is the evaluation metrics. Findings of the review show that researchers does not always provide the same metrics [Kum14, FEE+18] in their works evaluation which does not allow a fair comparison. Some of works did not provide experimental results like in [KCK+13] or did not even experiment their solution like in [JC14]. Evaluation metrics need to be fixed and processed in each work to have a reliable comparison even if the used metrics depend on the objectives and aspects on which each study focuses.

About the **treated attacks** in reviewed papers, as in Table 2.1, there is no work that takes into consideration all the threats at the same time. Normally NIDS based on anomaly or hybrid detection methodology should be able to detect all types of attacks but no one of the reviewed works concentrate on detecting the maximum attack types. [RWV13] is the only work which mentions that the solution could be expanded to detect

more than the experimented attacks.

IoT is an environment of coexisting protocols and technologies. Despite the heterogeneity aspect of IoT, [JC14, FYC+17, MRMB17, GP18] have the capability to detect attacks against multiple protocols. [RWV13, KCK+13, CPNS15, GP18] focus on intrusions in 6LoWPAN and RPL which are important techniques for IoT networks. Furthermore, complexity is high due to heterogeneity. Moreover, resource constraints challenge is considered by [RWV13, KCK+13, SU16, FYC+17, MRMB17, GP18]. Scalability, on the other hand, has been a subject of study in [KCK+13, JC14]. For [JC14], it was more concerning data scalability. Finally, [CPNS15] is the only proposal which considers mobility and connectivity. **Strengths and weaknesses** for each NIDS solution have been identified and illustrated in Table 2.1.

2.2 Network Intrusion Detection Systems based on Learning Techniques

Before moving from NIDSs for IoT to the ones based on learning techniques, we briefly introduce the machine learning ecosystem. Then, IoT NIDSs powered by learning techniques are surveyed, compared and discussed.

2.2.1 Learning Techniques

In year 1959 Arthur Samuel, a pioneer of Machine Learning, defined ML as "field of study that gives computers the ability to learn without being explicitly programmed" [Pug16]. It consists in the deployment of algorithms in order to obtain a predictive analysis from data (learning from examples). Deep Learning (DL) [SHM+16] is part of a particular family of ML methods based on learning high-level abstract representations. DL groups generic algorithms mimicking the biological functioning of a brain without being intended for a specific task. Technically, DL is the application of artificial neural

networks (ANNs) with multiple hidden layers. There are mainly three types of ML algorithms:

- Supervised learning is based on learning from labeled training data which means that training data includes both the input and the desired results.
- Unsupervised learning describes hidden structures from "unlabeled" data (no predefined classification or categorization in the observations).
- Semi-supervised learning is a combination of supervised and unsupervised machine learning methods. A semi-supervised algorithm learns from a training data that includes both labeled and unlabeled data. It is more like unsupervised learning with some prior knowledge about clusters / classes.

With the evolution, complexity and diversity of security attacks, researchers are focusing more on the use of artificial intelligence and machine learning for security threats detection. To do that, IDS must embed the machine intelligence and improve decision making capabilities [DLY+18]. Many studies apply ML in IDSs for traditional systems and prove promising results [AA15, BG16, FTM+17, HBH+17, WJ17, MVTP18]. Which is also the case for IoT IDSs as detailed in the different surveys [CMZ+19, ZMKdA17, BWH18].

2.2.2 State of the art of NIDS for IoT based on ML

Since our main focus is towards the deployment of intelligent IDS in IoT, we now singularly discuss NIDSs for IoT employing learning techniques. The rest of the chapter gives detailed description of each proposal and in the next section, researchers choices and results will be discussed.

Hodo et al. [HBH+16] used Multi-Layer Perceptron (MLP) which is a type of supervised ANN in an off-line IoT IDS. It is composed of three-layers with sigmoid transfer function in each of the hidden and output layers' neurons. The authors' analysis is built on internet packet traces and tends to detect DoS and DDoS attacks in IoT network.

The NIDS was tested on a simulation composed of four clients nodes and a server relay node. DOS/DDoS attacks were performed on the server node with 10 million UDP packets sent from a single host for DoS attack and with three hosts at wire speed for DDoS. The training dataset was composed of 2313 samples, out of which 496 samples were deployed for validation and 496 samples were used for testing. Overall attack detection accuracy was 99.4% with 0.6% false positive. Such results guarantee early detection of attacks and thus good network stability.

Nobakht et al. [NSB16] proposed a host-based IDS framework IoT-IDM for user-chosen smart devices in smart homes environment. IoT-IDM monitors traffic going through the devices to identify threats. The framework takes benefit of Software Defined Networking (SDN[14]) architecture with ML techniques to detect compromised hosts and mitigate these attacks by pushing the appropriate actions (like blocking the intruder or redirecting the malicious traffic) to underlying routers/switches. The SDN technology offers the opportunity of remotely managing the security which leads to provide the user of IoT-IDM with a Security as a Service (SaaS). Nobakht et al.'s solution is characterized with modularity in design. It is composed of five separate modules (Device Manager, Sensor Element, Feature Extractor, detection Unit and Mitigation Unit). Consequently, there is a flexibility to choose an ML algorithm from a set of given techniques. ML algorithms use learned signature patterns of known attacks to train the model. Besides the wide range of detected attacks, one of the drawbacks of IoT-IDM is that technically it cannot survey all home IoT devices due to high volume of network traffic with the detail that sensor elements are positioned on top of SDN controller. Consequently, IoT-IDM can only inspect chosen IoT devices that do not overload the SDN controller. Nobakht et al. tested IoT-IDM on a real IoT device which is the smart light bulb (Hue lights) and compare logistic regression and SVM (support vector machines). In unauthorized detection, the first one gives 94.25% of accuracy rate and 85.05% of recall rate against 98.53% and 95.94% for SVM.

[14]SDN abstracts network services. It separates control plane (the decision maker about data forwarding) from data plane (the responsible of sending the data).

Hosseinpour et al. [HVAP+16] proposed a novel real-time, distributed and lightweight IDS based on Artificial Immune System (AIS), in an effective combination of edge, fog[15] and cloud computing. It allows intelligent data processing at an intermediary level and thus reduces data transport to the cloud. Consequently, the processing takes place in hubs, routers or gateways. The AIS architecture of the IDS is composed of three parts:

1. A training engine: learns from an initial learning dataset and trains detectors (initialization phase of the AIS). This step is treated in the cloud layer since it needs complex and powerful processing units.

2. An analyzer engine: analyzes anomalies reported by the detectors to alert and reject the false positive signals. The authors use memory cell detectors and genetic algorithms as presented in previous works [HMR+13, HAFP14] to improve precision. This step requires more communication between the infected edge nodes and the main engine, hence the analyzer engine is deployed at the fog layer.

3. Detector sensors: detection logic is inserted in each node monitoring the network. The proposed IDS is doted with an intelligent and distributed detection where each type of attack, could be detected by a number of different detectors. If a threshold is reached, the anomaly will be reported to the analyzer engine, thus a deep intrusion alert is generated.

Following are the important work strengths: i) Fog computing enabled quality of service with low latency in data analysis; ii) Combination of lightweight analysis in fog layer with an advanced analysis in the cloud; iii) Detection of silent attacks such as botnet attacks using smart data strategy[16]; and iv) detection of unknown and zero-day attacks via AIS based on an online self training method with unsupervised machine

[15] Cisco introduced Fog computing concept to extend cloud computing at the network layer. The fog layer is between the IoT sensors and the cloud.

[16] Smart data strategy is "an active and intelligent data structure which facilitates the management of Big Data in IoT" [HVAP+16]

learning. Two datasets have been used to evaluate the lightweight IDS efficiency which are KDD-Cup99[17] [HB99] and SSH Brute Force from ISCX dataset [SSTG12]. According to experimental results, the three-layered proposed solution achieve 3.51% of false positive rate (FPR) with 98.35% of accuracy and 97.83% of precision.

Bostani and Sheikhan [BS17] suggested a real-time hybrid of anomaly-based and specification-based IoT IDS. It enables the detection of sinkhole and selective-forwarding attacks in 6LowPAN networks and can be extended to detect blackhole, rank and wormhole attacks. This IDS works mainly in two steps: specification detection in the router level and anomaly detection in the root level. For the first one, the routers analyze features locally from network traffic and host nodes. The results on the first step are sent to the root node for the second step and removed from routers to ensure lower consumption of memory and CPU cycles. The second step is the global intrusion detection where anomaly-based analysis is performed on incoming data packets at the root node. This step employs the unsupervised optimum-path forest algorithm (OPF) [RCF09] to create clustering models for each source node router. With a MapReduce architecture platform, a parallel, distributed execution of anomaly detection according to clustering models is ensured. The final decision about tagging a suspicious behavior as an attack is done with a voting mechanism. The proposed system neither uses additional control messages, nor makes use of additional infrastructure. Consequently, it saves on communication and setting cost compared to other IDS. Authors evaluated the proposed technique on their own simulation tool. They prove appropriate real-time detection results with three main experiments each one done with ten simulations: the first experiment deals with values of evaluation criteria, the second experiment tackles the scale of the networks (small and medium size) to confirm independent scale-network IDS and the third one proves the possibility of extending the detected attacks such as wormhole. The experimental results of simulated scenarios showed that when both sinkhole and selective-forwarding attacks were launched simultaneously, the proposed hybrid method can achieve true positive rate of 76.19% and FPR of 5.92%. However,

[17] More details about KDD-Cup99 are in Section 3.3.1

for wormhole attack the rates are 96.02% and 2.08%, respectively.

Bostani and Sheikhan resumed in [SB16, SB17] the same architecture as given in [BS17] (i.e., based on distributed MapReduce Model). They proposed an anomaly and misuse agents with supervised and unsupervised optimum-path forest model instead of anomaly and specification based detection. They also reduced dataset features with a hybrid feature selection algorithm which is built on mutual information and binary gravitational search algorithm.

Pajouh et al. [PJK+16] presented an anomaly IDS built with Two-layer Dimension Reduction and Two-tier Classification (TDTC) for IoT Backbone. They concentrated mainly on low-frequency, common attacks: User to Root and Remote to Local attacks while their experiments were based on NSL-KDD[18] dataset [noa16]. Pajouh et al. deployed a two-layer dimension reduction to limit dataset's high dimensionality:

- The first layer benefits from an unsupervised technique which is Principal Component Analysis (PCA) for feature dimension reduction (combine dataset features to construct new ones). So for NSL-KDD, the overhead complexity was reduced in TDTC since only 35 out of 41 data set features were used.

- The second layer uses a supervised technique: Linear Discriminant Analysis (LDA) to make PCA reduced features better for classification and to improve the speed of intrusion detection. After analyzing the dataset classes, LDA finishes with two dimension dataset for NSL-KDD.

This dimension reduction decreases the false positive, the DR and the computational complexity. The second step is the multilayer classification where TDTC uses Naive Bayes (NB) and Certainty Factor version of K-Nearest Neighbor (CF-KNN) to classify inputs. Pajouh et al. started with NB for anomaly detection then results are refined with CF-KNN. Their work proved computation reduction of about ten times with faster detection and less resource requirements. They achieved a DR of about 84.86% for binary classification with 4.86% of false alarm.

[18] More details about NSL-KDD are in Section 3.3.1

Lopez-Martin et al. [LMCSEL17] claimed ID-CVAE to be the first to apply conditional variational auto-encoder (CVAE) and the first to perform feature recovery in NIDS. CVAE, which is considered as an unsupervised technique, was trained in a supervised strategy where class labels are integrated within the decoder layers. Instead of using a threshold to identify intrusions, their anomaly-based method is based on a discriminative framework that utilizes intrusion labels to reduce the reconstruction error. Another key strength in their study is that ID-CVAE performs only a single training step to generate only one model from multiple trainings depending on the number of different labels like in variational auto-encoder (VAE). This characteristic makes the ID-CVAE a suitable option for IoT systems due to the efficiency in computation time, flexibility and accuracy results. The selected dataset for ID-CVAE training and testing was a refined version of NSL-KDD. It ended with 116 features and 23 possible labels. Lopez-Martin et al. achieved 80.10% of accuracy and recall and 81.59% of precision, which they show to be better than the results of well-known algorithms like random forest (RF), linear SVM, multinomial logistic regression and multi-layer perceptron. The authors insisted on the efficiency of their feature reconstruction algorithm and proved that the recovery of missing categorical features reached 99%, 92% and 71% of accuracy with three, 11 and 70 values, respectively.

Thing [Thi17] analyzed IEEE 802.11 network threats and proposed an anomaly network IDS to detect and classify attacks in IEEE 802.11 networks. This work is considered as the first work that employ deep learning algorithms for IEEE 802.11 standard. Thing experimented Stacked Auto-encoder (SAE) architecture with both two and three hidden layers. The author experienced different activation functions for the hidden neurons. To test his strategy, he used a dataset generated from a lab emulated Small Office Home Office (SOHO) infrastructure. He achieved an overall accuracy of 98.66% in a 4-class classification (legitimate traffic, flooding type attacks, injection type attacks and impersonation attacks).

Diro et al. [DC17] proposed a DL approach based on fog computing to detect known and unseen intrusion attacks. Known attacks represent 99% which leads to affirm that

zero-day attacks are crafted with small mutations in the old ones. Therefore, multi-layer deep networks enhance small changes awareness (in a self taught algorithm with compression capabilities) compared to shallow learning classifiers. Distributed DL approach is based on distributing the dataset to train each sub-dataset locally and rapidly than share and coordinate the learning parameters with neighbors. So the architecture ends with a master IDS which updates the parameters values of the down distributed IDSs and keeps synchronization. The studies show that the distributed parallel DL approach realize better results in accuracy than centralized DL NIDS and also than shallow machine learning algorithms. To train the models and evaluate the IDS, Diro et al. used NSL-KDD dataset after adding some modification on it to finish with 123 input features and 1 label. As results, they obtained multi-class detection consisting 4 labels (normal, DoS, Probe, R2L.U2R) to achieve 96.5% DR and 2.57% of false alarms for deep model in comparison to shallow classifier achieving 93.66% detection and 4.97% false DR. They also noted an increase in the overall detection accuracy while adding the number of fog nodes from 96% to 99%. The proposed approach took longer training time; however, real detection was fast and accurate.

Prabavathy et al. [PSS18] proposed a novel fog computing based intrusion detection technique using Online Sequential Extreme Learning Machine (OS-ELM). The distributed security mechanism respects interoperability, flexibility, scalability and heterogeneity aspects of IoT systems. The proposed system is composed of the following two major parts:

1. Attack detection at fog nodes: Prabavathy et al. use OS-ELM algorithm to detect intrusions in fog nodes. The IoT network is divided into virtual clusters where each cluster corresponds to a group of IoT devices under a single fog node. The OS-ELM classifies the incoming packets as normal or an attack. ELM is a single hidden layer feedforward neural network characterized by its fast learning phase. The input layer weights and hidden layer bias values are randomly selected to analytically deduce the output weights using simple matrix computations. However the online

nature of OS-ELM favors a streaming detection of IoT attacks.

2. Summarization at cloud server: to have a general idea about the global security state of the IoT system, detected intrusions are sent from the fog node to the cloud server. After the analysis and the visualization of the current state, Prabavathy et al. propose two actions; i) predict next attacker action using the attacker plan recognition approach; or ii) identify fog node geographical position based multi-stage, and DDoS attacks. Hence, an intrusion response can be activated.

Prabavathy et al. proposed a proof of concept to evaluate their proposal. They implemented OS-ELM using MATLAB and NSL-KDD as benchmark dataset. Authors claimed high accuracy and response time. They achieve 97.36% accuracy with reduced false alarm rate 0.37%. The DR with the fog node strategy was 25% faster when compared with cloud based implementation. An important advantage is that new online data can be incorporated in the learning process, which is not the case for ANN and NB.

Rathore et Park [RP18] deployed a novel fog detector using ELM-based Semi-supervised Fuzzy C-Means (ESFCM) NIDS. This distributed IDS handles geographically distributed and low-latency IoT detection for limited resources and networks via fog computing. Supervised ML does not detect unknown attacks despite its good accuracy. Unsupervised ML has lower accuracy but, has the capability to detect unknown / zero-day attacks. Hence, Rathore et Park proposed a semi-supervised approach using the supervised and unsupervised ML for labeled and unlabeled inputs. For the unsupervised learning, Fuzzy C-Means (FCM) was the chosen algorithm (one of the widely used in clustering). FCM selects unlabeled data and assigns each input to one or more clusters with several degrees of membership. While the supervised part deploys Extreme Learning Machine (ELM) for effective and efficient detection. Hence, the authors proposed an ESFCM classification where Semi-supervised Fuzzy C-Means (SFCM) works with ELM classifier for a faster detection of known and unknown attacks. The IDS starts by generating a model (M) after having trained the ELM classifier on labeled dataset. Then, SFCM algorithm learns from both, labeled and unlabeled data to assign a degree

of membership to the unlabeled inputs. The unlabeled instances that have better opportunity to belong to one class, are then classified using the trained model M and added to labeled data according to defined threshold. Moreover, the remaining unlabeled data are re-clustered with SFCM and retrained with ELM until all instances are assigned. Finally, a trained model is generated for labeled and unlabeled data. Two types of evaluation for the proposed algorithm were established using the NSL-KDD dataset after scaling and preprocessing; i) a comparison between the authors' distributed solution and a centralized cloud-based framework; and ii) the effectiveness of the ESFCM was compared with traditional machine learning methods in terms of standard measures. Results show better performance of 11 ms in terms of detection time and 86.53% accuracy.

Moustafa et al. [MTC18] proposed an ensemble network intrusion detection technique based on established statistical flow features to mitigate malicious events, particularly botnet attacks against DNS, HTTP and MQTT protocols utilized in IoT networks. Their solution can be divided into three steps:

1. A set of features are extracted from the network traffic protocols MQTT, HTTP and DNS protocols via deep analysis of the TCP/IP model. Authors employed Bro-IDS tool for the basic features and developed a novel extractor module (which works simultaneously with Bro-IDS) to generate additional statistical features of the transactional flows.

2. A feature selection step where correlation coefficient is applied on result features to extract the most important ones. This step enables the reduction of computational cost of NIDS.

3. An ensemble method where the network data is distributed with AdaBoost (Adaptive Boosting) algorithm. Then, Decision Tree (DT), NB and ANN ML algorithms are deployed to detect attacks. The choice of the classification techniques is justified by calculating the correntropy measure. The AdaBoost method enhances the performance of the detection compared to separate ML algorithms. It can deal

with the small differences of the feature vectors via computing an error function. The error function is assigned to each instance of the distributed input data to learn and decide which learners can correctly classify each instance.

To extract the best features and evaluate the proposed ensemble technique, Moustafa et al. used the UNSW-NB15[19] [MS15] and NIMS botnet datasets [noad] with simulated IoT sensor data. Experiments results have high DR and a low FPR compared to existing state-of-the-art techniques. The ensemble strategy achieved between 95.25% and 99.86% of DR and 0.01% to 0.72% of FPR.

Nguyen et al. [NMM+19] came with an autonomous self-learning anomaly-based IDS (DÏoT). Their solution is composed of a Security Gateways that monitor the system devices and an IoT Security Service (which could be a service provider) responsible for detecting anomalies in a device-type-specific mode. In other words, devices of the network are autonomously clustered into types on the basis of their manufacturer's hardware and software configurations. Then anomaly models will be generated for each device type. Nguyen et al. claimed to be the first to use the distributed federated learning approach in anomaly IDS. Models are trained locally in each Security Gateway then aggregated into a global model at the Security Service. The used ML algorithm is Gated Recurrent Units (GRU) which is an recurrent neural network (RNN) able to train efficiently with few training data according to the authors. Consequently, final GRU models are the result of a collective learning collected from the different Security Gateways while preserving privacy. This IDS solution is communication-efficient and seems to be suitable for distributed systems like IoT. The authors evaluate their proposal in a real-world smart home deployment (with more than 30 off-the-shelf IoT devices) in the detection of the Mirai malware. They show that DÏoT has a DR of 95.6% with no false alarm in 257ms (which is fast).

Illy et al. [IKMM+19] proposed a fog-to-things architecture for their IDS. They deploy the detection process on two levels: the fog and the cloud layers of the system.

[19] More details about UNSW-NB15 are in Section 3.3.1

This architecture allows the authors, on the one hand, to address their computationally intensive ML detection caused by ensemble learning (a combination of ML algorithms). On the other hand, it allows low latency detection thanks to fog detection and thus fast response. Therefore, an anomaly detection is first performed in the fog layer, if the traffic is identified as an attack, an alert will be sent to the security administrator and an additional analysis is processed in the cloud in order to classify the type of attack and provide it to him/her. Illy et al. tested different ML combinations in a multiexpert mode as well as in a multistage strategy. Their evaluation was made on the NSL-KDD dataset and achieved 85.81% and 84.25% of overall accuracy for binary and attack classification respectively.

2.2.3 Comparison and Discussion

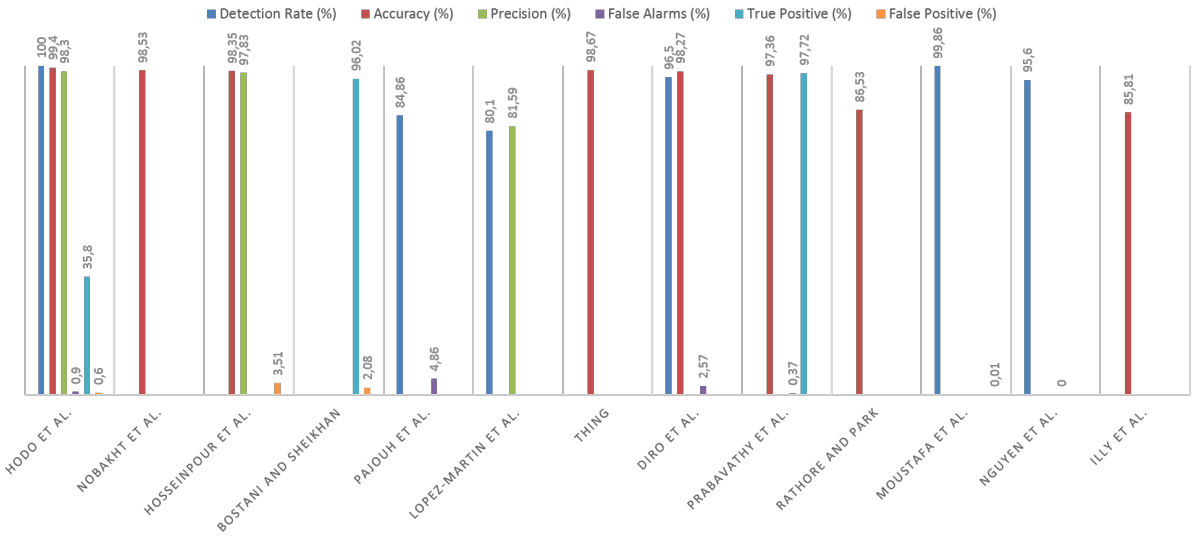

Figure 2.3: State of the art intrusion detection results

As presented in the previous section, many researchers give a special interest for IoT powered NIDS via ML algorithms. A comparison between the previously detailed proposals is illustrated in Table 2.2 where we focus mainly on IDS deployment, detection methodology, used dataset, treated threats and ML used algorithms.

References	IDS deployment	Detection Methodology	Used Dataset	Treated Threats	ML algorithms
Hodo et al. [HBH+16]	—	Anomaly-based	Simulation	DoS / DDoS	Multi-Layer Perceptron (MLP)
Nobakht et al. [NSB16]	—	Anomaly-Host based	Real IoT devices (Hue lights)	Unauthorized access	Logistic Regression vs SVM
Hosseinpour et al. [HVAP+16]	Distributed	Anomaly-based	KDD99 and SSH Brute Force from ISCX	Botnet attack	Artificial Immune System (AIS)
Bostani and Sheikhan [BS17, SB16, SB17]	Centralized / Distributed (Big Data architecture MapReduce)	Hybrid: Anomaly-based for the centralized part and specification-based for the distributed part [BS17]	Proprietary Simulator + NSL-KDD	Sinkhole / Selective Forwarding in 6LoWPAN and can be extended to Blackhole rank and Wormhole	Unsupervised Optimum Path Forest (OPF) in [BS17]
		Hybrid: Anomaly-based for the centralized part and misuse-based for the distributed part [SB16, SB17]			Supervised & Unsupervised Optimum Path Forest (OPF) in [SB16, SB17]
Pajouh et al. [PJK+16]	—	Anomaly-based	NSL-KDD	Low frequency attacks (such as U2R, R2L)	{Unsupervised Principal Component Analysis (PCA) + Supervised Linear Discriminant Analysis (LDA)} for Feature Reduction, & {Naive Bayes (NB) + Certainty Factor version of K Nearest Neighbors (CF-KNN)} for Classification
Lopez-Martin et al. [LMCSEL17]	—	Anomaly-based	NSL-KDD	DoS / R2L / U2R / Probe	Conditional Variational Auto-Encoder (CVAE)
Thing [Thi17]	—	Anomaly-based	Generated Dataset from lab SOHO	IEEE 802.11 attacks (flooding, injection and impersonation)	Stacked Auto-Encoder (SAE)
Diro et al. [DC17]	Distributed	Anomaly-based	NSL-KDD	DoS / R2L.U2R / Probe	Multi-Layer Deep Learning
Prabavathy et al. [PSS18]	Distributed	Anomaly-based	Emulation + NSL-KDD	Probe / R2L / U2R / DoS	Online Sequential Extreme Learning Machine (OS-ELM)
Rathore and Park [RP18]	Distributed	Anomaly-based	Simulation + NSL-KDD	Probe / R2L / U2R / DoS	ELM-based Semi-supervised Fuzzy C-Means (ESFCM)
Moustafa et al. [MTC18]	Distributed	Anomaly-based	UNSW-NB15 + NIMS + Simulation	Botnet attack	AdaBoost ensemble method using three techniques of DT, NB and ANN
Nguyen et al. [NMM+19]	Distributed	Anomaly-based	Real-world smart home	Mirai Malware	Gated Recurrent Units (GRU)
Illy et al. [IKMM+19]	Distributed	Anomaly-based	NSL-KDD	DoS / R2L.U2R / Probe	Different ML combinations in a multiexpert mode as well as in a multistage strategy

Table 2.2: Summary of NIDS for IoT based on learning techniques

While reviewing [HBH+16, NSB16, PJK+16, LMCSEL17, Thi17], we observe a lack of details on the **architecture deployments**. The above proposals concentrate on ML mechanisms for intrusion detection without discussing the architecture designs. Solutions of Hosseinpour et al. [HVAP+16], Bostani and Sheikhan [BS17, SB16, SB17], Diro et al. [DC17], Prabavathy et al [PSS18], Rathore and Park [RP18], Moustafa et al. [MTC18], Nguyen et al. [NMM+19] and Illy et al. [IKMM+19] deployed distributed architecture for intrusion detection, most suitable for IoT needs. In fact, IoT systems are distributed since they are composed of geographically distributed nodes. Such a criterion plays an important role when choosing the ML algorithm. Depending on where and how we want to deploy our NIDS, researchers have to pay attention and identify the algorithm for resource constrained smart objects. Hence, training an intensive algorithm on a limited node may not be feasible. However, the intensive task can be transferred to the fog or cloud layer. The best strategy is to process resource consuming tasks in cloud/server part, and execute lightweight parts in the IoT edge. It is the case in fog computing based NIDS such as in [HVAP+16, BS17, SB17, DC17, PSS18, IKMM+19]. The proposed solutions take advantage of the cloud or the fog layer for ML model training and use fog nodes and/or edge nodes for intrusion detection. Fog and edge based intrusion detection enables coordination for better, low latency detection (near to the source of data). It reduces network bandwidth consumption since partial data is sent at cloud. Only some details are reported to the centralized, source-intensive part of the IoT system to summarize and detect distributed attacks. Fog and edge concepts enable autonomic and parallel distributed attack detection [DC17]. [RP18, HVAP+16] claimed that, their proposals can be deployed in distributed IoT systems but they concentrated more on the distribution of traffic network data.

Regarding the **datasets**, recent information is necessary to train and evaluate IoT NIDS. Proposals such as [PJK+16, LMCSEL17, DC17, IKMM+19] are based on NSL-KDD dataset; non IoT dataset. NSL-KDD neither support IoT protocols like 6LowPAN, Zigbee, CoAP, nor IoT architecture and principles like mobility and heterogeneity. Details about the existant datasets are in Section 3.3.1. Unfortunately, **no IoT-dedicated NIDS**

dataset exists which explains the use of NSL-KDD. [HBH+16, NSB16, Thi17, NMM+19] evaluate their proposal with their own data. However, [HVAP+16, BS17, PSS18, RP18, MTC18] used a combination of real and synthetic data or simulate the environment.

Moreover, most of the studied researches are made to protect IoT systems from precise **types of attacks** mainly DoS, U2R, R2L and Probe since they get inspired from NSL-KDD dataset. Thing is the only proposal which concentrate specially on IEEE 802.11 attacks. However, unsupervised and semi-supervised ML solutions are evaluated against specific attacks. Meanwhile, they are able to detect unknown attacks such as in [HVAP+16, SB17, LMCSEL17], and [RP18].

The last important point to discuss is about **ML used algorithms**. ML algorithms such as SVM, NB and ANN were deployed separately and combined to improve outcome in general systems [FZHH14, LKT15]. As evoked in [MTC18], science started with applying each machine leaning algorithm separately than the trade on combining the algorithms in the same system takes place.

Hodo et al. [HBH+16], Nobakht et al. [NSB16], Hosseinpour et al. [HVAP+16], Lopez-Martin et al. [LMCSEL17], Thing [Thi17], Diro et al. [DC17], Prabavathy et al. [PSS18] and Nguyen et al. [NMM+19] use ML algorithms separately. However, all the rest use a combination of algorithms. Hodo et al. [HBH+16] use MLP, a part of ANN family in an off-line detection. Nobakht et al. [NSB16] executed a feature reduction heuristically and experimented two ML algorithms; LR and SVM for intrusion detection. LR is gradient descent which aims to find out the optimal parameters of a LR model. The accuracy of the obtained linear model with LR was less interesting than the non linear model of SVM (96.2% for LR whereas SVM achieves 100%). For Hosseinpour et al. [HVAP+16], they used AIS which is an unsupervised ML algorithm that is inspired from human immune system. It is characterized by a multi-layered protection structure. First line of defense responses immediately to previously seen problems then a non specific protection for unknown attacks is processed. It does not need prior knowledge of specific outsiders. Another important point for AIS is the memory aspect; AIS is efficient in unknown attacks detection. Authors achieve 98.35% of accuracy and

97.83% of precision which are remarkable results as noticed in Figure 2.3. However, AIS training needs resource which is why Hosseinpour et al. proceed it in the cloud layer. Prabavathy et al. [PSS18] took advantage of ELM algorithm in their intrusion detection proposal. They operated an online version of ELM (OS-ELM) for a real time analysis. Compared to ANN and NB, authors achieved better accuracy (97.36%) with lower FPR (0.37%) in a lower period of time (25% faster). A major advantage of OS-ELM is that it can incorporate new data online for learning which is not possible with the other compared algorithms.

Regarding the solutions with DL [LMCSEL17, DC17, Thi17, NMM+19], the good results (as we can notice in Figure 2.3) could be justified with the following DL advantages: i) training stability and generalization of DL; ii) its ability to reach a high accuracy rate if there is enough data and time [MRB+18]; iii) DL is a self-learning algorithm which means that it does not need manual feature engineering [RP18]; iv) DL extracts complex and non linear hierarchical features from training data of high dimension [DC17]. Lopez-Martin et al. [LMCSEL17] used CVAE which is a generative model based on VAE concepts. CVAE relies on two inputs: i) the intrusion features and ii) the intrusion class labels, instead of using only the intrusion features as input as in VAE. CVAE is better in flexibility and performance. The authors chose CVAE for its ability in feature reconstruction: its ability to retrieve missing features from incomplete dataset. Despite their use for an unsupervised DL algorithm, they benefit from labeled data in training phase for a deviation-based NIDS. Martin-Lopez et al. ensured a good computational time with a good flexibility though generating one model from multiple trainings in only one single training step. They proved better accuracy 80% than linear SVM (75%), MLP (78%) and RF (73%) algorithms. Meanwhile, Diro et al. [DC17] used Multi-layer DL algorithm in a distributed strategy which gives better results compared to centralized DL (99% vs 96% of accuracy). It is true that training phase takes longer time, but real-time detection is faster and more accurate. The authors chose multi-layer DL since it is the most prevalent form of DL. It shows training stability with a significant scalability on big data concept. Moreover, Diro et al. compared DL with ML in the

distributed context and proved that accuracy of the deep model is greater than that of shallow model (multi-class detection accuracy increase from 96.75% to 98.27%) and false alarms rate are lower for DL (from 4.97% for ML to 2.57% with DL in multi-class detection). About the DL used version of Thing [Thi17], he experimented SAE with two and three hidden layers but does not provide any primary choice arguments. SAE is a neural network built by stacking multiple layers of sparse auto-encoders. The output of each layer forms the input to the successive layer. Its hidden layers reduce the feature dimensionality and produce new set of features [AK16]. These new features are learned in cascade depths to improve precision. Nodes in the input and the output layer of SAE are the same [MVTP18]. The proposed solution achieved good accuracy results (98.66%) compared to J48 (an implementation of DT). The 2-hidden-layer model had a better performance over the 3-hidden-layer model. Nguyen et al. [NMM+19] are the only researchers that use federated learning for NIDS. It is an interesting strategy in DL that enables the learning from a distributed architecture since it guarantees the aggregation of the learned patterns. The authors deploy GRU as a DL network in the edges of the system for its efficiency in training with few data. GRU is a computationally less expensive RNN [WAB+17]. Thanks to this strategy, Nguyen et al. achieve 95.6% of DR. This was about DL proposals.

Another strategy in the use of ML algorithms that is taking more and more attention is the combination of different algorithms in the same system [PJK+16, MTC18, RP18, SB16, SB17, IKMM+19]. Pajouh et al. [PJK+16] used two simple ML techniques which are NB and K-nearest networks (KNN) for more exact class labels. NB is applied in the first place to identify anomalies. Then normal behaviors will be analyzed with KNN to refine normal instances. NB assumes the independence of all the characteristics of each sample in the given class label. It has the ability to measure good similarities of rare instances in the aim to handle imbalanced data. KNN uses a bucketing technique [FBF77] to accelerate the classification task. On the other side, Pajouh et al. applied dimension reduction before running the classification. To do so, they deployed both LDA (a supervised dimension reduction technique) and PCA (an unsupervised dimension reduction

technique). PCA provides a lower feature space by generating uncorrelated features from the initial correlated ones. LDA reduces the dimension of large working datasets by examining class labels. Hence, these two dimension reduction techniques represent a good strategy to i) reduce computational needs which is perfect for IoT systems and ii) fast the detection with less errors which is perfect for intrusion detection. Pajouh et al. achieved 84.86% of DR on NSL-KDD however Moustafa et al. [MTC18] succeeded to have 99.86% which is an impressive value. Their idea is based on an AdaBoost ensemble learning method which uses three ML techniques, namely DT, NB and ANN. The combination of these algorithms is done in a distributed parallel way. Data is divided into N sets (according to an error function) and each data subset will be treated with a chosen algorithm to finally update the distribution. Such a logic is guaranteed thanks to AdaBoost flow. Moustafa et al. applied also feature selection before starting the classification. By analyzing the treated attacks using correntropy, authors noticed that there are small variations between legitimate and suspicious vectors. Thus the ML algorithms to be used, should classify these small differences. That's how DT, NB and ANN were chosen. DT [MTC18] has multiple advantages while classifying network data. It selects important feature, prepare learning data points easily and manipulates directly the values of features. Even if a non-linear relations exist between parameters, DT performance is not affected. DT could be extended to better take into account unknown threats as proposed in [BC06]: it proposed a solution to deal with new instances that are not taken into account in training by assigning a new default class to the test instance that is not covered by the tree rules instead of assigning the default class with the one that contains the most elements. NB [MTC18] is known for its good detection of abnormal inputs. It needs less training data and scales linearly predictors and features values. It is simple in parameters optimization. ANN [MTC18] has many merits. It demands less formal statistical training and defines complex non-linear correlations between dependent and independent variables. Furthermore, it enables the detection of all possible interactions between predictors and variables. The third type of ML algorithms combination is presented in [RP18]. Rathore and Park composed

the semi-supervised SFCM with the unsupervised FCM algorithm which clusters inputs data and the supervised ELM classifier. FCM is one of the widely used techniques in unsupervised learning. It captures hidden and visible data structures. However, ELM algorithm [HZS06] is originally created to train single hidden-layer feedforward neural networks (SLFNs). ELM is efficient and doted of fast learning capacity in highly dynamic environment like IoT systems. Consequently, Rathore and Park achieved faster detection (11ms) with a better accuracy rate 86.53% comparing to traditional ML in their framework with the advantage of labeled and unlabeled data classification. About Sheikhan and Bostani solutions, they used in their works [BS17, SB16, SB17] mainly OPF which is an efficient graph-based ML. They used two variants of OPF; i) OPFC (OPF Clustering) which is an unsupervised ML and ii) MOPF (Modified OPF) which is a supervised algorithm. OPF main strength [PF09] is that it does not make any assumption about the shape of classes. The authors use OPFC to project clustering models on a MapReduce architecture. MOPF is used in a misuse-based detection engine with a feature selection module. The proposed solutions are simple and fast classifiers, they are parameter independent and originally support multi-class problems [PF09]. Finally, Illy et al. [IKMM+19] apply ensemble learning in a multiexpert combination as well as in a multistage strategy with different algorithms like DT, KNN, MLP, etc. and succeeded to reach 85.81% of accuracy.

Conclusion

As shown in this chapter, many IDS solutions for IoT are possible with or without ML, although ML techniques allow the detection of unknown threats and reduce the false alarms rate. NIDSs literature shows that many criteria could influence the IDSs results: i) the deployed architecture (centralized or distributed), ii) the detection methodology (signature-based, anomaly-based or hybrid), iii) the validation strategy, iv) the detected threats, v) the used dataset for training ML models, vi) the used ML algorithm for anomaly-based detection and finally and most importantly the whole strategy of de-

tection and prevention. The IoT context is a challenging one with several constraints that need to be respected such as heterogeneity, resource constraints and connectivity. To our knowledge, despite the many IDS solutions, there are no works in the state of the art that propose an IDS or an IPS for the oneM2M service layer. OneM2M standard is an important path towards a standardized IoT ecosystem with respect to the exiting worldwide networks and standards. Thus, it is important to guarantee its protection. Consequently, we will focus in this thesis on the proposal of an IDPS for the oneM2M service layer while respecting the IoT constraints and addressing multiple gaps that were neglected in the state of the art of NIDSs for IoT. We will try to benefit from the strengths of the works presented in this chapter and combine the various advantages into a single IDPS solution while avoiding their weaknesses. We will work on the autonomy, the scalability, the modularity and other aspects of the IDS. We will also focus on the prevention strategy (which has been underestimated in the state of the art) and, most importantly, we will deal with the continuous learning and improvement of the IDPS throughout the lifetime of the IoT system. Before tackling the strategy and deployment of the IDPS, it is important to start by analyzing the oneM2M standard that we intend to protect.

Chapter 3

OneM2M Standard Security and Dataset Creation

Contents

Introduction

The number of connected Things is growing at a frantic pace, which has led to vertical, proprietary IoT solutions. To ensure a horizontal IoT cross-industry interoperability,

eight of the word's leading ICT standards bodies introduce the oneM2M standard. Its main goal is to satisfy the need for a common Machine-To-Machine (M2M) Service Layer that guarantees the communication between heterogeneous devices and applications.

Since oneM2M is an international standard for IoT, its security implies the security of the IoT ecosystem. Hence, we focus our work on the security aspect of the oneM2M standard. In order to protect oneM2M-based IoT systems, we need to understand and examine its architecture. Moreover, we have to identify and study the attacks/scenarios we need to protect against.

This chapter starts with an overview of the oneM2M standard (Section 3.1): its architecture and the security mechanisms defined in its specifications. Section 3.2 details the proposed taxonomy of the threats related to the oneM2M standard. Moreover, Section 3.3 describes and discusses the dataset created related to the proposed taxonomy that would be used for our IDS proposal presented in Chapters 4 and 5.

3.1 OneM2M Standard and Security

OneM2M [One19] is a global standard initiative designed to converge towards an horizontal common platform for the multi-industry M2M applications (Figure 3.1) such as e-Health, intelligent transportation, industrial automation, smart homes, etc. Today,

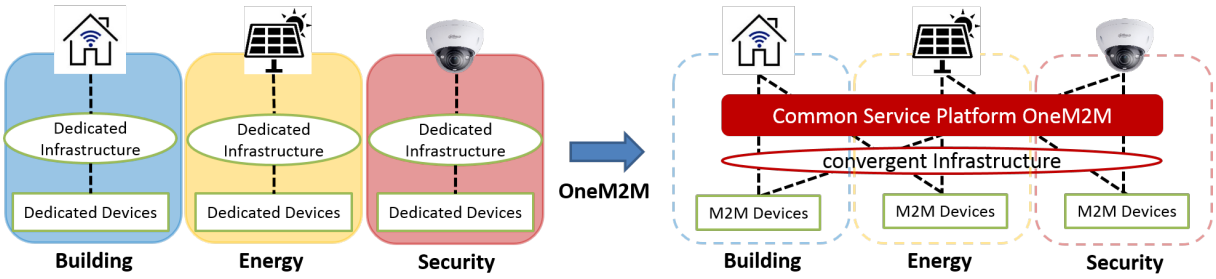

Figure 3.1: OneM2M commun service layer

many industries from different sectors rely on proprietary solutions with customized hardware and software for M2M systems hence IoT applications. Such vertical, mono-

industry solutions reinvent the wheel independently with non-interoperable technologies. Hence, eight of the word's leading ICT standards bodies [One19] initiated the international partnership project oneM2M in 2012. OneM2M main goal is to satisfy the need for a common M2M Service Layer which enables the communication of heterogeneous devices and applications with each other, regardless of their manufacturer or technical specifications with no need to redevelop common components. Therefore, deploying IoT and M2M solutions becomes less expensive in terms of money, time and complexity. It is important to mention that oneM2M takes into consideration the existing worldwide networks and standards as shown in Figure 3.2. It extends and standardizes the IoT ecosystem by interworking with other standards and protocols.

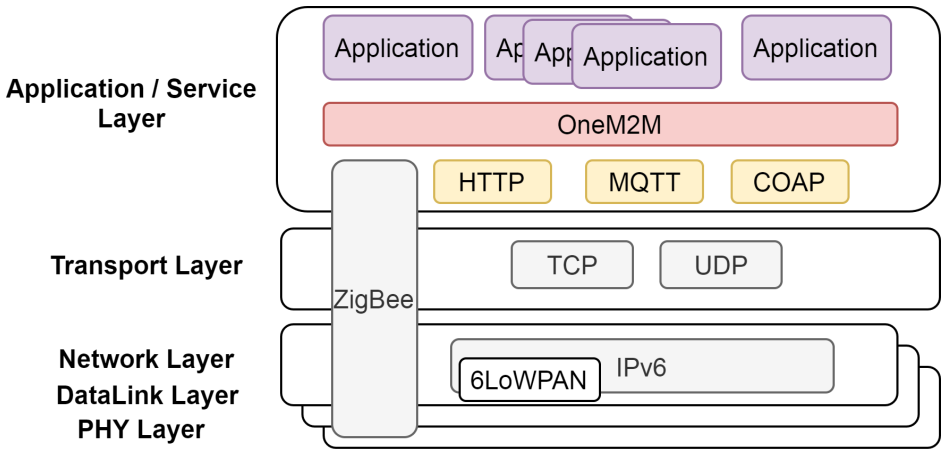

Figure 3.2: OneM2M service layer in the TCP/IP layer

OneM2M project defines a set of specifications for the standardization approach. It covers architecture details, security mechanisms, communication protocols, etc. In this section, we focus on the oneM2M architecture and its security.

3.1.1 OneM2M Architecture

The functional architecture of oneM2M is mainly composed of three layers as presented in Figure 3.3. First, the application layer provides functions related to the logic of the end-to-end M2M applications (e.g. remote blood sugar monitoring). Hence, an

application is represented by an Application Entity (AE). Furthermore, the common service layer exposes all the functions specific to the M2M environment such as data management, notification and subscription management, message handling, etc. The service layer relies on the Common Services Entity (CSE) as defined by the oneM2M team (with no dependence with the underlying networks). CSE takes a request as input (*RequestPrimitive*) and gives a response as output (*ResponsePrimitive*). Finally, the network layer relates the underlying network services (e.g. device management) to the layer of common services with the Network Services Entities (NSE).

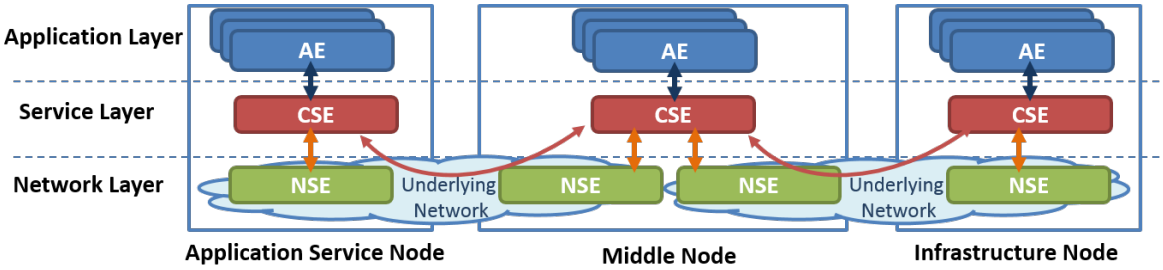

Figure 3.3: OneM2M architecture

A set of these layers forms a node which is the key component of M2M/IoT systems. Thus, a node is composed of a network layer, an application layer consisting of zero to several AE(s) and optionally a common services layer. However, a node without a CSE must be connected to another node that has one. Nodes only communicate together through the service layer or the application layer. Thus, there are several types of nodes: i) Infrastructure Node (IN) is the main node in a oneM2M domain and is unique in a multi-node architecture. Therefore, it contains all three functional layers and is characterized by additional features, such as its capacity to manage identifiers for all other nodes that are linked to it; ii) Middle Node (MN) represents a transition node and has all three layers; iii) Application Service Node (ASN) is like the MN except that no other node can connect to it (but it can connect to other nodes); and iv) Application Dedicated Node (ADN) looks like ASN but without the service layer. An M2M/IoT system is composed of a unique IN (as a main server), one or more MN (as gateways), many ASN (as devices) and many ADN (as constrained/small devices). Figure 3.3 schematizes the

main node types.

As detailed earlier, oneM2M is mainly about the service layer. Hence, at this point, we concentrate on the CSE. It allows to manage the resources of the node through oneM2M requests that follow the CRUD+N model (C for CREATE, R for RETRIEVE, U for UPDATE, D for DELETE and N for NOTIFY).

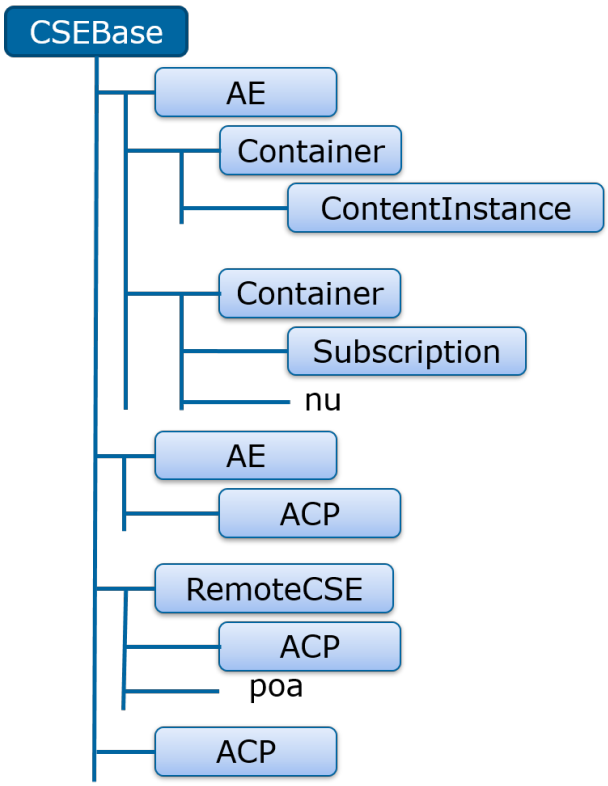

Figure 3.4: OneM2M resource tree

There are several types of resources in oneM2M [One18a] that are disposed in a resource tree model as illustrated in Figure 3.4. OneM2M logic is based on a resource data model. Hence, each service is represented as a uniquely identified resource (data structure). A resource has a set of attributes (e.g. resourceName, content, etc.) and a set of child resources (represented with rectangles in Figure 3.4). Below, we detail some of the characteristics of the resources we will need in the rest of the thesis:

- The CSEBase shall represent a CSE. It represents the root for all resources that are

residing in the CSE. A CSEBase can have a list of points of access (poa). A poa is used by the M2M system to communicate with a CSE on a M2M node. Typically, a poa contains information related to the network address.

- The AE refers to an application registered under the corresponding CSE. It has a poa attribute and can only have a CSEBase as a parent.
- The Container controls the data of an application. It is used to share information with other entities. Possible parents are AE and other Containers.
- The ContentInstance is a data instance that contains the useful data of a Container resource. The content of the ContentInstance can be encrypted. Unlike other resources, this resource shall not be modified once created. It can only be created, retrieved or deleted. Its only parent is a Container.
- The Subscription resource concerns subscription information about the oneM2M resource to which it is subscribed. NotificationURI (nu) refers to the list of one or more targets that the hosting CSE shall send notifications to when the corresponding Container has new data.
- The RemoteCSE is a representation of the CSE of a remote node. It can only have a CSEBase as a parent.
- The ACP refers to accessControlPolicy. It represents a set of access control rules [One16] defining which entities have the privilege to perform certain operations within specified contexts and are used by the CSEs in making access decision to specific resources. An ACP can be defined mainly for a CSEBase resource, an AE or a RemoteCSE. For resources like Container, ContentInstance and Subscription, they inherit the ACP of their parent/ancestor resource.

After having detailed the oneM2M architecture structure, we summarize the available security mechanisms proposed in the specifications of the project.

3.1.2 OneM2M Security

Among the specification documents of the oneM2M standard, the oneM2M community focuses on security and privacy aspects in TS-0003 [One18b]. OneM2M contributions in terms of information security seem to be an interesting step [TKG+17] to provide a sustainable development for both M2M and IoT applications while facing the challenges of M2M security and privacy.

The oneM2M security strategy is based on six main categories: a) identification and authentication, b) authorization, c) identity management, d) security association, e) sensitive data handling and f) security administration. To start with, identification (a) is the process of verifying the validity of the identity that asks to authenticate (e.g. if an AE (Application Entity) or CSE (Common Services Entity) fits a certificate). Then comes the authentication step where the validated identity needs to be associated to a trustworthy credentials (for example in the case of certificate authentication, it is a digital signature that needs to be checked). In oneM2M, it is possible to use a centralized key distribution server hosted by a 3rd party or by M2M Service Provider with a symmetric key access. In the second place, authorization function (b) regulates services and data access authorizations for already authenticated entities. This security mechanism is based on ACPs (Section 3.1.1) which are sets of conditions that reflect permitted accesses and role based access control which can be a token based framework (e.g. OAuth). Moreover, identity management (c) guarantees the anonymity of the entities in the oneM2M systems. In fact, oneM2M provides pseudonyms that play the role of temporary identifiers to protect the true identity of the nodes which is considered as sensitive data in the oneM2M architecture [One18a]. This mechanism is used independently of authentication and authorization functions. Furthermore, security association (d) is about equipping communicating entities with security services such as confidentiality and integrity of the exchanged information thanks to keys provided during the identification and authentication phase (a). Hence, the exchanged contents of resources are encrypted. As an example, at the service layer, exchanged messages be-

tween adjacent AE/CSE can be protected with a TLS or DTLS session. Last but not least, the sensitive data handling service (e) is mainly for the security of the application layer. It provides sensitive functions which enable secure storage, cryptographic operations and bootstrapping methods for initial secrets (e.g. GBA). Consequently, an isolated secure environment is guaranteed for data storing and retrieving such as credentials, subscriptions, personal information and for functions including security algorithms. Finally security administration (f) is introduced to manage all of the sensitive resources (data and functions) as well as to configure and extend the security services themselves.

Indeed oneM2M provides a large panel of security mechanisms to protect the service layer itself as well as the communication between the oneM2M architecture layers. However, none of the specified techniques protect the IoT systems once the malicious user has gained access to the system and bypassed the first-line security measures. Therefore, a second security line is required to detect and prevent intrusions from affecting the IoT systems. For this purpose, we have first to identify the attack scenarios we need to protect against.

3.2 OneM2M Threats

In order to protect the oneM2M standard, we decided to concentrate on threats related to the service availability in oneM2M which is, first and foremost, about services for M2M and IoT systems. Hence, we need to distinguish the related security threats. To do that, we specify threats by analogy with DoS taxonomy [DM04, SU14] since this type of attack is the one that corresponds to the availability of services in the network layer. We propose a taxonomy for service threats of the oneM2M standard. These attacks are based on legitimate behaviors that were exploited in malicious strategies. An attacker or a defective device could overwhelm, consciously or unconsciously, the resources and/or the network until bringing down the system. Hence, we assume in the attack descriptions that the malicious user has previously gained access to the system.

3.2.1 Proposed Taxonomy for OneM2M Threats

OneM2M threats could be classified into four types as shown in Figure 3.5.

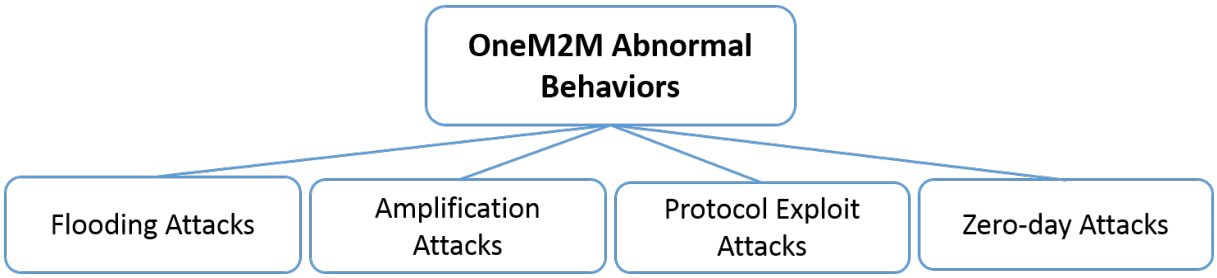

Figure 3.5: OneM2M threats taxonomy

- Flooding attacks are typically explicit attempts to disrupt legitimate users' access to services. It leads to services unavailability hence it costs time and data loss as well as money to mitigate the attacks and restore the services. In critical IoT systems, such as emergency fleet management (ambulance, police, etc.) or smart traffic signal systems, failures or delays in information exchange may cause serious problems. In oneM2M, we describe flooding attacks (Figure 3.6) as the submerge of the service layer with the legitimate oneM2M CRUDN operations. It is about bombarding a node with one or multiple types of operations. For example, a person with malicious intent can manipulate the different devices of an IoT network to create or retrieve a huge number of AE or Container resources in a target node thus, a target will dedicate all its resources to respond to the fake operations instead of real legitimate needs.

- Amplification attacks have the same goals as the flooding attacks. However, they differ in terms of strategy. In flooding attacks we specify direct actions that will occur during the attack, although in the amplification attacks, we put in place simple legitimate actions that will be later amplified to generate massive service operations. This could be based on amplification or reflection tactics. Let's take the example of "announceTo" mechanism in the oneM2M standard. An announced resource (let's say AE1, we consider AE1 as a child resource of CSE1) [One18a]

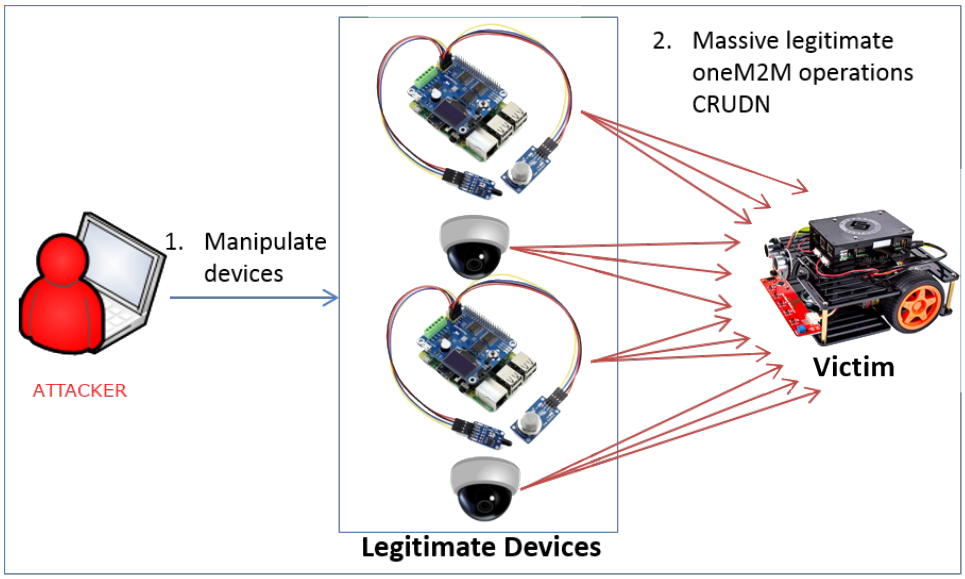

Figure 3.6: OneM2M flooding attack

is a representation (that we name (AEAnnc1)) of the resource (AE1) at a remote CSE (that we name CSE2) that is linked to the original resource (CSE1). AEAnnc1 maintains some of the characteristics of AE1. Hence, changes in these AE1 characteristics will be transfered / notified to all the remote presentations of AE1 which means to all the elements of the list in the "announceTo" (in our example to AEAnnc1). Consequently, as represented in Figure 3.7, if we put a large list in the "announceTo" and each element of the list will announce, in cascade, to another list, this will amplify the traffic when changes will be made on the first resource AE1.

- Protocol Exploit Attacks have a different strategy to consume resources. They exploit specific features or implementation bugs of the oneM2M protocol operations to overwhelm and/or bring down the device. An example of protocol exploit is the creation of loopholes/deadlocks in the notification system. It occurs when two resources register to the changes of each other. Hence, if a resource A has changes, a notification will be sent to resource B that will make changes as a result to the notification received from A. Since B has been changed, a notification will be sent

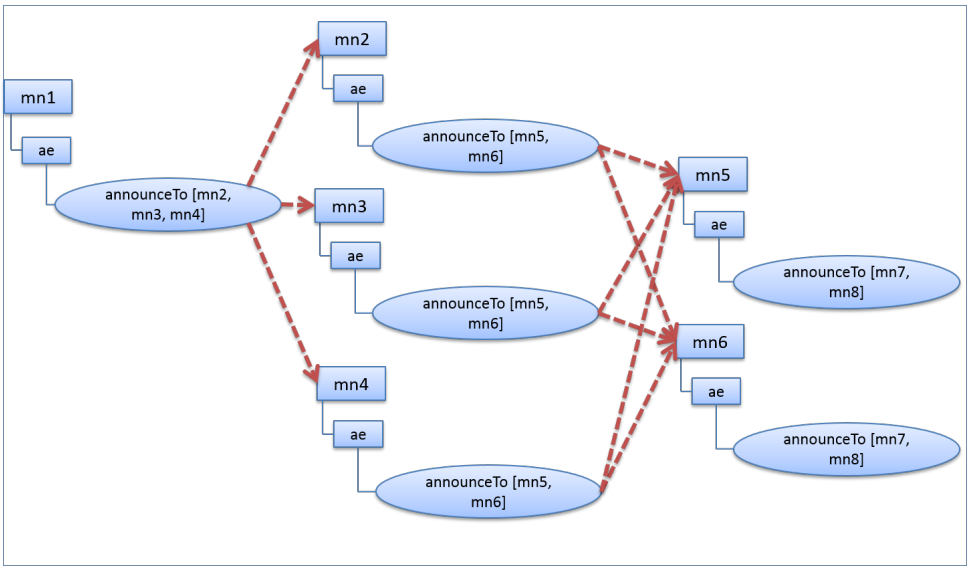

Figure 3.7: OneM2M amplification attack

to A to make new changes as well. Such a loop will consume resources in vain.

- Zero-day Attacks correspond to unknown or new abnormal behaviors which are not previously seen or at least analyzed. This term is widely used in the security community since it refers to unpredictable threats.

3.2.2 Attacks Implementation

In this part, we proposed examples of the implementation of each oneM2M attack category presented previously. These examples were set up to create our oneM2M security dataset, as well as to implement and test our IDS oneM2M solution proposal later. Regarding flooding attacks and amplification attacks, two parameters have been changed over the executions to have different instances in each type: N_OP which corresponds to the number of actions and N_TH which refers to the number of threads running the attack. Each type of attacks was running in centralized and distributed environments. Some of the attacks cause slowdown in the response to legitimate requests, others bring the target down.

3.2.2.1 Flooding Attacks

For this category of attack, we developed six types. The notation *N_OP* α *N_TH* stands for the expression "N_OP times with N_TH threads".

- AE Flooding (AF): In this attack we retrieve an AE resource *N_OP* α *N_TH*.
- Containers Flooding (CsF): In this attack we retrieve all the Container resources of a given AE *N_OP* α *N_TH*.
- Container Flooding (CF): In this attack we retrieve one Container resource of a given AE *N_OP* α *N_TH*.
- ContentInstance Flooding (CIF): We retrieve one ContentInstance resource of a given Container of a given AE *N_OP* α *N_TH*.
- Subscription Flooding (SF): In this attack we retrieve one Subscription resource of a given Container of a given AE *N_OP* α *N_TH*.
- Various Flooding (VF): In this attack we retrieve various resources from a given CSEBase *N_OP* α *N_TH*.

3.2.2.2 Amplification Attacks

For this category of attack, we need each time at least two AE resources under the same CSEBase or in two different related CSE nodes (e.g. A and B). We developed three types:

- Amplify One AE One Container (AOAOC): In this type, B will subscribe n times to the same Container of A with N_TH threads. These subscriptions are possible since each Subscription has a different identifier. After that, we create N_OP ContentInstance under the corresponding Container of A. Consequently, each new creation of a ContentInstance will be notified to B. Since we have n subscriptions for the same Container then, for each new ContentInstance creation, we have n * N_TH generated notifications as shown in Figure 3.8.

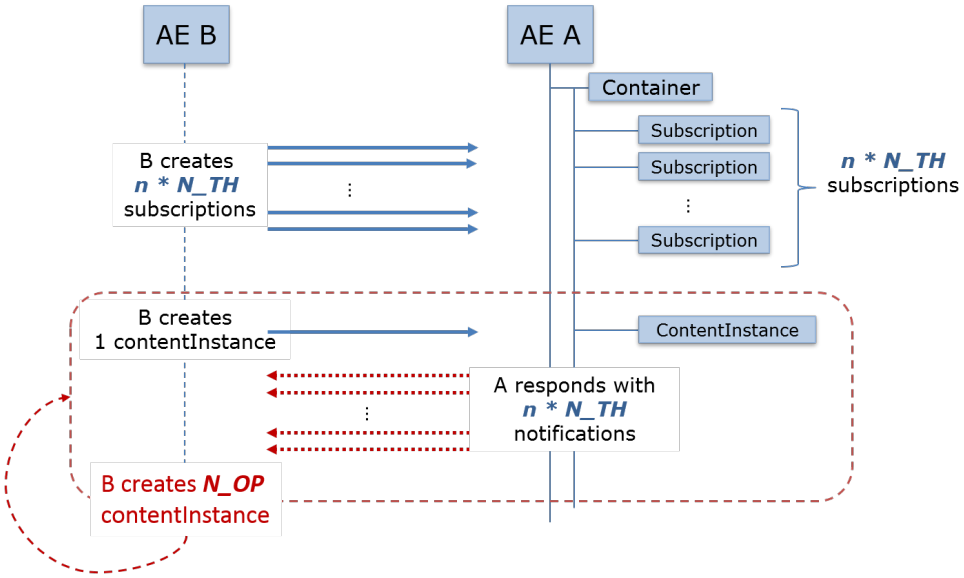

Figure 3.8: AOAOC attack

- Amplify One AE Multiple Containers (AOAMC): This type is similar to AOAOC but concerns not only one Container but N_OP Containers.

- Amplify Discovery AE (ADA): This attack is also based on the same principle as AOAOC, however it concerns all the contained AE under the same CSEBase resource as A. Thus B subscribes to all the Containers of all the AE resources at the same level as A.

3.2.2.3 Protocol Exploit Attacks

For this category, we implemented a loophole attack that is based on the *poa* attribute. We have an AE resource A with a *poa* value for example "http://foo:8181". A is registered under a CSEBase with the same value of *poa* as A ("http://foo:8181"). We have also another AE resource named B (under the same CSEBase as A or in a remote CSE nodes). A will subscribe to a Container of B. Hence, for each new value under this Container, A will be notified thanks to the declared *poa* address. Since both the CSEBase and the AE node A have the same *poa*, the notification will be received by the AE A that will redirect it to the CSEBase and hence we create a loop (Figure 3.9).

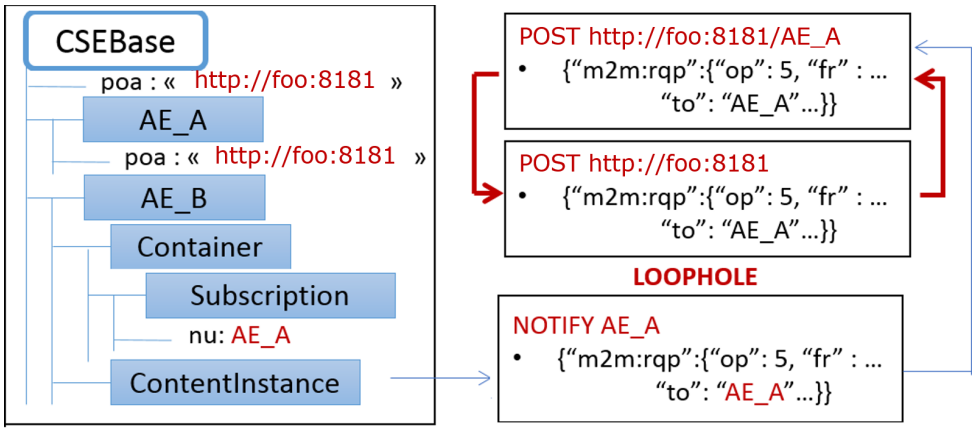

Figure 3.9: OneM2M loophole attack

3.3 OneM2M Dataset

After having analyzed the different threat scenarios against oneM2M standard, we present in this section the state of the art of free datasets used for the IoT NIDS creation. In addition, as no datasets exist for the oneM2M standard, we detail our own dataset creation.

3.3.1 State of the Art of Free Datasets

Free datasets can be used for NIDS implementation and/or validation. Unfortunately, there are no datasets created specifically for IoT networks. Hence, two strategies are possible: download an available dataset targeting traditional systems or deploy sniffing software in networks.

The most widely adopted datasets for NIDS are KDD99, and NSL-KDD which is an improved version of KDD99. Public datasets like PREDICT [pre], CAIDA [Ana], DEFCON [def], ADFA IDS [noaa], KYOTO [STO+11] and ISCX 2012 [isc] attack datasets are available for evaluation and testing. The latest are either composed of unlabeled data, or are inaccessible from some countries or are specific domain data. Moreover, datasets suffer from i) privacy issues; ii) the heavy inputs anonymization; and iii) the non reflection of current security attacks.

- KDD99 [HB99] is a dataset used for detection of "bad" connections from the "good" ones at the Third International Knowledge Discovery and Data Mining Tools Competition [XL05] for building the robust NIDS. The dataset is the feature extracted version of DARPA dataset [dar] (DARPA is a base raw dataset). KDD99 contains records from military network environment with injected attacks which can be categorized into: i) Denial of Service; ii) Remote to User; iii) User to Root; and iv) Probing. KDD99 is based on 41 features for each connection along with the class label using Bro-IDS tool (presented lately). The features are grouped into 4 types [HB99]:

  - 1-9: Basic features of individual TCP connections.
  - 10-22: Content features within a connection suggested by domain knowledge.
  - 23-31: Traffic features computed using a two-second time window.
  - 32-42: Host features are designed to assess attacks which last for more than two seconds.

  KDD99 is popular and is the most used by the researchers for experimental analysis. Different works [CN12, KH05, SSJ14, AOF+10, NFP10, AAD10] were established to reduce the number of features by selecting the most relevant ones from the initial 41 features. However, many researches have reported disadvantages of KDD99 like [GBBK12], [VHS11]. Some of the important ones are [ANMH16, MS15, MC03]:

  - The probability distribution of the testing and training sets are different, because of adding new attack records in the testing set [MS15]. In other words, KDD99 suffers from unbalanced classification methods. Balance between the types of attacks and normal traffic is not maintained anymore.
  - The dataset is out of date (1999).

  - There is evidence of simulation artifacts that could result in over-estimations of anomaly detection performances.

- NSL-KDD [MS15, noa16] is the upgraded version of KDD99 to overcome its limitations. First, duplicated records in the training and test sets are removed. Second, there are a variety of records selected from the original KDD99 to achieve reliable results from classifier systems. Third, the problem of unbalanced probability distribution is eliminated. The major problem that persists in this dataset is the lack of modern low foot print attack scenarios.

- UNSW-NB15 [MS15] was created in 2015 by the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) with IXIA PerfectStorm tool. Its goal is to generate hybrid real modern normal activities and synthetic contemporary attack behaviors. It is about two million and 540,044 records which are stored in four csv files. Those records are generated from 100GB captured raw traffic with tcpdump tool [tcp17] (in pcap files). This dataset has nine types of attacks, namely, Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms. Fig. 3.10 illustrates steps to generate UNSW-NB15 dataset.

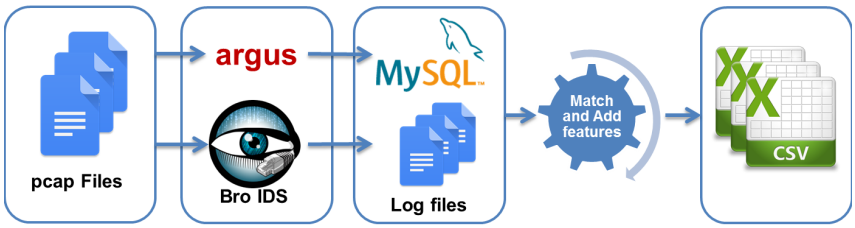

Figure 3.10: How to generate UNSW-NB15 dataset

- Sivanathan et al. IoT dataset [SSG+17, SHHS] addresses IoT device classification based on network traffic characteristics. Authors instrument a smart environment for 28 IoT devices like spanning cameras, lights, plugs, motion sensors, appliances and health-monitors. Furthermore, they synthesized network traffic traces from

their infrastructure for a period of six months released for research community. Sivanathan et al. present valuable insights about the network traffic patterns via statistical analysis using attributes such as activity cycles, port numbers, signaling patterns and cipher suites.

- CICIDS database [fCC17] is one of the recent Intrusion Detection/Intrusion prevention database released by Canadian Institute for Cyber-security, University of New Brunswick to reflect latest threats resembling the real-world data. It was built on the abstract behavior of 25 users based on the HTTP, HTTPS, FTP, SSH, and email protocols. The dataset is analyzed with CICFlowMeter [HLDGMG17] with labeled flows based on timestamp, initial and final IP, ports, protocols and attacks. To generate the realistic traffic, authors proposed B-Profile [SHLG18] approach to outline the behavior on HTTP, HTTPS, FTP, SSH and e-mail protocols. Authors implemented Brute force FTP, SSH Heartbleed and DDos attacks while capturing the data. The evaluation framework [GSLG16] identified eleven important features necessary to build a reliable benchmark dataset, unlike the existing traditional IDS datasets.

- CSE-CIC-IDS2018 [noa18] database is a unique IDS dataset which has evolved to replace the existing suboptimal datasets that limits IDS/NIDS experimental evaluations. To overcome the use of static and one-time datasets, CSE-CIC-IDS2018 is an anomaly based dynamically generated dataset consisting intrusion in network traffic. Authors included seven attack scenarios including i) Brute-force; ii) Heartbleed; iii) Botnet; iv) DoS; v) DDoS; vi) Web attacks; and vii) Local network infiltration attacks. Attack infrastructure has 50 nodes and victim organization has 5 departments with 30 servers and 420 hosts. Authors extracted 80 features from network traffic and machine logs captured via CICFlowMeter-V3.

In the following, the presented free network datasets are discussed. As shown in the comparison Table 3.1, KDD99 is the most popular network dataset. It has been used

Datasets	Advantages	Drawbacks
KDD99 [HB99]	• KDD99 is popular and the most used. • Labeled data. • It is based on 41 features for each connection along with the class label. • Implements Denial of Service, Remote to User, User to Root and Probing attacks. • Provides network traffic (PCAP).	• KDD99 suffers from unbalanced classification methods. • The dataset is out of date. • Not for IoT systems.
NSL-KDD [noa16]	• It is a better version of KDD99. • It overcomes KDD99 limitations. • No duplicated records in the training and test sets.	• Lack of modern low foot print attack scenarios. • Not for IoT systems.
UNSW-NB15 [MS15]	• It provides hybrid real modern normal activities and synthetics contemporary attack behaviors. • Provides network traffic (PCAP) and CSV files. • It has nine types of attacks, namely, Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms.	• It is more complex than the KDD99 dataset due to the similar behaviors of the modern attack and normal network traffic.
Sivanathan et al. Dataset [SSG+17]	• Network traffic IoT dataset. • It reflects real world IoT systems. • Provides network traffic (PCAP) and CSV files.	• Unlabeled data. • For IoT devices proliferation and traffic characterizing. • No attack data.
CICIDS [fCC17]	• Labeled network flows. • For machine and deep learning purpose. • Provides network traffic (PCAP) and CSV files. • Implements attacks such as Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet and DDoS.	• Not public. • Not for IoT systems.
CSE-CIC-IDS2018 [noa18]	• Labeled network flows. • For machine and deep learning purpose. • Provides network traffic (PCAP), CSV and log files. • Implements Brute-force, Heartbleed, Botnet, DoS, DDoS, Web attacks and Local network infiltration attacks. • Dynamically generated dataset. • It is modifiable, extensible, and reproducible.	• Not public. • Not for IoT systems.

Table 3.1: Comparison between free datasets

since 1999. Unfortunately, it is out of date. To overcome KDD99 limitations, NSL-KDD was created. It has balanced data with no duplicate records. Since NSL-KDD lacks modern attacks, UNSW-NB15 was proposed. It is a well reputed dataset with recent attacks. Meanwhile, it is more complex than KDD99 in terms of similarity between

the new attacks and the normal behaviors. As more recent network datasets, there are i) Sivanathan et al. dataset; ii) CICIDS and iii) CSE-CIC-IDS2018. Sivanathan et al. work is the only IoT network traffic dataset compared to the other presented ones. However, it is designed for IoT devices proliferation and not for intrusion detection. CICIDS and CSE-CIC-IDS2018 have labeled records but are not targeting IoT systems security despite their up-to-date attack list.

3.3.2 OneM2M Dataset Creation

As discussed previously, no dataset exists for the oneM2M standard, hence we decided to create our own dataset with respect to the taxonomy of oneM2M threats presented in Section 3.2.

3.3.2.1 OneM2M Dataset Features: *GFlows* Abstraction

We start by analyzing the communication model of the oneM2M service layer to decide about the features that will be stored for our dataset. OneM2M information exchange is

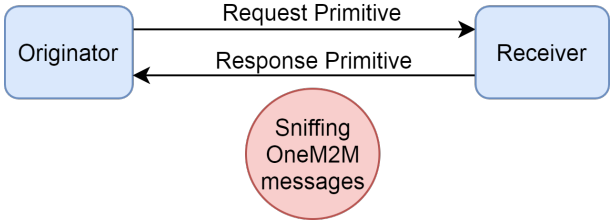

Figure 3.11: OneM2M flow

based on a pair of Request and Response messages referenced as a flow (Figure 3.11) in the oneM2M specifications [One18a]. Requests from an originator to a receiver contain mandatory and optional parameters depending on the requested operation and the involved oneM2M tree resources. As we need to create a dataset that could be used by all implementations of the oneM2M specifications, we need fixed characteristics to store. Therefore, our dataset is built only on the basis of the following mandatory parameters of the oneM2M messages:

- *requestIdentifier (rqi)* is a string key that enables the correlation between a request and its corresponding response.

- *From (fr)* is a string parameter that identifies the originator of the request. It is needed for the receiver to verify the originator identity in terms of access privilege.

- *To (to)* refers to the identity of the receiver.

- *Operation (op)* integer parameter reflects the operation to be executed at the receiver: CREATE, RETRIEVE, UPDATE, DELETE and NOTIFY.

- *responseStatusCode* indicates the result status of the requested operation if it has been successfully or unsuccessfully processed. For example 2000 corresponds to *OK* status, 2001 refers to *CREATED*, 2002 is *DELETED*, etc. There is a large amount of values of *responseStatusCode* that are specified in the oneM2M specifications [One16].

In analogy with the network IDS [CMZ+19], we notice that considering only basic features (only the mandatory attributes of each request / response) will not give a global and detailed information on the threat in progress. Consequently, we introduce a new abstraction of the oneM2M standard flows that we named *GFlows* for *Generated Flows*. We built these *GFlows* on the messaging mechanism. *GFlows* will be the inputs to our IDS. A *GFlow* encompasses multiple oneM2M original flows on the basis of the key {from, to, op and responseStatusCode}. Such aggregation of flows makes the analyses (presented in the rest of this thesis) lighter, which is important in the context of IoT. Thus, for each n exchanged flows, a set of *GFlows* is generated. Besides the *GFlow* key attributes, we generate the properties detailed in Table 3.2. To propose this flow abstraction, we have tried to cover as many combinations and properties as we think relevant to allow maximum threat detection. The final dataset considers all the *GFlow* properties except for *From* and *To* to respect anonymity. So in total we ended up with 26 features: *op* and *responseStatusCode* from the mandatory parameters and the 24

Property Name	Type	Description
counterKey	Integer	The number of request / response sharing the same key in NB flows
isSameFromTo	Boolean	To check if (From) and (to) have the same values in the *GFlows* key
isFromRemote	Boolean	To check if the request is from a remote CSE
isToRemote	Boolean	To check if the request is to a remote CSE
fromResourceType	Integer	The type of (from) resource: (1-AE), (2-Container), (3-ContentInstance), etc.
toResourceType	Integer	The type of (to) resource: (1-AE), (2-Container), (3-ContentInstance), etc.
counterSameFromRequests	Integer	The number of *GFlows* sharing the same (from) resource in NB flows
counterSameToResponses	Integer	The number of *GFlows* sharing the same (to) resource in NB flows
counterSameTypeResponses	Integer	The number of *GFlows* having the same responseStatusCode type in NB flows
counterSameCategoryResponses	Integer	The number of *GFlows* having the same responseStatusCode category in NB flows
duration	Long	The duration of the registered *GFlows*
counterFlows	Integer	The ranking of the *GFlows* in NB flows
counterSameOperations	Integer	The number of *GFlows* sharing the same (op) attribute in NB flows
counterSameFromTo	Integer	The number of *GFlows* sharing the same (from-to) attribute in NB flows
counterSameFromOp	Integer	The number of *GFlows* sharing the same (from-op) attribute in NB flows
counterSameFromResponseType	Integer	The number of *GFlows* sharing the same (from-responseType) attribute in NB flows
counterSameFromResponseCategory	Integer	The number of *GFlows* sharing the same (from-responseCategory) attribute in NB flows
counterSameFromOperationResponseType	Integer	The number of *GFlows* sharing the same (from-op-responseType) attribute in NB flows
counterSameFromOperationResponseCategory	Integer	The number of *GFlows* sharing the same (from-op-responseCategory) attribute in NB flows
counterSameToOperation	Integer	The number of *GFlows* sharing the same (to-op) attribute in NB flows
counterSameToResponseType	Integer	The number of *GFlows* sharing the same (to-responseType) attribute in NB flows
counterSameToResponseCategory	Integer	The number of *GFlows* sharing the same (to-responseCategory) attribute in NB flows
counterOperationResponseType	Integer	The number of *GFlows* sharing the same (op-responseType) attribute in NB flows
counterSameOperationResponseCategory	Integer	The number of *GFlows* sharing the same (op-responseCategory) attribute in NB flows

Table 3.2: OneM2M *GFlows* properties

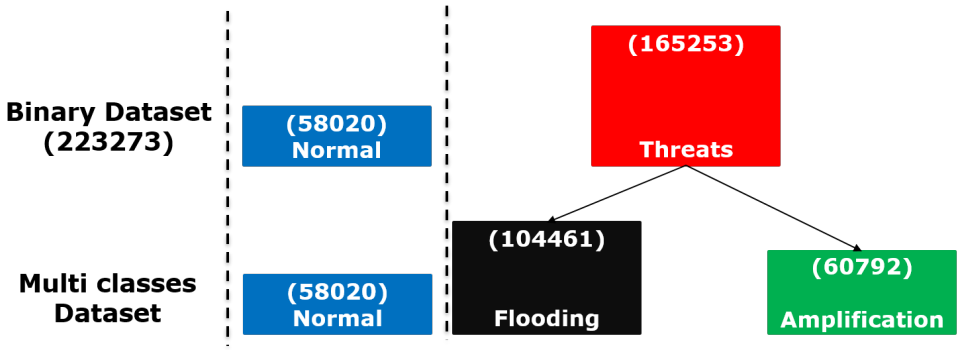

Figure 3.12: OneM2M dataset

properties detailed in Table 3.2 plus the label which could be either "*NORMAL*" or the exact name of the threat.

3.3.2.2 OneM2M Dataset Generation

In the rest of the thesis, we will concentrate mainly on oneM2M flooding and amplification attacks (previously detailed). To generate the threats dataset that will be used later in the IDS construction, we apply the different types of flooding and amplification attacks on a Raspberry Pi 3 Model B (Quad Core 1.2GHz Broadcom BCM2837 64bit CPU and 1GB RAM). We equipped the Raspberry Pi with a oneM2M instance to play the role of an ASN. The used oneM2M implementation is the *Codex Data Platform IoT* initiated in May 2017 by Atos Innovation Aquitaine Lab[1]. By running multiple attack instances (launched from one or many computers/devices), we generate a dataset composed of 223 273 of *GFlow* inputs: 165 253 lines of threats and 58 020 of benign flows (Figure 3.12). We summarize the number of *GFlow* generated for each type of attack: 22 258 of AF, 18 867 of CsF, 13 489 of CF, 9 569 of CIF, 14 922 of SF, 25 356 of VF, 24 449 of AOAOC, 17 865 of AOAMC and 18 478 of ADA. A *GFlow* is tagged as an attack if it causes more than 6 seconds of delay in the oneM2M response acquisition or if it causes errors / shutdown of the oneM2M platform. Otherwise, it is labeled as normal.

[1]https://atos.net/

Conclusion

IoT is increasingly widespread. Hence, industrials tend to put in place proprietary solutions. OneM2M is an international standard created to ensure horizontal IoT cross-industry interoperability. In this chapter, we start by presenting the oneM2M standard architecture as well as the security mechanisms already introduced in the oneM2M specifications. Moreover, we propose a threat taxonomy of the oneM2M standard and then we define and implement the threats scenarios. We conclude this chapter by presenting the state of the art of free datasets used for the creation of IDS and we detail our dataset creation. To create our oneM2M dataset, we define an abstraction of the oneM2M exchanged flows.

In the next chapter, we propose an intrusion detection and prevention system (IDPS) to secure the IoT systems based on the oneM2M standard.

Chapter 4

An Intrusion Detection and Prevention System for the Service Layer

Contents

Introduction

Due to the limitation of the security techniques presented in Section 3.1, in terms of detection and prevention of the IoT systems threats, we decided to propose an IDPS for the oneM2M service layer based on machine learning (oneM2M-IDPS). An IDPS [CMZ+19] is a system which detects and responds immediately to potential threats as soon as they are carried out. It is mainly composed of two phases: i) the detection phase

and ii) the response phase. We present in this chapter the different challenges respected and guaranteed with our IDPS proposal, then we point out its main goals. Moreover, we detail its strategy as well as the architecture and the design of each component / module of the implemented oneM2M-IDPS.

Before diving into this chapter, it is important to insist on our definition of the term "threat" as being either a security attack or abnormal/anomalous behavior that could lead to the bug or shutdown of the IoT system. Thus, for the rest of the dissertation, the words "threat", "attack", "abnormal behavior" and "anomaly" are used interchangeably to express any behavior that could affect the IoT system.

4.1 OneM2M-IDPS Challenges and Aims

Since we are in the context of security and IoT which are challenging domains as presented in Section 1.1.1.2.2, many characteristics need to be taken into consideration in order to build an efficient, effective and resilient IDPS system. We establish our proposal on the following features and goals:

- Interoperability: One of the strengths of our proposal is its interoperability with the other platforms implementing the oneM2M standard. In other words, any new device that wishes to join an existing IoT system or even any new platform using oneM2M that wishes to communicate with the already secured platform can easily be secured with our oneM2M-IDPS.

- Autonomy, scalability and resource constraints respect: We combine the fog and the cloud computing paradigms in our proposal (Section 4.2.2). Fog nodes are autonomous, scalable and have fast reaction in terms of detection and prevention. In other words, each device protects itself independently from the rest of the system. Moreover, the IDPS respects resource constraints and therefore deals with resource-consuming steps in the cloud. Consequently , the oneM2M-IDPS intelligence is distributed between the fog and the cloud. The fog is for autonomous

detection and prevention and the cloud is for retraining and continuous learning.

- Modularity in design: We decide to subdivide our IDPS into separate and independent modules in order to ensure better organization, separate functionalities and easier maintenance (modifications, evolutions, replacement, etc.).

- Adaptability and extensibility: It is about having an adaptable and easily extensible intrusion detection and prevention. The oneM2M-IDPS detection and prevention capabilities should not be frozen on known threats. First, the IoT threats are whether security attacks or abnormal behaviors that could lead to the IoT system bug or shutdown. Second, the solution should not only protect the system from already known oneM2M threats, but also be able to learn unknown threats over time. In other words, the system needs to be able to self-learn and adapt itself to new threats and new configurations of the IoT system. The IDPS is destined to supervise the environment and adjust itself according to every change.

- Active and real-time reaction: The prevention strategy against the detected threats should be through immediate and appropriate actions. It needs to be an active, real time prevention to assure the oneM2M services availability.

4.2 OneM2M-IDPS Strategy

Our oneM2M-IDPS strategy detects and responds immediately to potential threats as soon as they are carried out. It has the ability of the threat controlling in real time. It is composed of four main modules as shown in Figure 4.1:

- the data acquisition and features extraction module where the oneM2M messages are prepared to be injected to the IDS module,

- the IDS module to detect the threats,

- the IPS to prevent the threats from damaging the system,

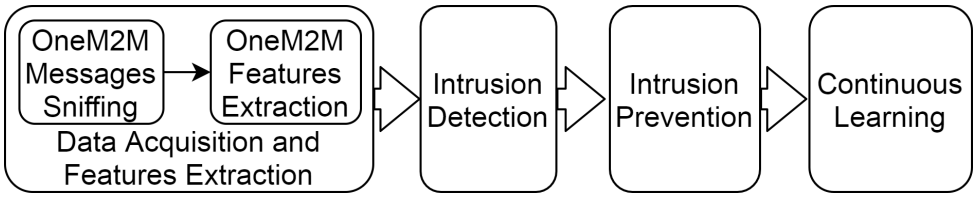

Figure 4.1: OneM2M IDPS strategy

- the continuous learning module to guarantee the update of the IDPS against the new threats.

We detail each module in the rest of this section.

4.2.1 Data Acquisition and Features Extraction

This module captures the oneM2M messages to be analyzed, processed and enriched. It sniffs the oneM2M flows composed of a pair of Request and Response messages as mentioned in Section 3.3. Then, features are extracted and processed to construct the *GFlows* that are fed to the intrusion detection module.

4.2.1.1 OneM2M Messages Sniffing

To sniff the oneM2M flows, there are two possibilities. The first uses existing network sniffing tools and the second is built on the oneM2M standard implementation platform.

4.2.1.1.1 Free, Open Source Network Sniffers A sniffing tool [HBB+14] aims to monitor the network transit traffic from source to destination. It can be used to capture, examine, analyze and visualize packets or frames. The idea behind such tool in the oneM2M-IDPS is to extract the oneM2M flows from the sniffed network packets. Since many sniffers exist in the state of the art, we choose to detail and compare the free, open source ones.

- *Tcpdump* [tcp17, HBB+14] is the most popular, powerful and widely used packet analyzer. It is a TCP/IP command-line tool which enables capturing, analyzing,

saving and viewing packet data. Van Jacobson, Craig Leres, and Steven McCanneIt develop *Tcpdump* at the Lawrence Berkeley Laboratory, UC, Berkeley. *Tcpdump* captures live packet data from a network interface. An interesting feature in *Tcpdump* is the possibility to save the captured packets in a pcap file for a further analysis. *Tcpdump* uses the *libpcap* library to capture packets. *Libpcap*, is frequently used by other capture programs. *Tcpdump* tool is available for most of the Linux/Unix based operating systems. The most popular open source GUI (Graphical User Interface) based on *Tcpdump* is *Wireshark* (third-party software) which reads *Tcpdump* pcap files enabling an easy-to-use, user friendly interface.

- *Wireshark* [noaf, HBB+14, AP12] is a popular, free and open source packet analyzer, under the GNU license. It is used for network sniffing and network analysis. It captures live packet data from a network interface. *Wireshark* runs on Unix-like operating systems, Solaris, and Microsoft Windows. It uses *libpcap* as a library to capture and filter packets; then displays records with its GUI. This tool enables reading *Tcpdump* outputs. *Wireshark* decodes a large panel of protocols (> 400). It supports preliminary inspection of attacks in the network. Its command line version is "*tshark*".

- *Ettercap* [noac, HBB+14] is a multi-platform network sniffer. It is "a multipurpose sniffer/interceptor/logger for switched LANs" [noac] written by Alberto Ornaghi and Marco Valleri. *Ettercap* is known for its powerful ability in launching several different types of man-in-the-middle attacks. In addition, it provides users with many separate classic attacks and reconnaissance techniques within its interface. It sniffs live connections and filters packets as well as many other features in both active or passive way.

- *Argus* [noa17a, HBB+14] is a tool to capture and analyze network flow data. It runs on several OS like like Linux and Windows. It focuses on developing network activity audit strategies. Moreover, *Argus* treats live and captured traffic data to

generate status reports / audits on detected flows with a semantic analysis. It processes *libpcap* and *Endaces's ERF* packet data to enable the user having an idea about what is going on a network. This tool provides information on almost all packet parameters like duration, rate, load, retransmission, delays, etc.

- *EtherApe* [noab, HBB+14] authored by Juan Toledo and Riccardo Ghetta in 2000 is a graphical packet sniffer and network monitoring tool. It supports only Unix platforms. *EtherApe* aims to represent packets, connections and data-flows visually with color coded hosts and links for the protocols. The tool also facilitates network troubleshooting. Furthermore, it supports real-time display of network packets via standard formats. Traffic may be consulted on one's own network, end-to-end (IP) or port-to-port (TCP).

In the following, the different free and open-source network sniffers are compared in Table 4.1. As it can be noticed, *Tcpdump* is the most popular network sniffer (all the other sniffers try to support its outputs). It is a long life product often updated and extended with multiple features. It is well documented and enjoys community support. However, *Tcpdump* is primarily developed for data capture unlike other tools equipped for network analysis. It is a command line tool with no real GUI. While the strength of *Wireshark* and *EtherApe* is in their graphical features, both can display real-time and captured network files. *Wireshark* is better known than *EtherApe*. Besides, *Wireshark* is a cross platform sniffer which is not the case for *EtherApe* supporting only Unix platform. Moreover, unlike *EtherApe*, *Wireshark* takes into account both header and payload details. About *Argus*, it is more a tool to audit network activities. It decodes several protocols for reports and audits. Regarding *Ettercap*, apart from network data in sniffer, interceptor and logger mode, it can manipulates the network and launch different MITM attacks. It is able to collect passwords, kill connections, inject packets and commands in active connections. Hence, it can be considered more of a hacker tool than a network sniffer.

As a first step in our work, we used *Wireshark* as a convenient, easy-to-use tool to get

a graphical overview. Then, we switched to *Tcpdump* since we do not need a graphical user interface for the final use.

Network sniffers	Advantages	Drawbacks
Tcpdump [FK05, AP12, HBB+14]	• Long product life with different updates and plenty of features. • Well documented with a good community support. • Easy remote access with Telnet connection. • Cross-platform (has even been ported to Windows too). • Less intrusive compared to *Wireshark*. • Captures live packet data from a network interface. • Saves captured packet data. • Lightweight in terms of installation.	• Lacks critical analysis. • Discards invalid packets (Not helpful for detecting broken packets) . • No real GUI or administrative console.
Wireshark [FK05, AP12, HBB+14]	• Well documented with a good community support. • Cross-platform. • Supports large number of protocols. • Graphical tool. • Captures live packet data from a network interface. • Saves and open packet data files. • Provides detailed protocol information.	• No abnormal behavior notifications (Not an IDS). • Gathers information but cannot manipulate the network. • Resource consuming in terms of installation.
Ettercap [noac, HBB+14]	• Cross-platform. • Can be used for LAN hacking techniques. • Decodes several protocols. • Collects passwords for multiple applications • Manipulates the network by killing connections, by injecting packets and commands into active connection(s). • Extensible with additional plug-ins.	• Sniffing is a secondary feature. • Can be used as a hacker tool. • Can be detected by other network tools (for example by *Ettercap* itself).
Argus [noa17a, HBB+14]	• Cross-platform. • Decodes several protocols. • Generates reports and audits about the network. • Native file system as well as MySQL support. • Efficient in large amount of network traffic analyzing.	• Not too obvious to master.
EtherApe [noab, HBB+14]	• Displays graphics for network activity with a color coded protocols mode. • Hosts and links change in size with traffic. • Can filter packets. • Supports multiple frames and packet types. • Supports file and real-time network traffic. • Good reputation among the system administrator community.	• Supports only Unix OS. • No command line version. • Captures only packet headers.

Table 4.1: Comparison between free, open-source network sniffers

4.2.1.1.2 OneM2M platform-based Sniffer The second solution for oneM2M messages sniffing is based on the standard implementation platform. For our case, we use the *Codex Data Platform IoT* initiated in May 2017 by Atos Innovation Aquitaine Lab. This implementation of the oneM2M standard is based on OSGi[1] which is a framework that allows to isolate each component of a Java application and to manage its lifecycle independently: installation, start-up, shutdown, update and uninstallation. Each component is called a bundle. To sniff the exchanged flows, we used the publish/subscribe communication paradigm [EFGK03]. It is a messaging pattern that connects the producers of an information/event called publishers with the consumers of this same information called subscribers. In other words, the producers publish messages asynchronously through a communication channel called broker (previously set up for that purpose) and the consumers receive messages by listening synchronously to that channel. Publishers do not program the messages to be sent directly to specific subscribers, but instead categorize published messages into classes without knowledge of which subscribers, if any, there may be. Similarly, subscribers express interest in one or more classes and only receive messages that are of interest, without knowledge of which publishers, if any, there are.

To put in place this paradigm, we used the Java Message Service (JMS) API [Ora] which is a Java message-oriented middleware API that handles the producer–consumer problem. So we started by preparing an ActiveMQTopic [Fou] as a message broker (a topic unlike a queue allows the use of many subscribers to receive a copy of the message). Then, we modified the core bundle of the platform to publish the primitive requests/responses in transit. And we implemented a bundle for data acquisition and features extraction as a consumer. The architecture of this component is detailed in Figure 4.2 It is true that this solution requires the modification of the oneM2M platform to integrate the sniffing process, but it is easy to implement for any oneM2M platform (any

[1]The OSGi technology [All] facilitates the componentization of software modules and applications and assures remote management and interoperability of applications and services over a broad variety of devices.

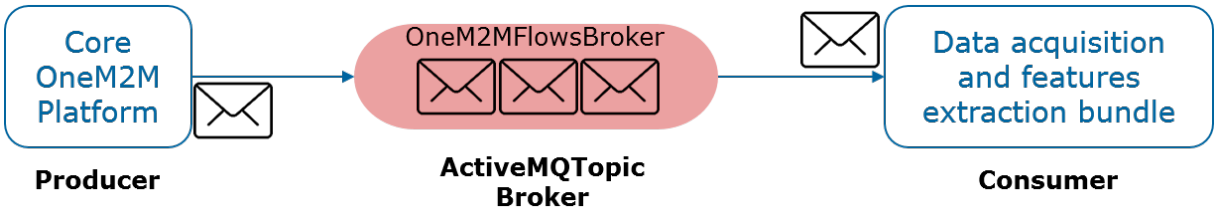

Figure 4.2: Architecture oneM2M platform-based sniffer

implementation of the oneM2M standard). The use of the publish/subscribe paradigm provides greater network scalability and a more dynamic network topology. Hence, we use this platform-based sniffer for the rest of the works and experiences.

4.2.1.2 OneM2M Features Extraction

The goal of this component is the extraction of the necessary information from the primitive requests/responses and the generation of the features composing the *GFlow*. As presented in Section 3.3.2.1, *GFlow* is a new abstraction for the oneM2M messages introduced in [CMZS19] based on the mandatory parameters of the primitive requests/responses. A *GFlow* encompasses multiple oneM2M flows on the basis of a key from, to, op and responseStatusCode. For each n exchanged flows, a set of *GFlows* is generated. n is a dynamic parameter that needs to be fixed by the user. A *GFlow* is composed of 2 properties from the key which are op and responseStatusCode and 24 generated features (such as flags and counters) as presented in Table 3.2. We do not base our final *GFlow* properties on the from and to for purpose of respect of the anonymity of the IoT devices' identities. Hence, even if a malicious person try to hack or sniff these *GFlows*, no identity is revealed.

As explained before, we implement a bundle that will subscribe to the exchanged primitive requests/responses, extract the key properties and generate the 26 features of the *GFlow*.

We summarize the workflow of the oneM2M features extraction module in Figure 4.3. For the rest of the chapter, we use *oneM2MFlowsBroker* as the name of our bro-

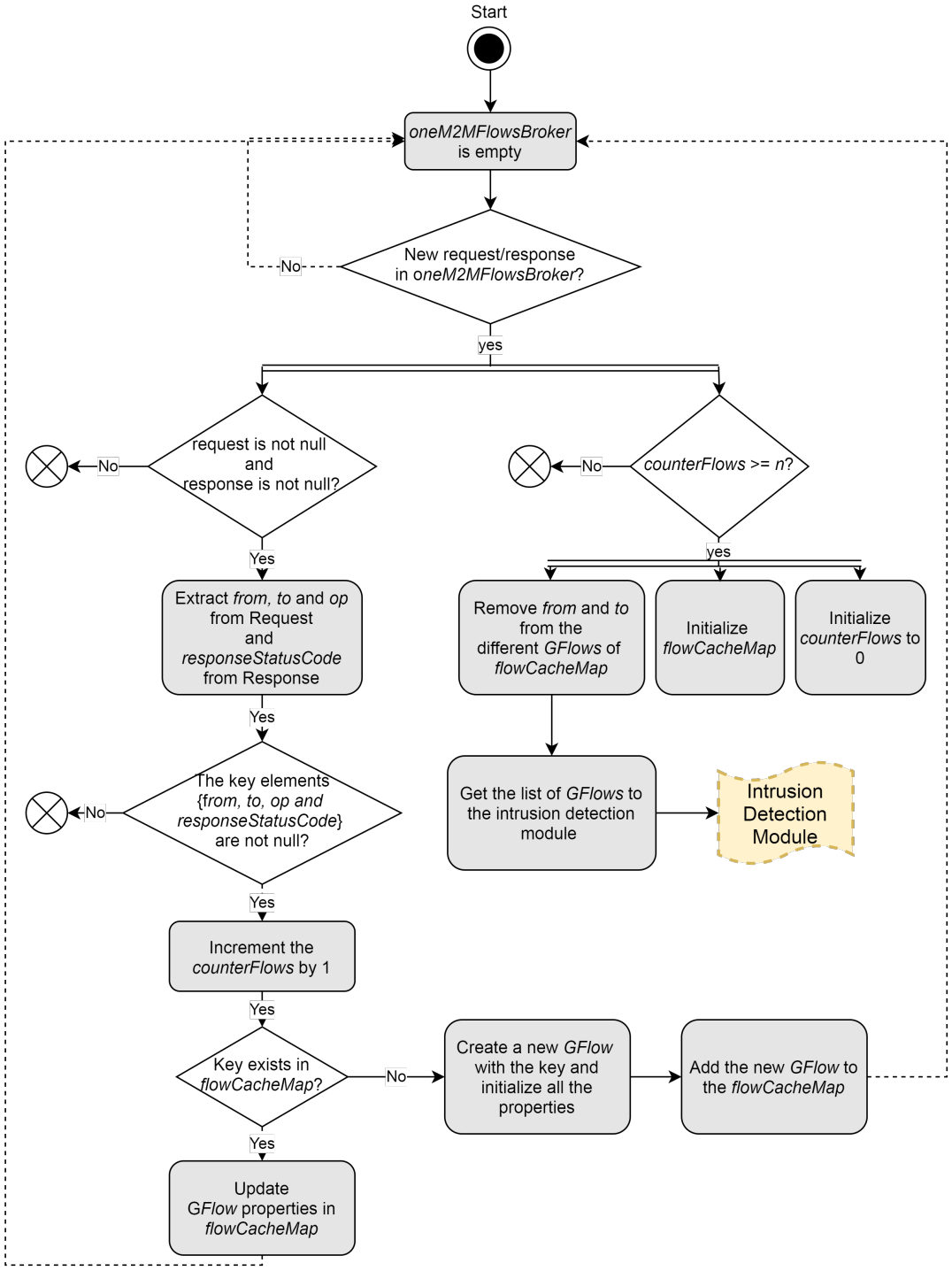

Figure 4.3: Workflow of the oneM2M features extraction module

ker. *counterFlows* reflects the number of valid exchanged flows, knowing that a flow is considered as valid when request and response are not null and that the keys from, to, op and responseStatusCode are not null, as well. The *flowCacheMap* is the Java map where we aggregate the *GFlows* and store the different features. We start with an empty *oneM2MFlowsBroker*. As soon as a new exchange of request/response flow takes place, we verify if the *counterFlows* has reached n. If it is the case, we remove *from* and *to* and send the *GFlow* to the IDS module. At the same time, we initialize the *flowCacheMap* and the *counterFlows*. If *counterFlows* has not yet reached n, then we increment it by 1. If the extracted key exists already in the *flowCacheMap*, we just update it's properties. If it does not exist, then we create a new *GFlow* with its key and its properties and finally add it to the *flowCacheMap*.

4.2.2 Intrusion Detection

The data acquisition and the features extraction phase has generated a list of *GFlows* which is the input to the IDS module. In order to have an efficient detection, we treat the *GFlows* in a "last in first out" (LIFO) order (Figure 4.4). In other words, we start by analyzing the last captured *GFlow* to quickly detect the threat and to get a real-time view of what's going on. All the *GFlows* will be analyzed in the end but with a notion of priority for the most recent exchanged messages. The processing of all *GFlows* is important in order to have a complete trace of the devices status over time.

After each analysis of a *GFlow*, the state "*NORMAL*", "*UNKNOWN THREAT*" or the exact type of the threat, is sent to the cloud in a Json format. The message contains the device identifier, the message identifier, the *GFlow* timestamp, the state as well as a description if needed. The cloud receives the state messages from the different devices as shown in Figure 4.4 and displays, thanks to the *Codex Data Platform IoT*, an overview of the devices in our IoT system, their current state as well as the history of their change of state. All these information are stored in the cloud for two main reasons. First, because we are in the context of resource-constrained devices, so we do not want

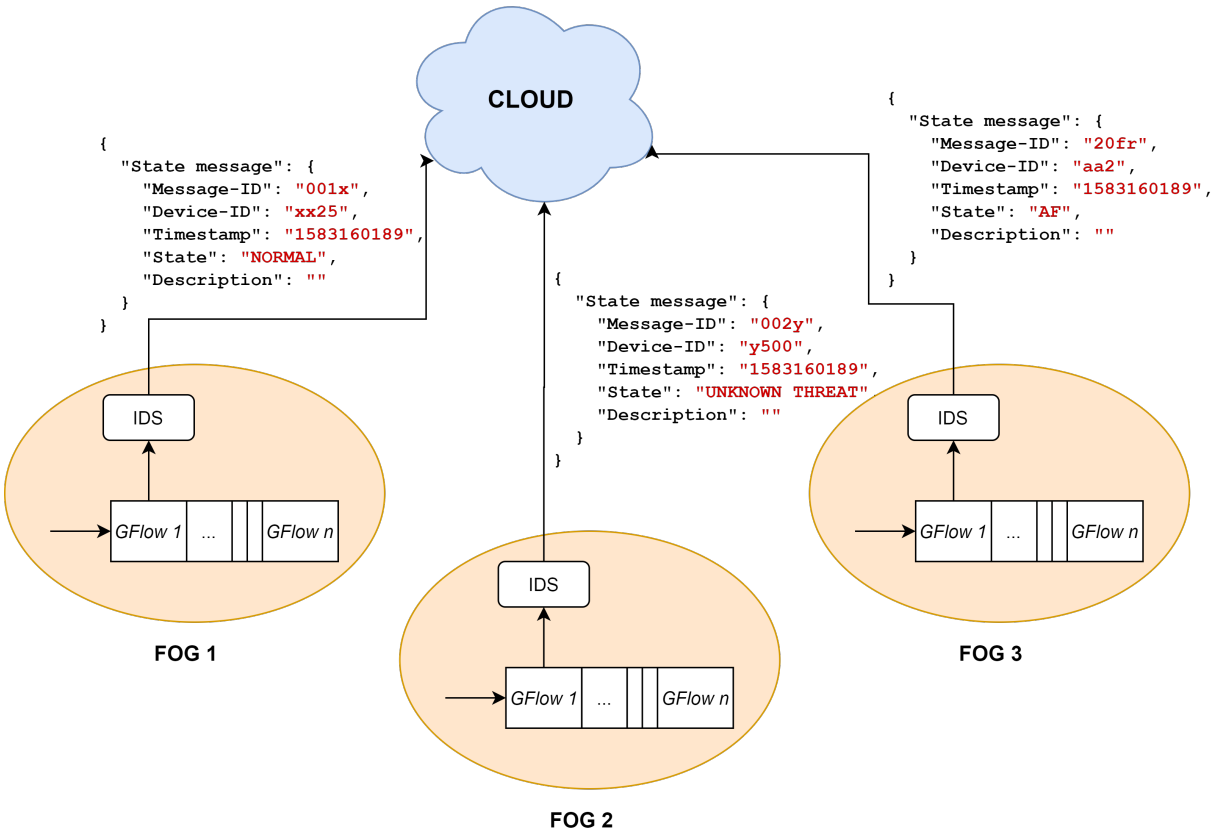

Figure 4.4: Fog to cloud state message

to overwhelm them. Second, to prevent their theft or deletion in case the device is infected or turned off.

The IDS module is built on a fog-to-things architecture where we push the intelligence and the processing logic down near to data sensors. Fog computing concept was first introduced by Cisco [Cis12] to extend cloud computing at the network layer. The fog layer is between the IoT sensors and the cloud. Fog computing places the intelligent processing and the computing power in the local area network level of the network architecture which means in hubs, routers or gateways (fog nodes). Consequently, building our IDS with a fog strategy guarantees the autonomy of the devices in their security with low latency. Given the effectiveness of machine learning algorithms for intrusion detection in general and zero-day attacks in particular compared to the traditional systems [CMZ+19], we have chosen to rely on them.

Therefore, in order to ensure a light detection given the resource constraints of the IoT devices, we have divided the detection into three levels of ML. As shown in Figure 4.5, a *GFlow* is first analyzed in a binary classification. It is either a benign input or a threat. If it is a benign *GFlow*, no action is taken. However, if it is a threat, further analysis is needed. By abandoning the in-depth analysis of normal flows as soon as possible, overloading and unnecessary consumption of resources is avoided. If the *GFlow* was identified as a threat to the IoT system, the *GFlow* needs to be analyzed more deeply to identify the exact type of the threat. It is where the second and third levels of detection come in. The *GFlow* can be classified as one of the main classes of oneM2M attacks (oneM2M flooding, oneM2M amplification, etc.). Otherwise, this *GFlow* is considered as a new threat that will be analyzed and added to the threats database in the next step of the IDPS flow. Finally, if the *GFlow* is classified as oneM2M flooding or amplification, further classification needs to be carried out by the third level to identify the exact type of the attack. This detailed classification allows the identification of the subclass of the threat in addition to its main family.

Such a fog three-level detection allows on the one hand the autonomy and the fast response in order to reduce the potential damage a threat can cause. On the other hand, it allows stopping the analysis as soon as possible, especially for benign inputs, which represents an interesting strategy for a better resource consumption in the context of IoT. The choice of the ML algorithms for the three levels as well as details about their deployment in the IoT system will be the main subject of the next Chapter 5.

4.2.3 Intrusion Prevention

This module allows to warn the security administrator when an intrusion is detected and to apply the appropriate countermeasures in order to protect the system. The ultimate goal is to prevent infection of targeted devices and to avoid the corruption of the entire system caused by security attacks or abnormal behaviour. We detail in this section the prevention workflow as well as the defensive actions against the known threats.

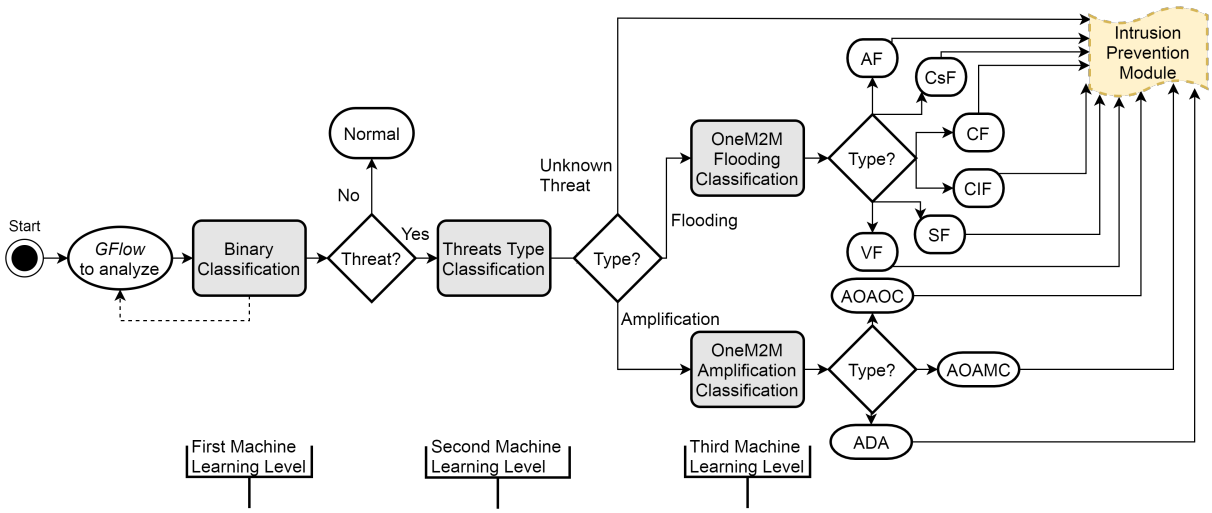

Figure 4.5: Flow chart detection

4.2.3.1 Prevention Workflow

This prevention phase consists of two types of response depending on the detected threat: passive prevention and active prevention. As soon as a threat is detected at the first ML detection level (locally in the fog node), an early warning is sent to the security administrator as shown in the prevention workflow in Figure 4.6. This first alert warns the administrator about an in-progress threat against the concerned device. The threat alert is sent as soon as possible so that the operating environment can be prepared in advance in case of a serious matter. Once the second ML level detection is complete and the threat is identified as a known one, the device gets the exact type of threat from the third ML level and autonomously apply the already associated security action. The prevention actions (of the threats already taken into consideration by the ML models) are stored in the fog. A new status alert will be sent to the administrator to inform them of the current status of the device. These first steps are part of the passive phase that does not require human intervention. The IoT system is capable of protecting itself autonomously against the already known threats.

If the second ML level tagged the *GFlow* as unknown, the active prevention phase starts. For this phase, we need the threats database which contains the different *GFlows*

that were identified previously as unknown and had not yet been processed by the continuous learning module. They can be already annotated by the administrator. In this case, the threats name is updated. If the administrator had also defined a security action for this threat, the table associating threat name to action will be also updated. This database is stored in the cloud to avoid the memory consumption of the fog layer. It is also shared for the entire IoT system.

The fog node verifies if the *GFlow* exists in the threats database. If it is annotated then it verifies if there is an associated action. In this case, the action is applied and a status alert is raised as explained in Figure 4.5. If no annotation is provided then an annotation alert is raised and if it is the same for the prevention action then it is the action alert that is sent to the security administrator.

In case the unknown *GFlow* does not exist in the database, it will be added and an annotation alert is triggered to capture the attention of the IoT human operator for an urgent intervention. The annotation procedure of the *GFlow* as well as the definition of the associated action need to be performed. The security administrator conducts his inspection about the suspicious *GFlow* by examining in detail the logs as well as the various provided indicators. The *GFlow* is enriched, on the one hand, by the source and destination addresses as well as the timestamp of the incident. On the other hand, data related to the device resource consumption / use are displayed in the *Codex Data Platform IoT* such as the disk, the memory, the network, the processes, the processor and the usb ports. The annotation is divided into families of threats and its sub-types such as flooding and amplification and their sub-types. The threat family must first be specified. If there are several threat types in the same family, then the exact type must be mentioned. Once the annotation is done and the prevention is defined, the action will be applied and the database will be updated.

Therefore, if the *GFlow* is unknown and no action is taken after five minutes, the infected device will be isolated from the network in order to avoid the whole system infection.

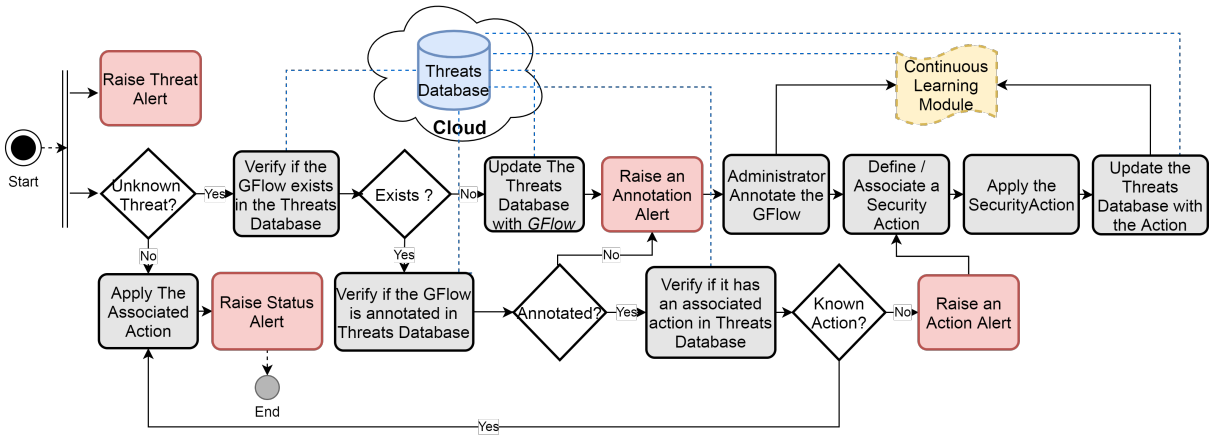

Figure 4.6: Flow chart prevention

4.2.3.2 Prevention Actions

In order to protect the IoT system against the already known threats (flooding and amplification), we use the ACP tool which is a security mechanism that regulates services and data access authorizations for already authenticated entities (Section 3.1.2). As presented in Section 3.1.1, the ACPs are a set of access control rules defining which entities (defined as accessControlOriginators) have the privilege to perform certain operations (defined as accessContolOperations) within specified contexts (defined as accessControlContexts) and are used by the CSEs in making access decision to specific resources. An ACP resource is composed of two main fields namely *privileges* and *selfPrivileges*. The field *privileges* control just the permissions of one resource over another with a list of access control rules. Similarly, the *selfPrivileges* are included in the ACP resource, but the rules present are only applied to the resource itself (the ACP itself). A rule consists of:

- a list of resources for which the rule applies: accessControlOriginators (*O*),
- and authorized operations : accessContolOperations (*Op*).

The value to be applied for the permitted operations is the sum of the corresponding values presented in Table 4.2. For example, if the ACP value is 20 then the authorized operations are updating resources and receiving notifications since 20=16+4.

Value	Operation
1	Create
2	Retrieve
4	Update
8	Delete
16	Notify
32	Discover

Table 4.2: Possible operations in an ACP

In the rest of this section, we define the prevention actions against the oneM2M flooding and amplification threats detailed in Section 3.2.2. The general idea is to define (or modify) an ACP for the threatened resource *R* against the originator *O* of the threat in order to prevent misused operations *Op* from compromising the target device. Since an ACP can be defined mainly for a CSEBase resource, an AE or a RemoteCSE, the resources Container, ContentInstance and Subscription inherit the ACP of their parent/ancestor resource.

We make the following assumptions for the remainder of this section:

- the author of the threat *O* does not have the rights to create, update or delete the *selfPrivileges* of the ACP in *R* if it exists,
- *R* itself has full rights on the *selfPrivileges* of the ACP if it exists,
- the proposed preventive actions are carried out by *R* itself.

For each threat, we need first to verify if there is an ACP that is already defined for *O* in *R* as detailed in Figure 4.7. If it does not exist, then we create a new ACP with 0 for both *selfPrivileges* and *privileges* for *O* in *R* to block all operations that *O* is trying to process. If an ACP is already defined, then we start by blocking all actions on *selfPrivileges*. After that, we verify if the threatening *Op* is allowed for *O* in *R*. If yes, we update the ACP with the old value minus the value of *Op*. We do not update the ACP with 0 for this exact case to guarantee the continuity of the rest of the services between *O* and *R*. However, if we find that *Op* is already not allowed, this means that there is an

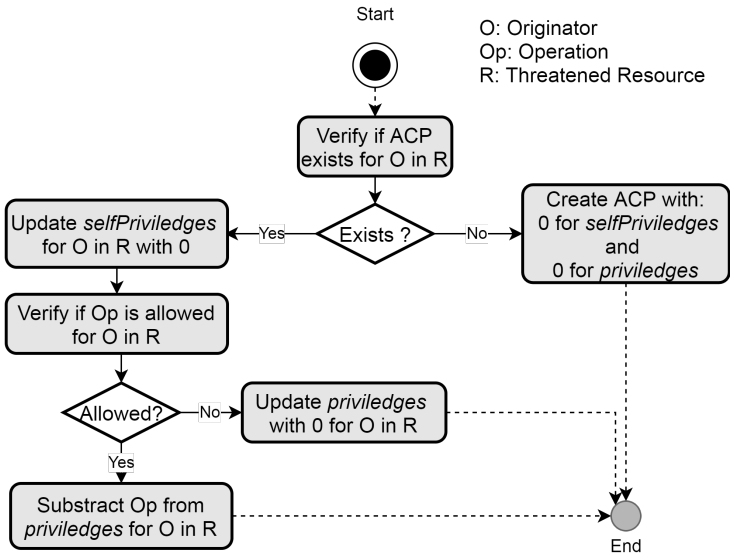

Figure 4.7: Prevention action workflow

abnormal situation since the exact *Op* is the one who threatened *R*. Hence, we update *priviledges* with 0 for *O* in *R* to block all services for this suspicious situation.

In the following, we examine flooding and amplification cases one by one to demonstrate the exact actions (summary in Table 4.3). We consider that there is already a defined ACP with the operations at risk that are allowed (which is the most common case). The core of AF, CF, CIF and SF attacks is the excessive retrieving of an AE resource A or its children resources. Hence, the *Op* that must be blocked is the retrieve. So we need to subtract 2 from the original ACP value for *O* in A (since the Container, ContentInstance and Subscription resources inherit the ACP of their parent). In CsF, unlike the latter, it is about retrieving all the Container resources of an AE. To do that, it is the discovery operation that is used and not the retrieve so we need to subtract 32 from the original *Op*. VF is about retrieving various resources (with retrieve or discovery operations) from a given CSEBase. Therefore, we subtract 34 to block both the retrieve (2) and discovery (32) for the CSEBase resource. The same action needs to be processed for children if they have their own ACP. If they do not, they will inherit those of the CSEBase. Regarding the amplification threats, they involve two resources A and B to set up the attacks. In AOAOC, B will subscribe multiple times to the same

Container of A. Each time a new ContentInstance is created under A, B will be notified. To stop this threat, the creation of subscriptions needs to be blocked for B (so minus 1) and both notification and creation mechanisms need to be denied for A (minus 17 for the ACP of A). Same for AOAMC expect that we need to block discovery also for A since B will discover all the Container resources of A in order to subscribe to them at the beginning (so in total minus 49). ADA has the same principle as AOAMC however, it concerns all the contained AE under the same CSEBase resource as A. Thus B subscribes to all the containers of all the AE resources at the same level as A. Consequently, the minus 49 should be applied to both the CSEBase of A and all the AE resources at the same level as A.

Threat Name	**Op to block for O in A**	**A Resource Type**	**Op to block for O in B**	**B Resource Type**
AF	Retrieve (2)	AE	—	—
CsF	Discover (32)	AE	—	—
CF	Retrieve (2)	AE	—	—
CIF	Retrieve (2)	AE	—	—
SF	Retrieve (2)	AE	—	—
VF	Retrieve & Discover (34)	CSEBase and its AE children	—	—
AOAOC	Create & Notify (17)	AE	Create (1)	AE
AOAMC	Create & Notify & Discover (49)	AE	Create (1)	AE
ADA	Create & Notify & Discover (49)	CSEBase and its AE children	Create (1)	AE

Table 4.3: Prevention actions for flooding and amplification

4.2.4 Continuous Learning

Threats against IoT evolve day by day into new variations (for the already known threats) as well as into new types that are completely different. As a result, intrusion detection models and prevention actions need to be updated frequently and smoothly on an automatic basis. This requirement is generally forgotten or neglected in the state of the art. The focus is more on the efficiency of detection than on the evolution and

adaptation of the system over time. It is true that machine learning techniques have the generalization power to detect threats similar to those already learned. However, it is less effective against significant changes. Consequently, we decided to add a module to our IDPS strategy to allow the system to autonomously learn about new threats and continuously update itself. This module uses the human annotation of the IPS phase to update the ML models of the different levels of the IDS module. We distinguish different cases where the ML models need to be updated. Depending on the case, the concerned ML level that requires updates differs. Before detailing the update cases, it is important to mention that there are two types of updates:

1. re-training or extending an existing ML model to take into account new behaviors / inputs (the new model will extend or replace the old one),

2. training a new model, previously non-existent, to have a new type of classification (the new model will be added next to the old ones).

Both the retraining and the training of a new model are carried out in the cloud (using threats database) to avoid the exhaustion of the devices resources. Once the models are ready, they are automatically injected in the corresponding devices (the size of the models will be studied in the next chapter). It is important to specify that before any ML training, the continuous learning module checks whether the data is sufficient and balanced between the different classes to be learned since imbalanced data degrades efficiency [Kra16]. Once a *GFlow* of the threats database has been considered in the models training, it will be deleted from the threats database and inserted in the learning database which is the one storing all the data used for training the models. This module will be activated in the following cases;

- When a new family of threats has been introduced by the security administrator while annotating the unknown *GFlows*, it is the second ML level that needs to be retrained or extended to take this new family of threats into consideration. Consequently, the old model that was able to differentiate only between unknown

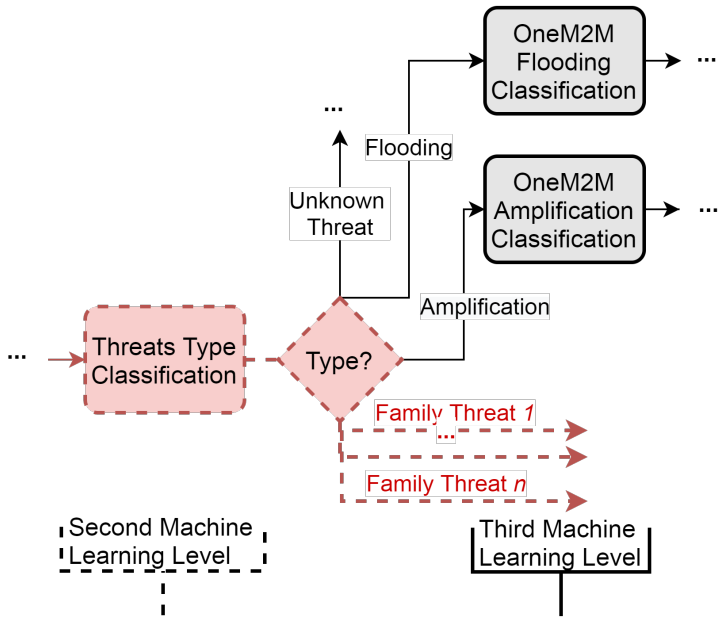

Figure 4.8: Continuous learning of families of threats

threats, flooding and amplification, will be able to detect also the new family as shown in Figure 4.8 with the dashed shapes. This update will be triggered automatically with no human intervention.

- When different types of threats emerge within the same family, this time it is the third level of ML that needs to be expanded with a new model that will classify these different types of threats. So like the sub-types of flooding and amplification in the third level, we will have a new model that will differentiate the sub-types of an other family as presented in Figure 4.9 with the dashed shapes. This case also will be initiated automatically with no human intervention.

- When the definition of normal *GFlow* changes, it is the first level of detection that needs to be updated. There are two possible cases:

  - a *GFlow* that was detected as threat in the first ML level, was annotated as normal by the security administrator. In this case, the first ML level needs to be automatically retrained to improve its efficiency.

  - the administrator has made changes in the parameters of the device which

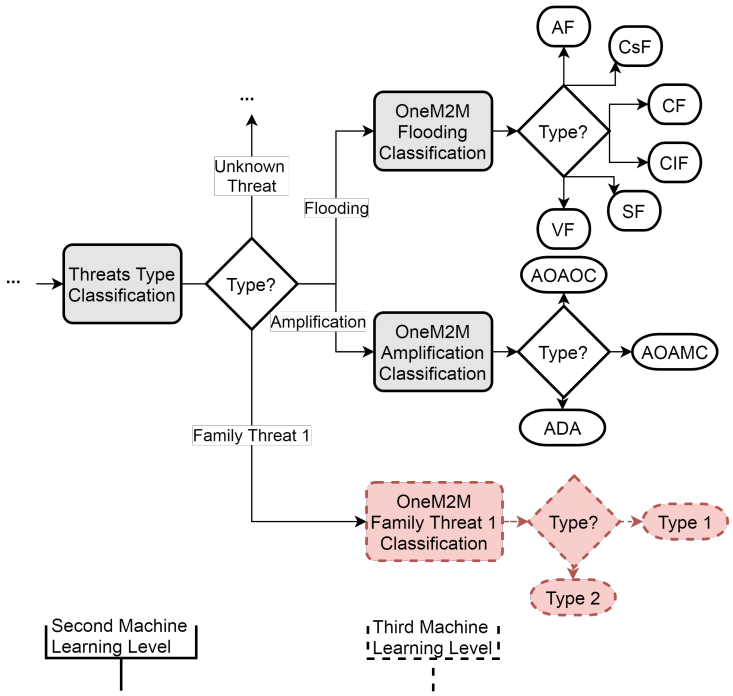

Figure 4.9: Continuous learning of new threat types under the same family

causes the modification of the definition of normal *GFlow*. Here the model needs to learn the new normal behavior. This case is launched by the administrator and not automatically by the system.

- When the whole detection system starts to raise an important number of false alarms or miss some threats, one or multiple ML detection levels could be retrained. It is the choice of the security administrator to choose and retrain manually one or more models. Our IDPS has a graphical interface where the administrator can verify the state of the devices and choose the ML models to retrain and inject in the IoT devices. Such a mechanism is guaranteed by the device management module offered by *Codex Data Platform IoT*. It is based on Open Mobile Alliance Lightweight M2M [DB15] (OMA LwM2M) that provides self-management of the M2M device configurations.

Prevention actions (used by the prevention module) are first stored in the cloud when the administrator defines them. Once new threats are handled by the continuous

learning module (used in the models training), the list of associated actions is copied from the cloud to the fog nodes to allow the devices to react autonomously.

To conclude, the proposed approach continuously integrates knowledge of newly discovered threats as well as changes, evolutions and new configurations into the basic behavior of the IoT system.

Conclusion

In this chapter we have presented the different challenges and goals of our IDPS proposal, namely interoperability, autonomy, scalability and resource constraints respect, modularity in design, adaptability and extensibility and finally the active and real-time reaction. Furthermore, we explained our strategy and detailed each module of our IDPS. We revealed the two possible ways to acquire oneM2M data after having presented the state of the art of the existing open source tools. We detailed the feature extraction approach. Moreover, we specify the detection methodology and the different steps and principles. Then, we described in details the prevention procedure. Finally, we presented the idea of continuous learning that characterizes our IDPS. The next chapter will concentrate on the experimentation of machine learning algorithms in the intrusion detection module.

Chapter 5

Machine Learning and Deep Learning for OneM2M Intrusion Detection

Contents

Introduction

In the previous chapter, we detailed the strategy of our oneM2M-IDPS. It starts by collecting and analyzing the exchanged messages in order to generate the needed abstraction (*GFlow*). Second, it detects and classifies threats on three different levels. It responds immediately to potential threats with appropriate actions and finally continuously learns new behaviors in order to be up to date. We concentrate in this chapter on the detection module (introduced in Section 4.2.2) with its three Machine Learning (ML) levels. We experiment with different ML and DL algorithms for each detection level in order to choose the most appropriate and efficient ones. We start this chapter by explaining our choice to adopt ML and Deep Learning (DL) techniques for the detection levels. Then, we present the metrics we relied on to evaluate the effectiveness of each algorithm in our oneM2M intrusion detection context, as well as the experimental environment. Furthermore, we concentrate on each detection level experiments; we introduce the used ML and DL algorithms (definitions, tools and frameworks), display the results and compare them with each other to finally choose the most appropriate one for each detection level. In addition, we are conducting some experiments on the effect of training data size and balance on detection results. In the last section of this chapter, we examine the one-class classification approach for both the first and second ML detection levels. For the first level which is the most crucial one that will affect the overall oneM2M-IDPS performance, such an approach highlights the detection of any behaviour different from normal (known for anomaly detection). For the second level, the one-class classification approach will be used to enable the unknown threats detection (called novelty detection).

5.1 Learning Techniques Adoption and Metrics

As presented in Section 2.2.1, there are mainly three types of ML algorithms:

- Supervised learning is based on learning from labeled training data.
- Unsupervised learning is based on clustering the input data into classes on the basis of their statistical properties. It uses unlabeled data.
- Semi-supervised learning is a combination of supervised and unsupervised machine learning methods. Thus, training data includes both labeled and unlabeled data.

5.1.1 ML Adoption

Researchers are putting more and more energy in exploiting ML algorithms in IDS for many reasons [CMZ+19]:

- Unknown/zero day attacks bypass traditional signature-based IDS; whereas supervised machine learning algorithms have an interesting potential in detecting new attacks.
- Traditional IDSs suffer from high false recognition rate which can be reduced with machine learning techniques. Compared to Signature/Non-Signature IDS, an ML equipped IDS employs statistical, genetic and heuristics or a combination of them to disseminate complex attack pattern to improve the detection rate with reduced False Negatives.
- Traditional solutions struggle with attacks that have complex properties, while machine learning can improve their detection accuracy and speed.
- Slight variations in attacks can not be effectively detected with traditional IDSs. Even a Heuristic detector can be evaded by inverting the attack pattern. However,

ML equipped IDSs learn recent traffic pattern continuously; hence, they effectively identify minor variations in traffic patterns. In other words, ML algorithms are efficient against the detection of variants [MVTP18].

- Cyber criminals deploy evolving attack patterns to evade the detectors. Traditional IDSs, especially signature-based IDSs require continuous updates. However, ML IDSs based on clustering and outlier detection do not necessitate regular updates.

- Traditional solutions and more precisely signature based IDSs match each signature with IDS database. The process is CPU consuming (large signature database which grows exponentially [LL05]); whereas, ML based IDS consume low to medium processing. Hence, the processing element can be used effectively.

Consequently, learning techniques seem to be a suitable solution especially with the good results that they achieve in the different domains.

5.1.2 ML Metrics and Experimental Environment

In machine learning, there are numerous metrics that could be used to compare the performance of the ML algorithms. All the metrics are calculated from the basis of the standard confusion matrix of the ML theory which is represented in Figure 5.1 (in its binary version) with:

- TP: True Positive which represents the correct classification of the oneM2M *GFlows* threats as threats.

- TN: True Negative which represents the correct classification of the normal oneM2M *GFlows* as normal.

- FP: False Positive which represents incorrect classification where the normal oneM2M *GFlow* were classified as threats. High FP value increases the computation time since these false alarms need time to be processed (to no avail). How-

ever, it is less harmful than the high FN value (defined below) especially in the security context.

- FN: False Negative which represents incorrect classification where oneM2M *GFlows* threats are predicted as normal flows. The high FN is a serious problem because it means that the threat has managed to get through the IDS.

		Actual Values	
		Threat (1)	**Normal (0)**
Predicted Values	**Threat (1)**	TP	FP
	Normal (0)	FN	TN

Figure 5.1: Confusion matrix

In our study, we concentrate on the following metrics:

- Recall or detection rate reflects the samples that should have been identified as threats. Recall is one of the most important metrics in the security context. In addition to its ability to calculate successfully detected threats, it also reflects the rate of false negatives (FNR), i.e. intrusions that are missed. We will not present FNR in our results comparison since it can be deduced from the recall (as shown below).

$$Recall = TP/(TP + FN) = 1 - FNR = 1 - (FN/(TP + FN)) \tag{5.1}$$

- Accuracy estimates the ratio of the correctly predicted flows to the entire dataset.

$$Accuracy = (TP + TN)/(TP + TN + FP + FN) \tag{5.2}$$

- Precision estimates the ratio of the correctly predicted threats to the number of all predicted threat records.

$$Precision = TP/(TP + FP) \tag{5.3}$$

- False Postive Rate (FPR) represents the false alarms: the ratio of the normal benign flows predicted as threats to all the actual normal inputs.

$$FPR = FP/(TN + FP) \tag{5.4}$$

- Model Size is the size (in kilobytes) of the generated model that needs to be injected in the IoT devices for the ML classification. The smaller the models are the better is since we are in the context of tiny devices (IoT).

- CPU Training Time represents the time needed to build the ML model. This metric is important for the continuous learning module of Section 4.2.4.

The higher the first three metrics are, the better the ML model is. The lower the last three metrics are, the better the ML model is. The ML classifications were made offline (on a machine with Intel(R) Xeon(R) CPU E3-1225 v3 @ 3.20GHz and 16Go of RAM).

5.2 Experimentation of Supervised Learning Algorithms for Intrusion Detection in OneM2M

In this section, we examined different supervised ML algorithms for the three ML detection levels. We chose supervised ML because the experiments conducted in this section fall within the context of classification where we know already the classes into which the model should categorize the *GFlow*. This section will be divided into two main parts. The first section will treat supervised ML detections for the three levels. In the second one, we study the effect of size and balance of training dataset on the detection process.

5.2.1 Supervised ML Detections

We begin this part with the definition of the experienced ML algorithms as well as the used tools. Then, for each detection level, we display the results, compare them and choose the most appropriate for each one.

5.2.1.1 Description of the Algorithms

It is important to remind that our generated oneM2M dataset is labeled as presented in Section 3.3. We chose to experiment four different shallow ML algorithms and a DL network.

- Naïve Bayes (NB) [WF02] is a probabilistic classifier based on Bayes' theorem. It is called "naïve" since the inputs features are simply assumed to be independent of each other, whereas in practice it is rarely true. This hypothesis of feature independence may be the cause of poor results when the data are not really independent of each other. NB classifier can handle continuous and categorical data.

- Support Vector Machine (SVM) [Vap13] is a classifier based on finding the best hyperplane (in the feature space) separating two data classes by maximizing the distance between the hyperplane and the closest data points of each class. The support vectors (from where comes the name of the classifier) are data points that lie on the margin of optimum separating hyperplane. If this hyperplane decision boundary has succeeded in separating the two classes well, it means that the resolution of the classifier is a linear combination of the support vectors. However, if the classes are not linearly separable, then one solution could be to map the data to a higher feature (dimensional) space where SVM can define a better separating hyperplane. This technique considers the kernel functions. Various types of dividing surfaces can be defined thanks to the kernel method, such as linear, polynomial, Gaussian Radial Basis Function (RBF), etc.

- Decision Tree (DT) is a tree-like structure composed of decision nodes, branches

and leaf nodes. A decision node represents feature (or attribute), the branch represents the conjunction of features that lead to the leaf nodes which represent the classification classes. DT can work with discrete as well as continuous value attributes. It learns to partition on the basis of the attribute value. There are many DT algorithms. The best known are ID3 [Qui86] and C4.5 [Qui14] that build the tree automatically using the concept of information entropy. Each node of C4.5 chooses the best attribute to split the set of data samples into subsets with the highest possible information gain (entropy). This operation is performed recursively on the smaller subsets until all the training examples have been classified.

- Random Forest (RF) is one of the popular ensemble classifiers that combines many DT. Each tree picks randomly a data features as input, processes DT algorithm, then by majority or weighted voting, the forest generates the prediction result. Unlike DT, RF is less humanly interpretable.

- Deep Neural Networks (DNN) [SHM+16] is an application of artificial neural networks (ANNs) with multiple hidden layers. It consists of at least three layers [BPS+02]. The input layer represents the beginning of the ANN workflow. It brings the input data into the network. The output layer decides or predicts the input. And finally one or more hidden layers are between the input and the output. They perform the needed computations for the ANN. The nodes of the hidden and the output layers are neurons with a nonlinear activation function. DNN is capable of approximating any continuous function and distinguish non-linearly separable data. It uses backpropagation technique for training. Hence, its training is about feedforward and back-propagation phases. In the first one, the hidden and output nodes calculate their activation functions. The second phase aims to propagate back the error (the difference between the output and the target value) from the output to the input. This step is about adjusting the different weights of the different neurons composing the network.

5.2.1.2 Used Tools and Frameworks

For the four first ML algorithms, we use Weka 3 tool [HFH+09] (developed in Java) which offers a collection of ML algorithms for data mining tasks in an easy-to-use software. It supports different modes of use: command line, GUI, Java API, and so on. We first use the GUI mode to facilitate the testing of the algorithms, then once the best algorithm is chosen, we integrate it into our IDPS using the Java API since the used oneM2M implementation (*Codex Data Platform IoT*) is developed in Java. Implementations of the NB and RF classifiers in Weka are named after the original algorithms. However, for DT, we work with J48 [Sal94] which is an open source Java implementation to generate a pruned or unpruned DT. For SVM, we adopt SMO [ZYX+08] which is the implementation of John Platt's Sequential Minimum Optimization algorithm for the SVM classifier.

Regarding the parameters of the algorithms, we tried different configurations, either for the first, or for the last detection level. In the end, as there are no big changes in results, we chose the default parameters as proposed by Weka.

For the DL, we used the Python Deep Learning library Keras [Ker19b]. In order to obtain good detection results with DNN, different hyper-parameters have to be adjusted until the best combination is found. There are two types of hyper-parameters:

- hyper-parameters for the layers:
  - the number of hidden layers reflects the number of layers deployed outside the input and output layers,
  - the number of units in each layer which is an integer hyper-parameter that represents the dimensionality of the output space,
  - the activation function [DA01] for each layer is an important feature in the artificial neural networks. It decides the relevance of the neuron information with a non linear transformation.

  - the kernel initializer [Ker19c] for each layer defines the way to set its initial weights,

- hyper-parameters for the learning process [Ker19b]:

  - the loss function which represents the objective function that the model will try to minimize by changing the parameters (weights) of the model,
  - the optimizer in machine learning is a function that describes how to adjust the parameters of the model for the loss minimization.

Scikit-learn library [noae], a general ML library built on top of NumPy[1], was also used for pre-processing operations such as the standardization and scaling of data.

5.2.1.3 The First Level of ML Detection

Our first experiments in the search for the best algorithm have been applied to our oneM2M dataset (Section 3.3). We divide it into two files: the training dataset which is composed of 66% randomly chosen inputs and the test dataset which contains the 34% remaining values. The *GFlows* in both files are labeled as normal or threat. As discussed earlier, the parameters of shallow algorithms were the default ones proposed by Weka. However, for DNN we explored different combinations and the best results were achieved by a network of three fully connected layers. The input layer was initialized by normally distributed weights (*random_normal* initializer) and has a Rectified Linear Unit [Ker19a] (*ReLu*) activation function. The second layer is composed of eight units with a *random_normal* as a kernel initializer and the *sigmoid* [Ker19a] function for the activation. Regarding the output layer, it is a two units layer since we are in the context of binary classification. *Softmax* was the used activation function to guarantee a probability between 0 and 1 for each classification. The used loss function was the *binary_crossentropy* and the optimizer was the *Stochastic Gradient Descent (SGD)* optimizer.

[1]NumPy is the fundamental package for scientific computing with Python.

ML algorithm	Recall (%)	Accuracy (%)	Precision (%)	False Positive Rate (%)	Model Size (Ko)	CPU Training Time (ms)
NB	79.70	71.05	80.90	53.70	**10**	**800**
SMO	**98.60**	84.14	83.20	57.10	15	36 968 840
J48	95.40	**87.81**	**88.90**	34.00	301	27 490
RF	89.90	83.84	88.50	**33.50**	307 673	138 010
DNN	97.41	86.91	86.61	**13.39**	18	596 463

Table 5.1: Comparison of the results of the binary classification before the removal of duplicates

As we can notice in Table 5.1, SMO achieves the best attack detection rate (recall) of 98.60%, followed by DNN and J48 with 97.41% and 95.40% respectively. However, SMO has the worst CPU training time. NB has the fastest learning phase and the smallest model compared to the rest. It reaches 800ms with 10Ko. Unfortunately, NB has the poorest accuracy. It is J48 which has the best results in terms of accuracy and precision, with 87.81% and 88.90%. It is true that it has a larger model than the NB but 301Ko is still acceptable for IoT fog nodes. In addition, J48 is the second fastest algorithm to train. Regarding FPR, it is DNN which achieves the lowest rate with 13.39%, followed by RF (33.50%) and J48 (34.00%). Therefore, considering the overall metrics, we can say that J48, RF and DNN are the most efficient ones. But since RF model is heavy (138 010Ko), J48 and DNN remain the most appropriate algorithms for our binary classification task of the first level of ML detection.

After these results which were published in [CMZS19, CMZS20], we thought to remove the duplicate entries from the oneM2M dataset which represent 4.27% of the initial data. The new results are compared in Table 5.2. As always, we use the default parameters for NB, SMO, J48 and RF. For DNN, we find that the best results were achieved by a network of three layers, as for the first case, however, the hidden layer is composed of 24 neurons with a *ReLu* activation function and we use *categorical_crossentropy* as a loss function and *Adam* as an optimizer. The network was trained for 200 *epochs* with a *batch_size* of 400.

As can be seen, the performance of NB, SMO, J48 and RF have improved proportionally compared to Table 5.1 (with a special enhancement for RF). However, DNN has

ML algorithm	Recall (%)	Accuracy (%)	Precision (%)	False Positive Rate (%)	Model Size (Ko)	CPU Training Time (ms)
NB	82.40	73.63	82.30	52.40	**10**	**620**
SMO	**98.70**	85.10	84.10	55.3	15	10 854 250
J48	94.90	**89.28**	**91.10**	27.30	285	30 950
RF	92.50	87.78	**91.30**	**26.20**	172 360	172 618
DNN	95.10	87.01	88.40	36.93	21	162 000

Table 5.2: Comparison of the results of the binary classification after the removal of duplicates

poorer recall and FPR compared to its results with data containing redundant entries (especially for PFR: from 13.39% to 36.93%).

Consequently, by process of elimination, we decide to use the J48 model for the first level of intrusion detection of our strategy.

5.2.1.4 The Second Level of ML Detection

As we have discussed in Section 2.1, the second ML level detection identifies the threat family of an incoming *GFlow* if it is a known threat (flooding or amplification). If it is a new one that we have not seen before, the model needs to classify it as unknown. In this section, we will experiment only the classification of known family threats. The whole model with unknown threats detection will be discussed later in this chapter.

In this part, we use only the *GFlows* that are considered as threats in the oneM2M dataset (Section 3.12) with the labels corresponding to either flooding or amplification.

In Table 5.3, we compare the results of different ML algorithms for threat family classification experiments. J48 and RF achieve the best recall (100%), accuracy (99.98%) and precision (100%) with zero FPR. However, J48 is lighter and faster to train. DNN has close results (two fully connected layers with 2 neurons for the output layer, *softmax* as an activation function and *categorical_crossentropy* as a loss function). Regarding NB, it has the fastest CPU training time (1 080ms) and the smallest model (1Ko). Meanwhile, it has the worst results in terms of the remaining metrics.

ML algorithm	Recall (%)	Accuracy (%)	Precision (%)	False Positive Rate (%)	Model Size (Ko)	CPU Training Time (ms)
NB	95.30	95.30	95.60	3.30	**11**	**1 080**
SMO	99.90	99.88	99.90	0.20	15	556 710
J48	**100**	**99.97**	**100**	**0**	22	11 230
RF	**100**	**99.98**	**100**	**0**	1 761	113 850
DNN	99.87	99.90	99.97	0.05	16	15 696

Table 5.3: Comparison of the results of the classification of threat families

5.2.1.5 The Third Level of ML Detection

The third ML level tends to identify the exact sub-type of a threat. At this point, we have flooding and amplification threats in the oneM2M dataset (Figure 3.12). As presented below, we experienced four shallow ML algorithms with Weka tool and DNN with Keras.

Flooding Classification For this experiment, we use only the flooding *GFlows* with the sub-types labels. As reported by Table 5.4, the J48 algorithm achieves the best results with 93.80%, 92.32%, 92.95% and 1.53% of detection rate, accuracy, precision and FPR, respectively. Even though, the NB algorithm is the one which generates the smaller model, 290 Ko (the size of the J48 model) is still always acceptable for the IoT context. Hence, we decide to adopt the J48 algorithm for the flooding classification.

DT algorithms use the entropy computation [WS84] technique for the features reduction. Tree based models calculate feature importance to keep the best performing features as close to the root of the tree. By analyzing the generated tree of J48 algorithm for flooding types classification, we remark that it eliminates 6 features from the oneM2M dataset features (Table 3.2): isSameFromTo, isToRemote, fromResourceType, counterSameFromOperationResponseType, counterSameToResponseType and counterSameOperationResponseCategory. To inject a lighter model into the fog nodes, we trained the flooding models with only the relevant features.

Amplification Classification Regarding the amplification classification (Table 5.5), the J48 algorithm achieves only the best accuracy (65.04%). SMO seems to have better

ML algorithm	Recall (%)	Accuracy (%)	Precision (%)	False Positive Rate (%)	Model Size (Ko)	CPU Training Time (ms)
NB	73.90	73.87	40	5	**24**	**530**
SMO	80.90	80.93	88.10	5.10	26	1 568 690
J48	**93.80**	**92.32**	**92.95**	**1.53**	290	9 280
RF	89.90	89.88	89.90	2.4	196 773	66 230
DNN	86.62	82.60	83.82	3.52	32	152 435

Table 5.4: Comparison of flooding-classification results

ML algorithm	Recall (%)	Accuracy (%)	Precision (%)	False Positive Rate (%)	Model Size (Ko)	CPU Training Time (ms)
NB	49.00	49.03	46.70	27.70	**12**	**320**
SMO	**68.40**	58.96	**64.60**	17.90	16	1 505 340
J48	62.77	**65.04**	62.83	17.07	953	12 500
RF	63.70	63.74	64.00	**16.80**	211 092	41 560
DNN	61.28	63.43	60.81	**16.80**	29	126 504

Table 5.5: Comparison of amplification-classification results

recall (68.40%) and the best precision (64.60%). It is true that it does not have the best accuracy (6% less than J48), nor the best FPR (1% more than RF and DNN), nor the smallest model (4ko more than NB) but it remains the best in terms of overall performances. The only problem with SMO is its long CPU training time (about 25 minutes). Consequently, we decide to deploy the SMO model for the amplification classification.

5.2.2 Effect of Dataset Size on Detection Results

In this section, we consider two different cases related to data size that can affect the continuous learning module presented in Section 4.2.4. As detailed, this module needs to update ML detection models. For a quick training as well as an efficient consideration of new upcoming *GFlows* (without waiting for a large data availability), we need to find the minimum data size necessary to generate a reliable model. Fast training with a small amount of data generates lightweight and easily updatable models which is important in the security domain and interesting in the context of IoT and fog computing. First, we study the evolution of the models performances in binary classification (first ML

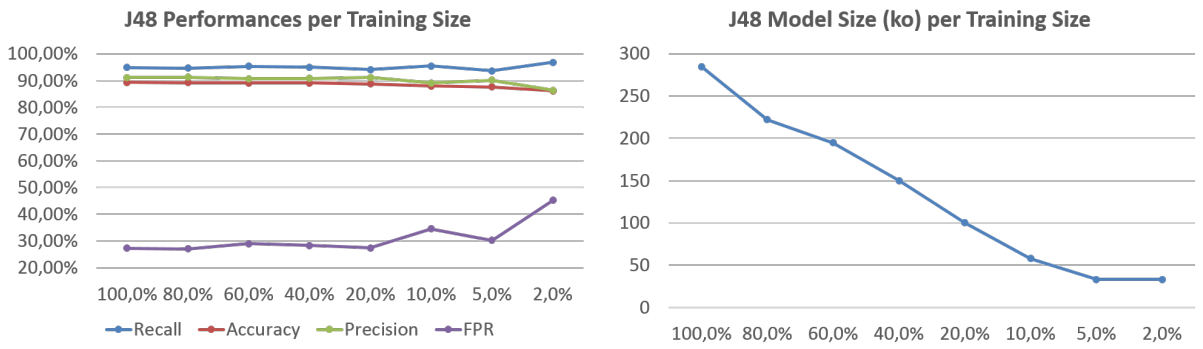

Figure 5.2: Effect of training dataset size

level) while decreasing the dataset size. The second experiment concerns the effect of data balance/imbalance on model performance (also for the first detection level). The testing dataset was extracted before the two experiments (34% of the whole dataset) and the percentages of data removed were chosen randomly. Both experiments were conducted on oneM2M dataset without redundant entries. In addition, we use the J48 algorithm since it had the best results in terms of threat detection for the first ML level as discussed in Section 5.2.1.3

5.2.2.1 Effect of Training Dataset Size

As presented in Figure 5.2 (left side), while decreasing the size of the training data, we note that the measures remain stable at the beginning until only 20% of the initial data remains. After 20%, the FPR starts to increase significantly. As expected, the size of the models decreases as the dataset decreases (Figure 5.2 right side). Consequently, for an acceptable ML performance, we can update the models with only 20% of the total oneM2M dataset size (only 44 654 *GFlows* instead of 223 273).

5.2.2.2 Effect of Balanced / Imbalanced Training Dataset on the Detection Results

We continued our experimentation with the J48 algorithm. Since with only 20% of the initial data, J48 succeeded to have acceptable performances, we experimented our data balance on that basis. We describe the imbalance of classes in terms of a ratio, i.e.

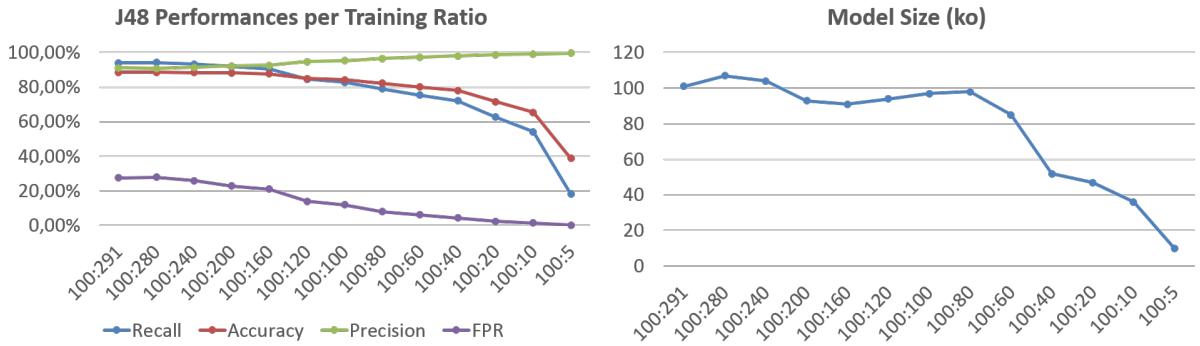

Figure 5.3: Effect of data imbalance (decrease threat *GFlows*)

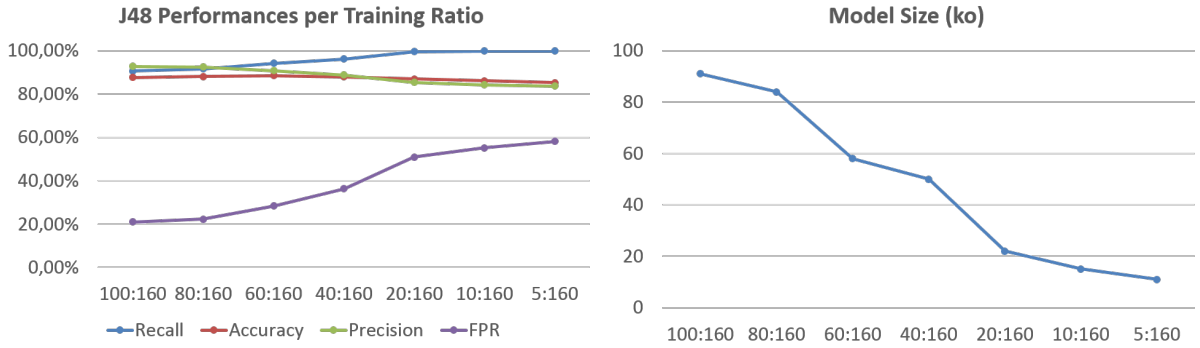

Figure 5.4: Effect of data imbalance (decrease normal *GFlows*)

$1:100$ means that for every one example of normal *GFlow*, there are 100 examples of the other class which is threats. As shown in Figure 3.12, the oneM2M dataset is already imbalanced. First, in Figure 5.3, we fixed the amount of normal *GFlows* in the dataset and started to decrease the size of threat inputs. We found that the performance of the J48 remained acceptable up to the $100:160$ ratio. After that, the recall and accuracy start to drop significantly.

In Figure 5.4, we experimented the opposite. We started from the stable point of the last experiment which is a ratio of $100:160$, we fixed the threat *GFlows* and started to decrease the number of normal inputs. We found that the most stable results are with ratios $100:160$ and $80:160$. Consequently, it is better to update the models starting from a ratio of $80:160$.

5.3 One-Class Classification Approach

In order to have a better intrusion detection at the first ML level of our IDPS, as well as to enable the detection of unknown threats at the second ML level, we wanted to experiment one-class classification algorithms (OCC). For the first level, OCC reflection seems to be more intuitive for this type of binary classification. Since the main goal of this IDS level is to detect any behavior that is different from the normal one (whether it is security attack or just an abnormal functioning of the system), learning only from the normal behavior without considering the threat inputs appears to have more sense. Such concept is a sub-domain of ML known under different terms in the literature as anomaly detection, one-class classification [MKH93], outlier detection [RG97], novelty detection [Bis94] or concept learning [JMG95]. Such ML algorithms are considered as unsupervised since they attempt to model "normal" examples without labels rather than using examples from both classes (normal and abnormal). They classify new examples as either normal or abnormal (e.g. outliers). Consequently, for the first level of our IDS, we trained the model only on normal *GFlows* and then evaluate it with normal and threat *GFlows*. Regarding the second ML level, the main goal is to detect unknown (never seen before) threats on the one hand, and to classify the known threats on the other hand. Consequently, we need to adapt the OCC approach for multi-classification.

We start this section with a presentation of the categories of OCC algorithms, focusing on those that we have experimented for our IDS. Then, we detail our experimentation process for the choice of the best algorithm and parameters. Moreover, we discuss the results and the final choice of the ML algorithm for the first detection level. We conclude this section, with details and experiments about multi-class novelty detection for the second level of ML detection.

For better consistency in definitions and discussions, most of the details in this section will be presented for the first level of the IDS. The second level will only be covered by Section 5.3.3.4.

5.3.1 One-Class Methods

There are three main categories of OCC [Tax02]:

- Probability density estimation relies on estimating the density of the training target class and setting a threshold on this estimation to obtain a target and an outlier region. Several distributions can be assumed [BL78], such as a Gaussian or a Poisson distribution, etc.

- Boundary methods are about optimizing a closed boundary around the target set instead of estimating a complete data density as in the first category of OCC. Unlike binary classification where the decision limit is supported from both classes, the OCC has to decide how narrow the limit should be around the data with only data from one class, which is not always obvious. Some boundary algorithms are nearest neighbor and one-class SVM.

- Reconstruction methods have not been primarily constructed for OCC. Their main goal is to model the data. In the context of OCC, these methods use prior knowledge to generate a model that best matches the target class data in order to classify new entries that belong to the same learned description. The auto-encoder networks are one of the best known examples.

We consider only four OCC algorithms for our oneM2M dataset. One of the boundary category which is one-class SVM (OC-SVM) and three of the reconstruction category namely the auto-encoder (AEnc), the sparse auto-encoder (SAE) and the variational auto-encoder (VAE). We will detail each algorithm in the rest of the section. To apply these methods we used the already splitted dataset used in Section 5.2.1.3 (without redundant entries). We extract 67% of the normal *GFlows* of the training dataset for the OCC algorithms training and use the remaining 33% of the normal *GFlows* with the whole threat *GFlows* of the training dataset for the validation process and the threshold setting (for algorithms in the reconstruction category). The testing phase is accom-

plished with the same test set of Section 5.2.1.3, thus on 34% of the initial dataset. We always use the same test set in order to have comparable results.

5.3.2 OC-SVM

OC-SVM [SWS+00], as the name suggests, is based on the binary classifier SVM. It is used for novelty detection in imbalanced datasets. OC-SVM is an unsupervised algorithm that learns a decision boundary in the feature space that contains all targets. More precisely [PCCT14], "it defines the novelty boundary in the feature space corresponding to a kernel, by separating the transformed training data from the origin in the feature space, with maximum margin". It provides non-linear kernel functions such as the Radial Basis Function (RBF). We use the class *OneClassSVM* of the scikit-learn library for this experiment. *OneClassSVM* has mainly three important hyperparameters to configure: the kernel to use, the kernel coefficient *gamma* and the *nu* hyperparameter that controls the sensitivity of the support vectors. So to find the best configuration for our oneM2M dataset, we varied the hyperparameters and tuned the model until identifying the best results. We use the RBF as a kernel with $gamma = 1/number\_of\_features$ and $nu = 0.038$. Unfortunately, the results are far from being good in terms of FPR (96.30%). For the rest of metrics, we had acceptable results: 97.96% of recall, 74.16% of accuracy and 75.07% of precision.

5.3.3 AEnc, SAE and VAE

Since DNN was one of the best algorithms in Section 5.2.1.3, we decided to experiment neural networks in their one-class version which comes to the use of auto-encoders. We present, in the following, the different AEnc variations that we have examined for our use-case.

5.3.3.1 Algorithms Description

An AEnc [JMG95] is a neural network approach created to build identity for the training data while trying to compress the features space into smaller representation as shown in Figure 5.5. Hence, an AEnc should be able to reproduce the inputs at the output layer. An AEnc [AC15] is composed of two main components: an encoder that maps the input to a hidden representation and a decoder that reconstructs the original input space from the hidden representation by the same transformation as the encoder. The smallest hidden representation is called the latent space, the bottleneck. The training of the AEnc aims to minimize the difference between the input and the output i.e. minimize the reconstruction error (RE). After being trained, the model must be able to rebuild previously unseen instances that are of the same data distribution as the training set. If the new input does not belong to the same distribution, the RE will be high. So the idea behind using this type of algorithms is to train the model to learn only about the distribution of normal *GFlows*. Once an anomalous *GFlow* is given to the network, it will have a high RE. Many reconstruction error functions could be used such as the Mean Squared Error (MSE) which reflects the average squared difference between the estimated values (the output of the AEnc) and the actual value (the input).

SAE [Ng11] is a type of AEnc where we add a sparsity constraint on the hidden layers. In other words, the weights of the AEnc are penalized in the cost function. This technique is mainly used to avoid overfitting. In AEnc, the number of neurons in hidden layers is less than the neurons of the input/output layers. Meanwhile, in SAE, the number could be less or greater. However, thanks to the sparsity constraint, not all the neurons will be "active". Some of them will be disabled as in Figure 5.6 (gray links). A neuron is considered "active" or as "firing" if its output value is close to 1. If it is close to 0, then it is being "inactive". Mathematically, an extra penalty term (penalizing activations of hidden layers) will be added to the optimization objective (the cost function) so that only a few nodes are encouraged to activate when a single sample is fed into the network. Many choices of the penalty term exist in the literature

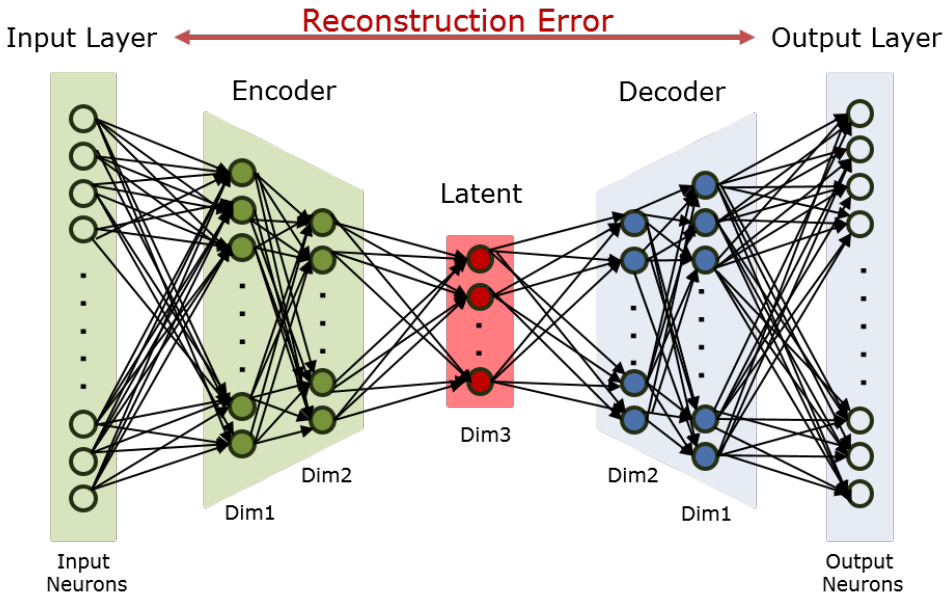

Figure 5.5: Auto-encoder architecture

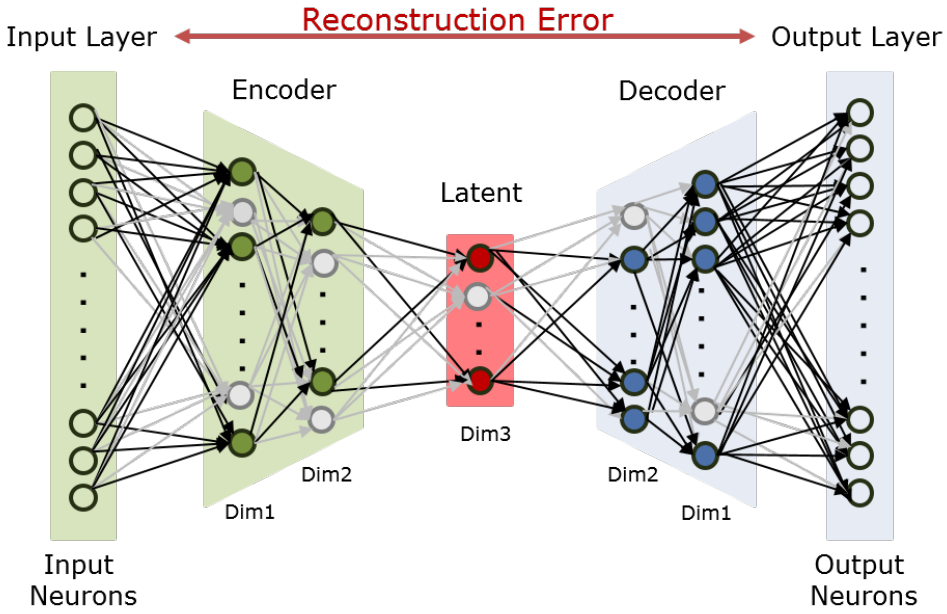

Figure 5.6: Sparse auto-encoder architecture

[JRP+15] such as L1 Regularizer (based on the L1 norm), L2 Regularizer (based on the L2 norm) and the Kullback-Leibler (KL) divergence (measures the difference between two distributions).

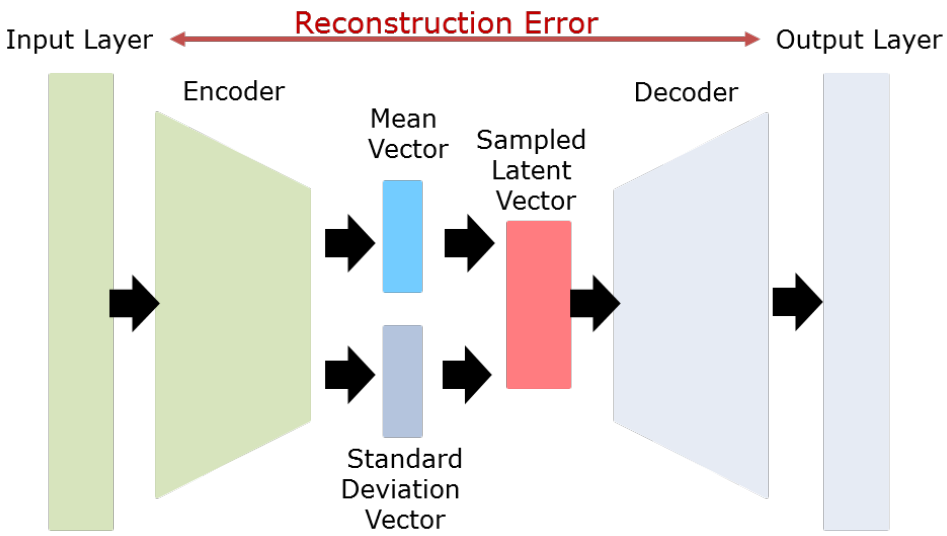

Figure 5.7: Variational auto-encoder architecture

VAE [KW14] inherits the architecture of traditional AEncs but instead of encoding the input into a fixed vector (the bottleneck), it maps it into a distribution (Figure 5.7). Thus, the encoder learns a data generating distribution that allows the decoder to take random samples from the latent space and generate outputs with similar characteristics to the inputs.

5.3.3.2 Threshold Determination

Apart from the training phase, we need to determine the threshold of the RE above which *GFlows* are considered as threats. In other words, the training phase consists in learning how to minimize the RE for the "normal" *GFlows* as much as possible so that when we have an abnormal *GFlow*, its RE will be high. In order to identify the threshold, we use the validation dataset.

Threshold State of the Art According to the literature, the threshold is typically set as the maximum of the RE of the training set [DCS14]. However, experimentally, we remarked that such method is not appropriate for our use-case. Other threshold methods have been proposed in the state of the art, such as the three standard deviations

method in [ERKL16] and the reduced reconstruction error in [KT17] which are both based on statistics of the training data set such as the mean, the standard deviation as well as quartiles. There is also the inlier reconstruction error method based on TP and FP proposed in [KT17] and a density estimation strategy detailed in [CNM16].

Decision Tree based Threshold In our work, we propose a new threshold determination method based on the use of a decision tree algorithm on the validation dataset. Since the latter contains both normal and abnormal inputs, we can compute the RE for both classes. Applying a DT on the results will enable to find the best RE that separates the normal RE values from the anomalous ones.

We start by training our AEncs models on the training dataset trying to minimize the loss function as much as possible. Once our model is ready, we evaluate it on the validation dataset which is composed of the remaining 33% of the normal *GFlows* with the whole threat *GFlows* (as detailed in the beginning of Section 5.3). At the end of this step, we will have a map of the input initial classes (the true class of the entry: normal or anomalous) with the associated RE (the prediction of the AEnc model). As presented in Section 5.2.1.1, a DT chooses the best attribute value to split the set of data samples into subsets with the highest possible information gain (entropy). So when we apply this algorithm on our map (true_class / RE), it will determine the best RE value that can split the validation dataset into normal and anomalous. DT performs the operation recursively until the leaves. However, in our threshold determination process, we will stop at the first data split and retrieve the separation RE value (root of the tree) that we will use as threshold for the test dataset. Figure 5.8 illustrates, as an example, the REs of the validation dataset (for both classes) with the threshold computed by our DT-based solution.

The only disadvantage of this method is that it needs a validation dataset that contains instances from both classes which is the case of our validation dataset (it contains normal and anomalous *GFlows*).

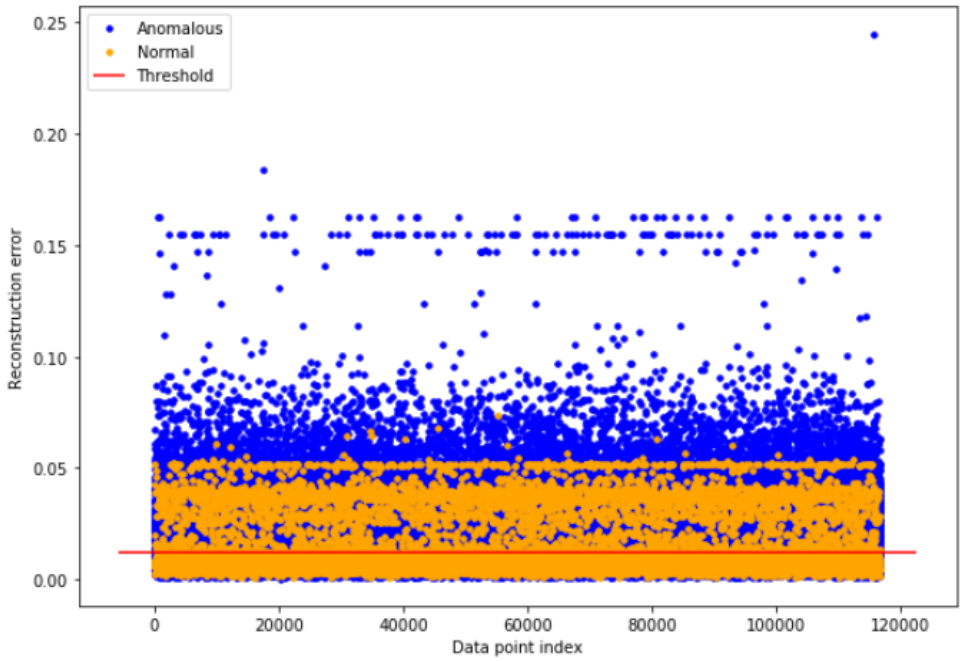

Figure 5.8: Reconstruction errors of the validation dataset with an AEnc

AEncs results with Decision Tree based Threshold To find the best AEncs that enables to detect threats in the oneM2M context, we implement our own randomized search for hyperparameter optimization. We evaluate the AEnc and the SAE with one, three and five hidden layers (with the reflective mirror effect as in Figure 5.5). For the VAE, we experimented one and two layered encoder/decoder with and without sparsity (SVAE). We fix the same hyperparameters for all the networks and search only for the best combinations of the number of neurons (units). These choices have been made after multiple tests: the activation function is *ReLu*, the kernel initalizer is *glorot_uniform*, the loss function is the MSE and the optimizer is *Adam*. For SAE, we use L1 regularizer as activity regularizer and L2 regularizer as kernel regularizer both with $1e-4$ as a regularization factor. For the VAE, we examined also sparse VAE with the same details as SAE and we use the *sigmoid* activation function for the latent layer. The number of the explored combinations varied from 25 (for the one-layered AEncs) to more than 100 combinations for the five-layered ones. We present in Table 5.6 the best results for each type of AEnc and for each number of layers. It is important to mention that in

ML algorithm	Dim1	Dim2	Dim3	Recall(%)	Accuracy(%)	Precision(%)	FPR(%)
AEnc (1 hidden layer)	18			79.34	69.81	80.09	58.38
AEnc (3 hidden layers)	7	3		74.53	72.18	86.38	34.76
AEnc (5 hidden layers)	15	13	7	**85.20**	76.63	83.80	48.74
SAE (1 hidden layer)	11			61.97	66.66	**90.40**	**19.48**
SAE (3 hidden layers)	19	13		62.73	66.82	89.80	21.08
SAE (5 hidden layers)	25	12	6	71.14	71.42	88.35	27.76
SAE (5 hidden layers)	15	10	5	85.13	**77.21**	84.49	46.25
VAE (1 hidden layer)	25	4		65.87	66.90	86.64	30.06
SVAE (1 hidden layer)	13	7		83.19	76.27	84.78	44.21
SVAE (1 hidden layer)	20	4		64.32	67.61	89.36	22.65
VAE (3 hidden layers)	14	5		66.67	67.21	86.35	31.20
SVAE (3 hidden layers)	10	4		71.50	69.86	85.81	35.00

Table 5.6: Best AEncs results with DT-based threshold

the presented results, the sparsity for SAE and for sparse VAE was added only to the first layer. The columns "Dim1", "Dim2" and "Dim3" reflect the number of neurons in the first, second and third layer respectively (Figure 5.5). Our AEncs are all symmetric so for example for the three-hidden-layered AEnc we will have a first input layer with the initial 26 features of a *GFlow*, then the first hidden layer with "Dim1" neurons, the second hidden layer with "Dim2" neurons (the latent layer), the third layer with "Dim1" neurons and finally the output layer with 26 features. For the VAE, "Dim1" is for the first layer and "Dim2" is for the latent layer. For the two layered encoder/decoder VAE, the second layer is composed of "Dim1 / 2" neurons.

As presented in Table 5.6, the best recall is achieved by an AEnc of 5 hidden layers with 85.20% which remains less than J48 binary classifier (about 10% less). This model has a high FPR of 48,74%. It is SAE with 1 hidden layer that has the lowest FPR of 19.48% but its recall is not very high (61.97%). An in-between results model is the 3 hidden layered AEnc and 5 hidden layered SAE. Both of them have acceptable recalls (greater than 71%) and a comparable FPRs (less than 35%) which remain close to the FPR of the binary DNN classifier. Both have also comparable accuracy and precision.

AEncs results with Statistically based Threshold We compare in this paragraph our DT-based threshold to some statistically-based ones. To do that, we retrieve the AEncs models that have achieved more than 60% of recall and less than 50% of FPR in

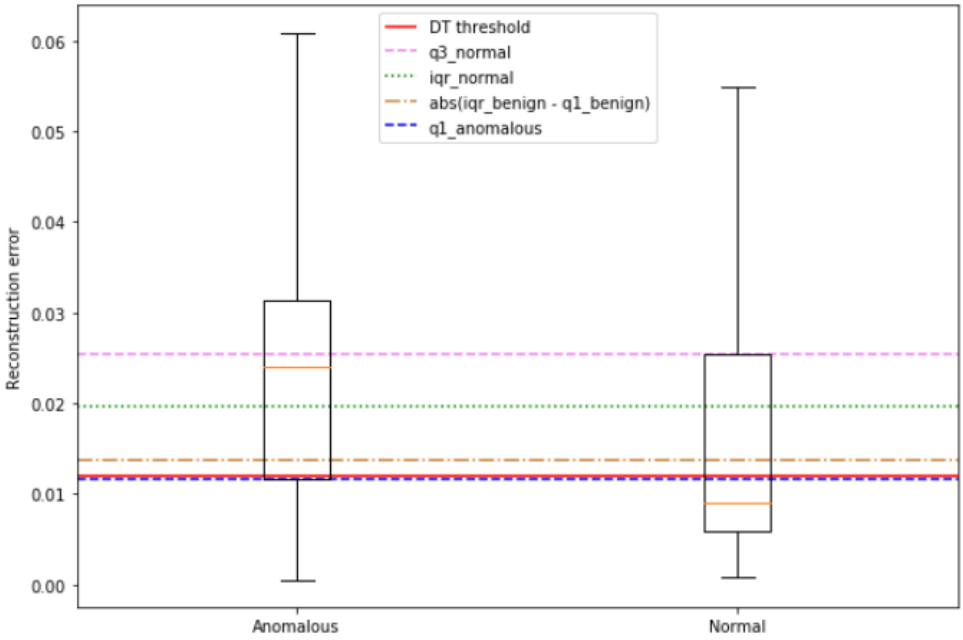

Figure 5.9: Box plot of the reconstruction errors quartiles with thresholds

the DT-based threshold search of the paragraph above. Then, we apply four different statistically-based thresholds on these selected models. Three of them are computed only on the normal entries of the validation dataset which are:

- the upper quartile *q3_normal* which is also called the 75th empirical quartile where 75% of the normal data lies,
- the inter-quartile *iqr_normal* which is the difference between the upper quartile and the lower quartile (where 25% of the normal data lies) of the normal data,
- and the absolute difference between the inter-quartile and the lower quartile *abs(iqr_normal - q1_normal)*.

The fourth statistically-based threshold uses only the anomalous RE of the validation data namely the lower quartile *q1_anomalous*. Figure 5.9 shows the quartiles of the distribution of anomalous and normal entries in the validation data set. It displays also the different experimented thresholds. The presented reconstruction errors are associated to the three-layered AEnc of Table 5.6.

	ML algorithm	Dim1	Dim2	Dim3	Threshold Type	Threshold	Recall(%)	Accuracy(%)	Precision(%)	FPR(%)
1	SAE (5 hidden layers)	19	15	14	q1_anomalous	0.00060799	**75.05**	71.32	84.83	39.73
2	SAE (5 hidden layers)	11	9	3	q1_anomalous	0.01196436	74.86	71.74	85.53	37.48
3	AEnc (3 hidden layers)	7	3		DT	0.01204153	74.53	**72.18**	**86.38**	34.76
4	SAE (5 hidden layers)	19	15	14	iqr_normal	0.00061692	73.89	70.85	85.15	38.14
5	AEnc (3 hidden layers)	7	3		abs(iqr_normal - q1_normal)	0.01317281	73.19	71.69	**86.87**	**32.75**
6	AEnc (5 hidden layers)	19	12	11	iqr_normal	5.054e-05	70.49	68.04	84.18	39.19

Table 5.7: Best AEncs results with the associated thresholds

In Table 5.7, we present the best AEncs results with their associated thresholds. The table is ordered in ascending order of recall and only models that achieve more than 70% of recall and less than 40% of FPR are shown.

The best recall is of 75.05% achieved by the first model with *q1_anomalous* threshold, however its FPR is the worst. The second and third models have almost the same recall of 74% but the third one that uses our DT threshold has the best accuracy, the best precision as well as the second best FPR of about 34% (2% more than the best FPR). The model number 5 which uses *abs(iqr_normal - q1_normal)* threshold has the best FPR. It has also the best precision as well as a comparable recall (2% less than the best recall). Consequently, we can consider that the threshold based on DT and *abs(iqr_normal - q1_normal)* give the best results for our oneM2M use-case with 3 hidden layers AEnc of 7, 3 and 7 neurons for each one. Therefore, in the case where anomalous validation data are available, it is preferable to use DT-based threshold; however, when only normal data are available, *abs(iqr_normal - q1_normal)* seems more appropriate.

5.3.3.3 Final Choice for the First Level of ML Detection

As we can notice in Table 5.8, J48 has the best results in terms of recall, accuracy, precision and FPR but the AEnc has a lighter model and a faster training time. These results are related to the initial test set that we have prepared in Section 5.2.1.3. Since the model size as well as the training time of J48 remain acceptable for IoT context and considering the important difference between the two models especially with 20% less in terms of recall for AEnc, we first choose J48 as our algorithm for the first level of ML

ML algorithm	Recall (%)	Accuracy (%)	Precision (%)	False Positive Rate (%)	Model Size (Ko)	CPU Training Time (ms)
J48	**94.90**	**89.28**	**91.10**	**27.30**	285	30 950
AEnc	74.53	72.18	86.38	34.76	**20**	**22 532**

Table 5.8: Comparison of detection results on the initial test set for the first detection level

detection. This being so, it is important to remember that learning only from normal behaviors enables a better generalization for the unknown and zero-day threats, thus models necessitate less updates. This is also essential, especially when no abnormal data are initially available for the models training (which is the case most of the time in the field of security).

In order to have a better basis for comparison regarding the generalization of the models, we decided to test both of them with new attack data that we had never seen before. This test is crucial for our final choice. Therefore, we recorded new oneM2M *GFlows* data where we changed the frequency of sending GPS data of a position sensor in our system. Changing the sending frequency (two times less) is considered abnormal behaviour. To do that, we collected 11 349 new *GFlows* which are labeled as threats. J48 was not able to detect this zero-day threat (0.1% of recall), however, AEnc achieves 97%. Since the generalization of the model for the detection of zero-day threats is really crucial, our final choice for the first level of detection will be the use of the AEnc model.

One possible way to improve AEnc results (regarding the test set) is to choose a higher threshold value, which will unfortunately lead to an increase in FPR (which will bother the administrator with false alarms) but could at least be a guarantee that we will not miss any threats. This increase in FPR could be compensated by the continuous learning module after a period of time.

5.3.3.4 One-Class Approach for Multi-Classification

This section concentrates on the second level of ML detection of our IDS (Section 4.2.2). In order to classify a threat as one of the already known threats or as an unknown one

as presented in Figure 4.5, we need a model capable of:

- classifying the threats into the different known classes (flooding, amplification, etc.),
- detecting new threats (novelties) that are not available during the training stage.

To do this, we have opted to adapt the OCC approach to the multi-classification task, which is called multi-class novelty detection [JS14].

Techniques Description Different studies have been proposed for multi-class novelty detection but they are mostly based on support vectors algorithms [LTVNF16] or on clustering based techniques [DLBM14]. In our work, we propose to adapt the AEnc of the one-class classification approach to the multi-classification issue. We have considered two solutions. The first one use two different models: the AEnc with a decision tree model, however the second one is based mainly on the AEnc.

- AEnc + DT: we start with an AEnc that classifies a *GFlow* as known or unknown (novelty). If it is known then it will be analyzed with the DT model to get the exact threat family (Figure 5.10 (left side)). We chose DT since J48 had the best results in Section 5.2.1.4. The AEnc will be trained on the known threats as one class. So in the prediction phase, if the reconstruction error of the *GFlow* is greater than the fixed threshold, it is considered as a novelty. Regarding the DT model, it is trained only on the known threats (flooding and amplification at the starting point like in Section 5.2.1.4).
- AEnc: same as above, we begin with novelty detection using the same AEnc. However, if it is a known *GFlow*, the exact threat family will be determined thanks to a deep neural network composed of the same encoder function of the initial AEnc followed by fully connected layers as shown in Figure 5.10 (right side). This DNN loaded the weights of the AEnc encoder and trained only the new fully connected layers, making the training phase light.

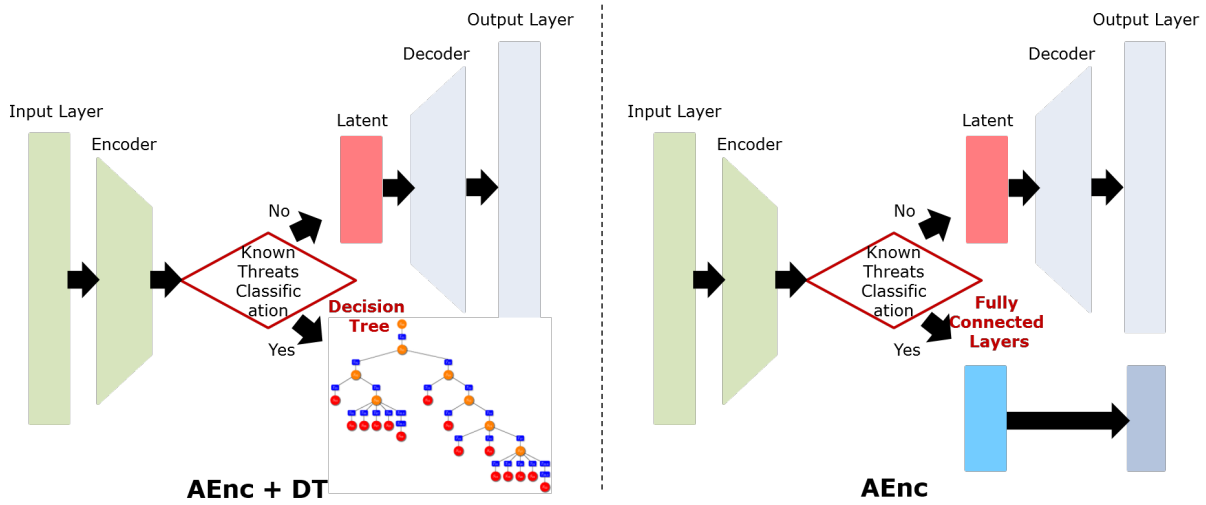

Figure 5.10: Multi-classification with AEnc

Experiments and Results Two use cases were considered regarding the experiments for the second level of ML:

1. We consider that we know only the flooding attack (at the starting point) and we try to detect the amplification threat (as unknown) with an AEnc.

2. We consider that thanks to the continuous learning module of Section 4.2.4, our model knows at this second stage both flooding and amplification, and needs to be able to detect a new unknown threat that we have never seen before. For this use case, we use the recorded oneM2M *GFlows* data where we changed the frequency of sending GPS data (introduced in Section 5.3.3.3). Changing the sending frequency (two times less) is considered abnormal.

We start this section by finding the best AEnc model for the unknown threats detection (since it is common for both multi-class novelty detection techniques as well as for both use cases) then we compare the two different techniques for the multi-classification step.

For the first use case, we need to detect amplification threats with an AEnc trained only on flooding threats. The best results have been achieved by a three-layered AEnc with 20 neurons for the hidden layer. Regarding the threshold, we assumed that we

ML algorithm	Recall (%)	Accuracy (%)	Precision (%)	False Positive Rate (%)	Model Size (Ko)	CPU Training Time (ms)
DT-based	**99.93**	**99.93**	**99.95**	**0.087**	20	**0.462026**
Encoder-based	99.29	99.34	99.65	0.59	**19**	470 047

Table 5.9: Final comparison of ML techniques for the second detection level

have both examples of flooding and amplification at the validation step. So if the RE is below the threshold, the *GFlow* is a threat of flooding. If it is above, it is classified as amplification. Our DT-based threshold method gave the best results: 98.45% of recall, 98.91% of accuracy, 99.83% of precision and 0.29% of FPR with a threshold value of 0.0046.

For the second use case, we have to use the same AEnc network (three-layered AEnc with 20 neurons for the hidden layer) as in the first use case (since it needs to be fixed for the continuous learning module). This time, AEnc is trained on flooding and amplification threats and must be able to detect changes in sending frequency as an unknown threat. Regarding the threshold, we consider that we do not have data of the new unknown threat at the validation step, hence we can not use DT-based threshold. We tested the different threshold methods presented in Section 5.3.3.2. Unfortunately, we obtained unpromising results. However, good results (94.86% of recall, 95.72% of accuracy, 99.99% of precision and 0.04% of FPR) were performed with a threshold set as the final validation loss value (0.0015) that we had at the training stage.

After experimenting the unknown threat detection and finding the best model to do that, we compare at this stage the two strategies for known threats classification. Both strategies with a decision tree and using the AEnc encoder network have been implemented and tested in python. For the encoder-based model, we added just one fully connected layer of two neurons with *softmax* activation function. We trained this final layer for 1000 epochs with 400 as a *batch_size*, *categorical_crossentropy* as a loss function and *Adam* as an optimizer. As we can notice from Table 5.9, the results are close in terms of recall, accuracy, precision, FPR and model size. However, the DT model is much faster than a fully connected encoder-based network.

Conclusion

This chapter was about the experimentation of machine learning and deep learning algorithms for the different layers of our intrusion detection system. We started by explaining our choice to adopt learning techniques and we presented the used metrics as well as the experimental environment. Moreover, we described the different algorithms experimented for each layer of our IDS and we presented their detection results. Furthermore, we focused in the last section on the one-class classification approach that we have experimented for anomaly detection in the first level of our IDS. In addition, we have proposed, compared and discussed our DT-based threshold method for the OCC approach. And finally, we adapted it (OCC) to enable multi-classification as well as novelty detection in the context of unknown threats.

Chapter 6

Conclusion and Perspectives

In recent years, the spread of IoT devices throughout the world has been advancing rapidly. The connected devices are now deployed in all areas such as healthcare, smart-cities, education, etc. To integrate this rapid commercialization flow, little attention has been paid to the safety and security of IoT devices and networks which endangers IoT users and in turn disrupts the entire Internet-connected ecosystem including web-sites, applications, social networks and servers. In addition, security attack vectors have evolved bothways, in terms of complexity and diversity. Therefore, more attention needs to be paid to the analysis of these attacks, their detection, as well as to infection prevention and system recovery after attacks.

In this thesis, we have studied and proposed an Intrusion Detection and Prevention System (IDPS) based on Machine Learning (ML) for the IoT ecosystem in order to immediately detect and respond to potential threats as soon as they occur. We focused our security framework towards the international oneM2M standard which enables the communication between heterogeneous devices and applications by defining a common M2M Service Layer for the multi-industry M2M applications. To the best of our knowledge, our proposal is the first IDPS for the oneM2M service layer. It represents a consistent framework with a complete security workflow, from data collection to

threat detection and activation of appropriate actions. It also has a continuous learning module that provides an up-to-date IDPS, evolving with the evolution and emergence of new threats. Our proposal is also characterized by its interoperability, autonomy, scalability and respect for resource constraints, modularity in design, adaptability and extensibility, as well as active and real-time response.

The thesis was organized in six chapters. In Chapter 1, we presented the IoT security ecosystem and categorized its threats as well as its traditional defense mechanisms. We had also detailed our motivation and the different paper contributions made during the PhD. In Chapter 2, we detailed and discussed the literature proposals dealing with the network IDPS in IoT systems, based or not on ML techniques. We compared their strategies, their architectures as well as their results and many other characteristics. In Chapter 3, we gave an overview of the oneM2M standard and its security mechanisms. Then, we focused on oneM2M threats related to service availability, for which we proposed a taxonomy and various implementations. We finished this chapter with details about the creation of our oneM2M security dataset which is based on a new abstraction for the oneM2M flows. In Chapter 4, we exposed the different challenges and aims respected and guaranteed with our oneM2M-IDPS proposal. Furthermore, we detailed its strategy as well as the architecture and the design of each of its four modules: i) the data acquisition and features extraction module, ii) the IDS module, iii) the IPS module and iv) the continuous learning module. In Chapter 5, we concentrated on the ML aspect of our IDPS. We implemented our detection module with its three ML levels using the oneM2M dataset. We experimented with different ML and Deep Learning (DL) algorithms for each detection level. We studied two main approaches: i) detection based on supervised ML algorithms and ii) detection based on one-class classification (OCC) approach.

In the rest of this chapter, we will detail the main contributions of this study. We will then discuss their limitations and propose some directions for the future.

6.1 Contributions of Research

The key contributions of this research are as follows.

- The review of the literature proposals dealing with the network IDPS in IoT systems, with a special focus on those based on ML techniques: we provided a brief description for each work then we discussed and compared them on the basis of the architectures, detection methodologies, validation strategies, treated threats as well as the used ML algorithms and datasets.

- The proposal of an abstraction (*GFlow*) for the oneM2M messages: we proposed this abstraction by aggregating oneM2M flows (request/response) on a dynamic basis (the number of flows to be combined as a parameter). This aggregation avoids the resources consumption, which is important in the context of IoT. It also respects the anonymity of the IoT devices' identities.

- The design of a oneM2M dataset: with respect to the proposed threats taxonomy, we created a oneM2M dataset with a real IoT system. This dataset is based on the *GFlow* abstraction. It is labeled and composed of benign inputs as well as real oneM2M threats.

- The design of a oneM2M-IDPS: it is a complete security framework for the oneM2M standard composed of four main modules: i) the data acquisition and features extraction module where the oneM2M messages are treated and the *GFlows* are generated and prepared to be injected to the IDS module, ii) the IDS module to detect the threats, iii) the IPS to prevent the threats from damaging the system and iv) the continuous learning module to guarantee the update of the IDPS against the new threats. The last two modules have been neglected by most state of the art works. OneM2M-IDPS is characterized by its interoperability, autonomy, scalability and respect for resource constraints, modularity in design, adaptability and extensibility, as well as active and real-time response.

- The design and testing of a three-level Intrusion Detection module (IDS): it guarantees a light detection by eliminating the in-depth analysis of normal *GFlows* as soon as possible, which respects the resource constraints of the IoT devices. It avoids overloading and unnecessary consumption of resources.

- The proposal and experimentation of two different detection strategies for the IDS module: we studied i) detection based on supervised ML algorithms and ii) detection based on one-class classification approach. For each strategy, different ML algorithms were implemented, tested and compared.

- The proposal and testing of a new threshold determination method for anomaly detection based on the decision tree (DT) algorithm: it is based on the validation dataset which contains both normal and abnormal *GFlows*.

- The adaptation of one-class classification for a multi-class novelty detection: in the second level of IDS, we use the OCC approach to enable the detection of unknown threats as well as the multi-classification of already known threats.

These contributions were accompanied throughout the PhD by the implementation and deployment of our oneM2M-IDPS proposal on the IoT system that we have in the Atos Innovation Aquitaine Lab.

6.2 Limitations and Future directions

Obviously, improving the efficiency of machine learning detection results remains an open research topic. The IoT security community must always strive for 100% detection with zero false alarms while respecting IoT constraints. In this section, we will outline some limitations of our contributions and propose future directions to address them.

- The created oneM2M dataset contains only few service availability threats. It would be interesting to incorporate new varieties of threats, whether related to

the service availability or to another type. This will allow better training for the models on the one hand, and better testing for the models chosen for the three levels of ML detection on the other hand. Another interesting direction regarding the oneM2M dataset is to extend it with the values sent by the connected devices (payload). This would be a difficult task because each device has its own data type or data structure. It will therefore be challenging to have a common format or to find an adaptation of the ML algorithms so that they can take into account the different heterogeneous formats.

- Our IDPS is only proposed for the oneM2M service layer, which mainly allows only oneM2M-related threats to be handled. First, it would be curious to test if with the current framework it is possible to detect threats other than those related to the oneM2M standard. Second, it would be interesting to extend our oneM2M-IDPS to take into consideration the different layers of the IoT stack such as the network layer.

- OneM2M-IDPS deals not only with security attacks but also with abnormal behaviors. However, the notion of abnormal behavior is related to device types or so-called device profiles. Therefore, the design of an identification tool capable of autonomously building and detecting device profiles will be a promising enhancement for our IDPS.

- Regarding the learning process as well as model updates, the proposed oneM2M-IDPS builds the models offline from a database that is frozen at the time of training and then replaces the old models completely with the new ones. An important line of research for our proposal would be to enhance our continuous learning module with an incremental online learning mechanism to enable real-time learning. In other words, the models will be updated dynamically each time a new data is available.

- The deployment of the proposed oneM2M-IDPS in large-scale applications with

a federated learning strategy [KMY+17], which is a communication-efficient and privacy-preserving approach, also appears to be a challenging research topic. This will first of all check the scalability of our proposal. In addition, the federated learning approach will enable distributed model learning from multiple clients (without sending data to a remote server or sharing local training data). Each client trains a local model and sends only updates to a centralized entity for aggregation. This approach highlights edge and fog computing that we have adopted in our proposal.

Bibliography

[AA15] S. Agrawal and J. Agrawal. Survey on Anomaly Detection using Data Mining Techniques. *Procedia Computer Science*, 60:708–713, January 2015.

[AAD10] A. O. Adetunmbi, S. O. Adeola, and O. A. Daramola. Analysis of KDD '99 Intrusion Detection Dataset for Selection of Relevance Features. In *Proceedings of the World Congress on Engineering and Computer Science*, volume 1, San Francisco, USA, October 2010.

[AAT18] S. A. Alabady and F. Al-Turjman. Low Complexity Parity Check Code for Futuristic Wireless Networks Applications. *IEEE Access*, 6:18398–18407, April 2018.

[AATD18] S. A. Alabady, F. Al-Turjman, and S. Din. A Novel Security Model for Cooperative Virtual Networks in the IoT Era. *International Journal of Parallel Programming*, July 2018.

[AC15] J. An and S. Cho. Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE, 2(1)*, 2015.

[ADMC17] M. R. Asghar, G. Dán, D. Miorandi, and I. Chlamtac. Smart Meter Data Privacy: A Survey. *IEEE Communications Surveys Tutorials*, 19(4):2820–2835, 2017.

[AFGM+15] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys Tutorials*, 17(4):2347–2376, 2015.

[AIZ+17] S. Anwar, Z. Inayat, M. F. Zolkipli, J. M. Zain, A. Gani, Nor Badrul Anuar, Muhammad Khurram Khan, and Victor Chang. Cross-VM cache-based side channel attacks and proposed prevention mechanisms: A survey. *Journal of Network and Computer Applications*, 93:259–279, September 2017.

[AK16] M. E. Aminantoa and K. Kimb. Deep Learning in Intrusion Detection System : An Overview. In *International Research Conference on Engineering and Technology (2016 IRCET). Higher Education Forum, 2016.*, 2016.

[Ald19] A. Aldaej. Enhancing Cyber Security in Modern Internet of things (IoT) Using Intrusion Prevention Algorithm for IoT (IPAI). *IEEE Access*, pages 1–1, 2019. Conference Name: IEEE Access.

[All] The OSGi Alliance. OSGi™ Alliance – The Dynamic Module System for Java (https://www.osgi.org/).

[AM14] A. W. Atamli and A. Martin. Threat-Based Security Analysis for the Internet of Things. In *2014 International Workshop on Secure Internet of Things*, pages 35–43, September 2014.

[Ana] CAIDA: Center for Applied Internet Data Analysis. CAIDA Data - Overview of Datasets, Monitors, and Reports (https://www.caida.org/data/overview/index.xml).

[ANMH16] M. Ahmed, A. Naser Mahmood, and J. Hu. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, 60:19–31, January 2016.

[AO17] A. Aris and S. F. Oktug. Poster: State of the Art IDS Design for IoT. In *Proceedings of the 2017 International Conference on Embedded Wireless Systems and Networks*, EWSN '17, pages 196–197, USA, February 2017. Junction Publishing.

[AOF+10] N. Araújo, R. de Oliveira, E. Ferreira, A. A. Shinoda, and B. Bhargava. Identifying important characteristics in the KDD99 intrusion detection dataset by feature selection using a hybrid approach. In *2010 17th International Conference on Telecommunications*, pages 552–558, April 2010.

[AP12] P. Asrodia and H. Patel. Analysis of various packet sniffing tools for network monitoring and analysis. *International Journal of Electrical, Electronics and Computer Engineering*, 1(1):55–58, 2012.

[AT18] F. Al-Turjman. QoS—aware data delivery framework for safety-inspired multimedia in integrated vehicular-IoT. *Computer Communications*, 121:33–43, May 2018.

[ATA18a] F. Al-Turjman and S. Alturjman. Confidential smart-sensing framework in the IoT era. *The Journal of Supercomputing*, 74(10):5187–5198, October 2018.

[ATA18b] F. Al-Turjman and S. Alturjman. Context-Sensitive Access in Industrial Internet of Things (IIoT) Healthcare Applications. *IEEE Transactions on Industrial Informatics*, 14(6):2736–2744, June 2018.

[ATEE+17] F. Al-Turjman, Y. K. Ever, E. Ever, H. X. Nguyen, and D. B. David. Seamless Key Agreement Framework for Mobile-Sink in IoT Based Cloud-Centric Secured Public Safety Sensor Networks. *IEEE Access*, 5:24617–24631, October 2017.

[BBJJ+17] S. Bhandari, W. Ben Jaballah, V. Jain, V. Laxmi, A. Zemmari, Manoj Singh Gaur, Mohamed Mosbah, and Mauro Conti. Android inter-

app communication threats and detection techniques. *Computers & Security*, 70:392–421, September 2017.

[BC06] Y. Bouzida and F. Cuppens. Detecting Known and Novel Network Intrusions. In S. Fischer-Hübner, K. Rannenberg, L. Yngström, and S. Lindskog, editors, *Security and Privacy in Dynamic Environments*, IFIP International Federation for Information Processing, pages 258–270, Boston, MA, 2006. Springer US.

[BG16] A. L. Buczak and E. Guven. A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. *IEEE Communications Surveys Tutorials*, 18(2):1153–1176, 2016.

[BHDA14] E. Bou-Harb, M. Debbabi, and C. Assi. Cyber Scanning: A Comprehensive Survey. *IEEE Communications Surveys Tutorials*, 16(3):1496–1519, 2014.

[BHL+17] S. Bhandari, F Herbreteau, V. Laxmi, A. Zemmari, P. S. Roop, and M. S. Gaur. Detecting Inter-App Information Leakage Paths. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '17, pages 908–910, Abu Dhabi, United Arab Emirates, April 2017. Association for Computing Machinery.

[Bis94] C. M. Bishop. Novelty detection and neural network validation. *IEE Proceedings - Vision, Image and Signal Processing*, 141(4):217–222, August 1994. Publisher: IET Digital Library.

[BJCF+18] W. Ben Jaballah, M. Conti, G. Filè, M. Mosbah, and A. Zemmari. Whac-A-Mole: Smart node positioning in clone attack in wireless sensor networks. *Computer Communications*, 119:66–82, April 2018.

[BJMYZ13] W. Ben Jaballah, M. Mosbah, H. Youssef, and A. Zemmari. Lightweight Source Authentication Mechanisms for Group Communications in Wire-

less Sensor Networks. In *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, pages 598–605, March 2013. ISSN: 1550-445X.

[BJMYZ15] W. Ben Jaballah, M. Mosbah, H. Youssef, and A. Zemmari. Lightweight secure group communications for resource constrained devices. *International Journal of Space-Based and Situated Computing*, 5(4):187–200, January 2015. Publisher: Inderscience Publishers.

[BL78] V. Barnett and T. Lewis. Outliers in statistical data. In *Wiley series in probability and mathematical statistics*. John Wiley & Sons Ltd., 2nd edition., 1978.

[BPS+02] A. Bivens, C. Palagiri, R. Smith, B. Szymanski, and M. Embrechts. NETWORK-BASED INTRUSION DETECTION USING NEURAL NETWORKS. page 6, 2002.

[BS17] H. Bostani and M. Sheikhan. Hybrid of anomaly-based and specification-based IDS for Internet of Things using unsupervised OPF based on MapReduce approach. *Computer Communications*, 98(Supplement C):52–71, January 2017.

[BSP+11] S. Babar, A. Stango, N. Prasad, J. Sen, and R. Prasad. Proposed embedded security framework for Internet of Things (IoT). In *2011 2nd International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology (Wireless VITAE)*, pages 1–5, February 2011.

[BTC15] S. S. Basu, S. Tripathy, and A. R. Chowdhury. Design challenges and security issues in the Internet of Things. In *2015 IEEE Region 10 Symposium*, pages 90–93, May 2015.

[BWH18] E. Benkhelifa, T. Welsh, and W. Hamouda. A Critical Review of Practices and Challenges in Intrusion Detection Systems for IoT: Towards Universal and Resilient Systems. *IEEE Communications Surveys Tutorials*, pages 1–1, June 2018.

[CBO17] R. E. Crossler, F. Bélanger, and D. Ormond. The quest for complete security: An empirical analysis of users' multi-layered protection from security threats. *Information Systems Frontiers*, April 2017.

[CDL16] M. Conti, N. Dragoni, and V. Lesyk. A Survey of Man In The Middle Attacks. *IEEE Communications Surveys Tutorials*, 18(3):2027–2051, 2016.

[Cis12] Cisco. A Cisco Guide to Defending Against Distributed Denial of Service Attacks, October 2012.

[CMO12] P. Casas, J. Mazel, and P. Owezarski. Unsupervised Network Intrusion Detection Systems: Detecting the Unknown without Knowledge. *Computer Communications*, 35(7):772–783, April 2012.

[CMZ+19] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki. Network Intrusion Detection for IoT Security based on Learning Techniques. *IEEE Communications Surveys Tutorials*, 2019.

[CMZS19] N. Chaabouni, M. Mosbah, A. Zemmari, and C. Sauvignac. An Intrusion Detection System for the OneM2M Service Layer based on Edge Machine Learning. *18th International Conference on Ad Hoc Networks and Wireless, AdHoc-Now 2019 Luxembourg*, pages 508–523, October 2019.

[CMZS20] N. Chaabouni, M. Mosbah, A. Zemmari, and C. Sauvignac. A OneM2M Intrusion Detection and Prevention System based on Edge Machine Learning. *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, April 2020.

[CN12] N.S. Chandolikar and V.D. Nandavadekar. Selection of Relevant Feature for Intrusion Attack Classification by Analyzing KDD Cup 99. *MIT International Journal of Computer Science & Information Technology*, 2(2):85–90, August 2012.

[CNM16] V. L. Cao, M. Nicolau, and J. McDermott. A Hybrid Autoencoder and Density Estimation Model for Anomaly Detection. In J. Handl, E. Hart, P. R. Lewis, M. López-Ibáñez, G. Ochoa, and B. Paechter, editors, *Parallel Problem Solving from Nature – PPSN XIV*, Lecture Notes in Computer Science, pages 717–726, Cham, 2016. Springer International Publishing.

[CPNS15] C. Cervantes, D. Poplade, M. Nogueira, and A. Santos. Detection of sinkhole attacks for supporting secure routing on 6lowpan for Internet of Things. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 606–611, May 2015.

[DA01] P. Dayan and L. F. Abbott. Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems. 2001.

[dar] 1999 DARPA Intrusion Detection Evaluation Dataset | MIT Lincoln Laboratory (https://www.ll.mit.edu/r-d/datasets/1999-darpa-intrusion-detection-evaluation-dataset).

[DB15] S. K. Datta and C. Bonnet. A lightweight framework for efficient M2M device management in oneM2M architecture. In *2015 International Conference on Recent Advances in Internet of Things (RIoT)*, pages 1–6, April 2015. ISSN: null.

[DC17] A. A. Diro and N. Chilamkurti. Distributed attack detection scheme using deep learning approach for Internet of Things. *Future Generation Computer Systems*, September 2017.

[DCS14] H. A. Dau, V. Ciesielski, and A. Song. Anomaly Detection Using Replicator Neural Networks Trained on Examples of One Class. In G. Dick, W. N. Browne, P. Whigham, M. Zhang, L. T. Bui, H. Ishibuchi, Y. Jin, X. Li, Y. Shi, P. Singh, K. C. Tan, and K. Tang, editors, *Simulated Evolution and Learning*, Lecture Notes in Computer Science, pages 311–322, Cham, 2014. Springer International Publishing.

[def] DEF CON® Hacking Conference - Capture the Flag Archive (https://www.defcon.org/html/links/dc-ctf.html).

[Dem08] Arthur P. Dempster. Upper and Lower Probabilities Induced by a Multivalued Mapping. In Roland R. Yager and L. Liu, editors, *Classic Works of the Dempster-Shafer Theory of Belief Functions*, Studies in Fuzziness and Soft Computing, pages 57–72. Springer, Berlin, Heidelberg, 2008.

[DLBM14] X. Ding, Y. Li, A. Belatreche, and L. P. Maguire. An experimental evaluation of novelty detection methods. *Neurocomputing*, 135:313–327, July 2014.

[DLY+18] L. Deng, D. Li, X. Yao, D. Cox, and H. Wang. Mobile network intrusion detection for IoT system based on transfer learning algorithm. *Cluster Computing*, January 2018.

[DM04] C. Douligeris and A. Mitrokotsa. DDoS attacks and defense mechanisms: classification and state-of-the-art. *Computer Networks*, 44(5):643–666, April 2004.

[EFGK03] P. T. Eugster, P. A. Felber, R. Guerraoui, and A. M. Kermarrec. The many faces of publish/subscribe. *ACM computing surveys (CSUR) 35*, pages 114–131, 2003.

[ERKL16] S. M. Erfani, S. Rajasegarar, S. Karunasekera, and C. Leckie. High-dimensional and large-scale anomaly detection using a linear one-class

SVM with deep learning. *Pattern Recognition*, 58:121–134, October 2016.

[FBF77] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An Algorithm for Finding Best Matches in Logarithmic Expected Time. *ACM Trans. Math. Softw.*, 3(3):209–226, September 1977.

[FBL+15] P. Faruki, A. Bharmal, V. Laxmi, V. Ganmoor, M. S. Gaur, M. Conti, and M. Rajarajan. Android Security: A Survey of Issues, Malware Penetration, and Defenses. *IEEE Communications Surveys Tutorials*, 17(2):998–1022, 2015.

[fCC17] Canadian Institute for Cybersecurity (CIC). IDS 2017 | Datasets | Research | Canadian Institute for Cybersecurity | UNB, 2017.

[FEE+18] G. Francia, L. Ertaul, L. H. Encinas, E. El-Sheikh, and K. Daimi. *Computer and Network Security Essentials*. Springer Publishing Company, Incorporated, 2018.

[FK05] F. Fuentes and D. C. Kar. Ethereal vs. Tcpdump: A Comparative Study on Packet Sniffing Tools for Educational Purpose. *J. Comput. Sci. Coll.*, 20(4):169–176, April 2005.

[FK14] Y. Fu and O. Koné. Security and Robustness by Protocol Testing. *IEEE Systems Journal*, 8(3):699–707, September 2014.

[FMA+18] M. A. Ferrag, L. Maglaras, A. Argyriou, D. Kosmanos, and H. Janicke. Security for 4G and 5G cellular networks: A survey of existing authentication and privacy-preserving schemes. *Journal of Network and Computer Applications*, 101:55–82, January 2018.

[Fou] The Apache Software Foundation. ActiveMQ (https://activemq.apache.org/how-does-a-queue-compare-to-a-topic).

[Fou17] The Open Information Security Foundation. Suricata (https://suricata-ids.org/), 2017.

[FTM+17] Z. M. Fadlullah, F. Tang, B. Mao, N. Kato, O. Akashi, T. Inoue, and K. Mizutani. State-of-the-Art Deep Learning: Evolving Machine Intelligence Toward Tomorrow's Intelligent Network Traffic Control Systems. *IEEE Communications Surveys Tutorials*, 19(4):2432–2455, 2017.

[FYC+17] Y. Fu, Z. Yan, J. Cao, O. Koné, and X. Cao. An Automata Based Intrusion Detection Method for Internet of Things, May 2017.

[FZHH14] W. Feng, Q. Zhang, G. Hu, and J. X. Huang. Mining network data for intrusion detection through combining SVMs with ant colony networks. *Future Generation Computer Systems*, 37:127–140, July 2014.

[GAL+20] J. Gajrani, U. Agarwal, V. Laxmi, B. Bezawada, M. S. Gaur, M. Tripathi, and A. Zemmari. EspyDroid+: Precise reflection analysis of android apps. *Computers & Security*, 90:101688, March 2020.

[GBBK12] P. Gogoi, M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita. Packet and Flow Based Network Intrusion Dataset. In *Contemporary Computing*, Communications in Computer and Information Science, pages 322–334. Springer, Berlin, Heidelberg, August 2012.

[GBMP13] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645–1660, September 2013.

[GLT+18] J. Gajrani, V. Laxmi, M. Tripathi, M. S. Gaur, D. R. Sharma, A. Zemmari, M. Mosbah, and M. Conti. Unraveling Reflection Induced Sensitive Leaks in Android Apps. In N. Cuppens, F. Cuppens, J. L. Lanet, A. Legay, and J. Garcia-Alfaro, editors, *Risks and Security of Internet and Systems*, Lec-

ture Notes in Computer Science, pages 49–65, Cham, 2018. Springer International Publishing.

[GMS15] J. Granjal, E. Monteiro, and J. Sá Silva. Security for the Internet of Things: A Survey of Existing Protocols and Open Research Issues. *IEEE Communications Surveys Tutorials*, 17(3):1294–1312, 2015.

[GP18] Jorge Granjal and Artur Pedroso. An Intrusion Detection and Prevention Framework for Internet-Integrated CoAP WSN, 2018. ISSN: 1939-0114 Library Catalog: www.hindawi.com Pages: e1753897 Publisher: Hindawi Volume: 2018.

[GR14] S. Game and C. Raut. Protocols for detection of node replication attack on wireless sensor network. *Journal of Computer Engineering*, 16(1):01–11, January 2014.

[Gra14] A. Grau. The Internet of Secure Things – What is Really Needed to Secure the Internet of Things? | Icon Labs, March 2014.

[GSLG16] A. Gharib, I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani. An Evaluation Framework for Intrusion Detection Dataset. In *2016 International Conference on Information Science and Security (ICISS)*, pages 1–6, December 2016.

[HAFP14] F. Hosseinpour, P. V. Amoli, F. Farahnakian, and J. Plosila. Artificial Immune System Based Intrusion Detection : Innate Immunity using an Unsupervised Learning Approach. *JDCTA Int. J. Digit. Content Technol. its Appl.*, 8(5):1–12, October 2014.

[HB99] S Hettich and S.D. Bay. KDD Cup 1999 Data - The UCI KDD Archive. Irvine, CA: University of California, Department of Information and Computer Science., 1999.

[HBB+14] N. Hoque, M. H. Bhuyan, R. C. Baishya, D. K. Bhattacharyya, and J. K. Kalita. Network attacks: Taxonomy, tools and systems. *Journal of Network and Computer Applications*, 40:307–324, April 2014.

[HBH+16] E. Hodo, X. Bellekens, A. Hamilton, P. L. Dubouilh, E. Iorkyase, C. Tachtatzis, and R. Atkinson. Threat analysis of IoT networks using artificial neural network intrusion detection system. In *2016 International Symposium on Networks, Computers and Communications (ISNCC)*, pages 1–6, May 2016.

[HBH+17] E. Hodo, X. Bellekens, A. Hamilton, C. Tachtatzis, and R. Atkinson. Shallow and Deep Networks Intrusion Detection System: A Taxonomy and Survey. *arXiv:1701.02145 [cs]*, January 2017. arXiv: 1701.02145.

[Hen16] K. Hengst. DDoS through the Internet of Things An analysis determining the potential power of a DDoS attack using IoT devices, July 2016.

[HFH+09] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA Data Mining Software: An Update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.

[HFH15] M. M. Hossain, M. Fotouhi, and R. Hasan. Towards an Analysis of Security Issues, Challenges, and Open Problems in the Internet of Things. In *2015 IEEE World Congress on Services*, pages 21–28, June 2015. ISSN: 2378-3818.

[HHS+17] W. Haider, J. Hu, J. Slay, B. P. Turnbull, and Y. Xie. Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling. *Journal of Network and Computer Applications*, 87:185–192, June 2017.

[HLDGMG17] Arash Habibi Lashkari, Gerard Draper Gil, Mohammad Saiful Islam Mamun, and Ali A. Ghorbani. Characterization of Tor Traffic using Time

based Features:. In *Proceedings of the 3rd International Conference on Information Systems Security and Privacy*, pages 253–262, Porto, Portugal, 2017. SCITEPRESS - Science and Technology Publications.

[HMR+13] F. Hosseinpour, A. Meulenberg, S. Ramadass, P. V. Vahdani Amoli, and Z. Moghaddasi. Distributed Agent Based Model for Intrusion Detection System Based on Artificial Immune System. *JDCTA Int. J. Digit. Content Technol. its Appl*, 7:206–214, May 2013.

[Hol16] D. Holmes. What's the Fix for IoT DDoS Attacks? | SecurityWeek.Com, October 2016.

[HVAP+16] F. Hosseinpour, P. Vahdani Amoli, J. Plosila, T. Hämäläinen, and H. Tenhunen. An Intrusion Detection System for Fog Computing and IoT based Logistic Systems using a Smart Data Approach. *International Journal of Digital Content Technology and its Applications*, 10, December 2016.

[HZS06] G. B. Huang, Q. Y. Zhu, and C. K. Siew. Extreme learning machine: Theory and applications. *Neurocomputing*, 70(1):489–501, December 2006.

[IDC15] United Nations IDC, Intel. A Guide to the Internet of Things Infographic, February 2015.

[IJL16] I. B. Ida, A. Jemai, and A. Loukil. A survey on security of IoT in the context of eHealth and clouds. In *2016 11th International Design Test Symposium (IDT)*, pages 25–30, Hammamet, Tunisia, December 2016.

[IKMM+19] P. Illy, G. Kaddoum, C. Miranda Moreira, K. Kaur, and S. Garg. Securing Fog-to-Things Environment Using Intrusion Detection System Based On Ensemble Learning. In *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–7, April 2019. ISSN: 1525-3511.

[isc] Datasets | Research | Canadian Institute for Cybersecurity | UNB (https://www.unb.ca/cic/datasets/index.html).

[JC14] C. Jun and C. Chi. Design of Complex Event-Processing IDS in Internet of Things. In *2014 Sixth International Conference on Measuring Technology and Mechatronics Automation*, pages 226–229, January 2014.

[JMG95] N. Japkowicz, C. Myers, and M. Gluck. A novelty detection approach to classification. *IJCAI*, pages 518–523, August 1995.

[JRP+15] N. Jiang, W. Rong, B. Peng, Y. Nie, and Z. Xiong. An empirical analysis of different sparse penalties for autoencoder in unsupervised feature learning. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, July 2015. ISSN: 2161-4407.

[JS14] V. Jumutc and J. A. K. Suykens. Multi-Class Supervised Novelty Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(12):2510–2523, December 2014. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[KCK+13] P. Kasinathan, G. Costamagna, H. Khaleel, C. Pastrone, and M. A. Spirito. DEMO: An IDS Framework for Internet of Things Empowered by 6lowpan. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, pages 1337–1340, New York, NY, USA, November 2013. ACM.

[Kep16] N. Kephart. The DDoS Attack on Dyn's DNS Infrastructure, October 2016.

[Ker19a] Keras. Activations - Keras Documentation (https://keras.io/activations/), May 2019.

[Ker19b] Keras. Guide to the Sequential model - Keras Documentation (https://keras.io/getting-started/sequential-model-guide/), May 2019.

[Ker19c] Keras. Initializers - Keras Documentation (https://keras.io/initializers/), May 2019.

[KH05] N Kayacik and M Heywood. Selecting Features for Intrusion Detection: A Feature Relevance Analysis on KDD 99 Intrusion Detection Datasets. In *The 3rd Annual Conference on Privacy, Security and Trust (PST)*, 2005.

[KKSG16] C. Kolias, G. Kambourakis, A. Stavrou, and S. Gritzalis. Intrusion Detection in 802.11 Networks: Empirical Evaluation of Threats and a Public Dataset. *IEEE Communications Surveys Tutorials*, 18(1):184–208, 2016.

[KMY+17] J Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon. Federated Learning: Strategies for Improving Communication Efficiency. *arXiv:1610.05492 [cs]*, October 2017. arXiv: 1610.05492.

[KPSV13] P. Kasinathan, C. Pastrone, M. A. Spirito, and M. Vinkovits. Denial-of-Service detection in 6lowpan based internet of things. In *IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications*, pages pp. 600–607, 2013.

[Kra16] B. Krawczyk. Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, 5(4):221–232, November 2016.

[KT17] S. S. Khan and B. Taati. Detecting unseen falls from wearable devices using channel-wise ensemble of autoencoders. *Expert Systems with Applications*, 87:280–290, November 2017.

[KU14] S. Krushang and H. Upadhyay. A Survey: DDOS Attack on Internet of Things. *International Journal of Engineering Research and Development*, Volume 10(Issue 11):58–63, November 2014.

[Kum14] G. Kumar. Evaluation Metrics for Intrusion Detection Systems - A Study. (11):7, November 2014.

[KW03] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: attacks and countermeasures. In *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications, 2003.*, pages 113–127, May 2003.

[KW14] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. *arXiv:1312.6114 [cs, stat]*, May 2014. arXiv: 1312.6114.

[Lir16] S. Liron. Mirai: The IoT Bot that Took Down Krebs and Launched a Tbps Attack on OVH, October 2016.

[LKT15] W. C. Lin, S. W. Ke, and C. F. Tsai. CANN: An intrusion detection system based on combining cluster centers and nearest neighbors. *Knowledge-Based Systems*, 78:13–21, April 2015.

[LL05] K. Leung and C. Leckie. Unsupervised Anomaly Detection in Network Intrusion Detection Using Clusters. *Proceedings of the Twenty-eighth Australasian conference on Computer Science*, 38:333–342, 2005.

[LLL+10] B. Lei, X. Li, Z. Liu, C. Morisset, and V. Stolz. Robustness testing for software components. *Science of Computer Programming*, 75(10):879–897, October 2010.

[LMCSEL17] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret. Conditional Variational Autoencoder for Prediction and Feature Recovery Applied to Intrusion Detection in IoT. *Sensors*, 17(9):1967, August 2017.

[LTVNF16] A. E. Lazzaretti, D. M. J. Tax, H. Vieira Neto, and V. H. Ferreira. Novelty detection and multi-class classification in power distribution voltage waveforms. *Expert Systems with Applications*, 45:322–330, March 2016.

[MC03] M. V. Mahoney and P. K. Chan. An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection. In *Re-*

cent Advances in Intrusion Detection, Lecture Notes in Computer Science, pages 220–237. Springer, Berlin, Heidelberg, September 2003.

[Mic05] Microsoft. The STRIDE Threat Model, 2005.

[MKH93] M. M. Moya, M. W. Koch, and L. D. Hostetler. One-class classifier networks for target recognition applications. *NASA STI/Recon Technical Report N*, 93, 1993.

[MPB+13] C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, and M. Rajarajan. A survey of intrusion detection techniques in Cloud. *Journal of Network and Computer Applications*, 36(1):42–57, January 2013.

[MPVT17] P. Mishra, E. S. Pilli, V. Varadharajan, and U. Tupakula. Intrusion detection techniques in cloud environment: A survey. *Journal of Network and Computer Applications*, 77:18–47, January 2017.

[MRB+18] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth. Machine learning for internet of things data analysis: a survey. *Digital Communications and Networks*, 4(3):161–175, August 2018.

[MRMB17] D. Midi, A. Rullo, A. Mudgerikar, and E. Bertino. Kalis A System for Knowledge Driven Adaptable Intrusion Detection for the Internet of Things. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 656–666, June 2017.

[MS15] 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)*, pages 1–6, November 2015.

[MTC18] N. Moustafa, B. Turnbull, and K. R. Choo. An Ensemble Intrusion Detection Technique based on proposed Statistical Flow Features for Protecting Network Traffic of Internet of Things. *IEEE Internet of Things Journal*, pages 1–1, September 2018.

[MVTP18] P. Mishra, V. Varadharajan, U. Tupakula, and E. S. Pilli. A Detailed Investigation and Analysis of using Machine Learning Techniques for Intrusion Detection. *IEEE Communications Surveys Tutorials*, pages 1–1, June 2018.

[MYAZ15] R. Mahmoud, T. Yousuf, F. Aloul, and I. Zualkernan. Internet of things (IoT) security: Current status, challenges and prospective measures. In *2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)*, pages 336–341, December 2015.

[NFP10] H. Nguyen, K. Franke, and S. Petrovic. Improving Effectiveness of Intrusion Detection by Correlation Feature Selection. In *2010 International Conference on Availability, Reliability and Security*, pages 17–24, February 2010.

[Ng11] A. Ng. Sparse autoencoder. *CS294A Lecture notes 72.2011 (1-19)*, 2011.

[NMM+19] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A. R. Sadeghi. DÏoT: A Federated Self-learning Anomaly Detection System for IoT. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 756–767, July 2019. ISSN: 2575-8411.

[noaa] ADFA-IDS-DATASET (https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-IDS-Datasets/).

[noab] EtherApe, a graphical network monitor (https://etherape.sourceforge.io/).

[noac] Ettercap Home Page (https://www.ettercap-project.org/).

[noad] Network Information Management and Security Group (NIMS) (https://projects.cs.dal.ca/projectx/Download.html).

[noae] scikit-learn: machine learning in Python — scikit-learn 0.22.2 documentation (https://scikit-learn.org/stable/).

[noaf] Wireshark · Go Deep. (https://www.wireshark.org/).

[noa16] NSL-KDD | Datasets | Research | Canadian Institute for Cybersecurity | UNB, 2016.

[noa17a] ARGUS- Auditing Network Activity, 2017.

[noa17b] Metasploit | Penetration Testing Software, Pen Testing Security, 2017.

[noa18] CSE-CIC-IDS2018 | Datasets | Research | Canadian Institute for Cybersecurity | UNB, 2018.

[NSB16] M. Nobakht, V. Sivaraman, and R. Boreli. A Host-Based Intrusion Detection and Mitigation Framework for Smart Home IoT Using OpenFlow. In *2016 11th International Conference on Availability, Reliability and Security (ARES)*, pages 147–156, August 2016.

[One16] OneM2M. TS-0004-V2.7.1 Service Layer Core Protocol Specification. page 427, August 2016.

[One18a] OneM2M. TS-0001-V2.18.1 Functional Architecture. page 427, March 2018.

[One18b] OneM2M. TS-0003-V2.12.1 Security Solutions. page 427, March 2018.

[One19] OneM2M. oneM2m - Home (http://www.onem2m.org/), April 2019.

[Ora] Oracle. Java Message Service Concepts - The Java EE 6 Tutorial (https://docs.oracle.com/javaee/6/tutorial/doc/bncdq.html).

[PCCT14] M. A. F. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko. A review of novelty detection. *Signal Processing*, 99:215–249, June 2014.

[PF09] J. P. Papa and A. X. Falcão. A Learning Algorithm for the Optimum-Path Forest Classifier. In A. Torsello, F. Escolano, and L. Brun, editors, *Graph-Based Representations in Pattern Recognition*, Lecture Notes in Computer Science, pages 195–204. Springer Berlin Heidelberg, May 2009.

[PJ14] B. Prabadevi and N. Jeyanthi. Distributed Denial of service attacks and its effects on Cloud environment- a survey. In *The 2014 International Symposium on Networks, Computers and Communications*, pages 1–5, June 2014.

[PJK+16] H. H. Pajouh, R. Javidan, R. Khayami, D. Ali, and K. K. R. Choo. A Two-layer Dimension Reduction and Two-tier Classification Model for Anomaly-Based Intrusion Detection in IoT Backbone Networks. *IEEE Transactions on Emerging Topics in Computing*, PP(99):1–1, November 2016.

[PKRM12] S. Patil, P. Kulkarni, P. Rane, and B.B Meshram. IDS vs IPS. *International Journal of Computer Networks and Wireless Communications*, V 2(Issue 1), 2012.

[PP07] A. Patcha and J. M. Park. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer Networks*, 51(12):3448–3470, August 2007.

[pre] Prediction Market (https://predict.org/).

[PSS18] S. Prabavathy, K. Sundarakantham, and S. M. Shalinie. Design of cognitive fog computing for intrusion detection in Internet of Things. *Journal of Communications and Networks*, 20(3):291–298, June 2018.

[PTPB10] C. R. Perez-Toro, R. K. Panta, and S. Bagchi. RDAS: Reputation-Based Resilient Data Aggregation in Sensor Network. In *2010 7th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, pages 1–9, June 2010. ISSN: 2155-5494.

[Pug16] J. F. Puget. What Is Machine Learning? (IT Best Kept Secret Is Optimization), May 2016.

[Qui86] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, March 1986.

[Qui14] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Elsevier, June 2014. Google-Books-ID: b3ujBQAAQBAJ.

[RCF09] L. M. Rocha, F. A. M. Cappabianco, and A. X. Falcão. Data clustering as an optimum-path forest problem with applications in image analysis. *International Journal of Imaging Systems and Technology*, 19(2):50–68, 2009. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/ima.20191.

[RG97] G. Ritter and M. T. Gallegos. Outliers in statistical pattern recognition and an application to automatic chromosome classification. *Pattern Recognition Letters*, 18(6):525–539, June 1997.

[Rif14] J. Rifkin. The Zero Marginal Cost Society: The Internet of Things, the Collaborative Commons, and the Eclipse of Capitalism: Book. April 2014.

[Ros14] C. Rossow. Amplification hell: Revisiting network protocols for DDoS abuse. *Network and Distributed System Security Symposium*, February 2014.

[RP18] S. Rathore and J. H. Park. Semi-supervised learning based distributed attack detection framework for IoT. *Applied Soft Computing*, 72:79–89, November 2018.

[RWV13] S. Raza, L. Wallgren, and T. Voigt. SVELTE: Real-time intrusion detection in the Internet of Things. *Ad Hoc Networks*, 11(8):2661–2674, November 2013.

[Sal94] S. L. Salzberg. C4.5: Programs for Machine Learning by J. Ross Quinlan. Morgan Kaufmann Publishers, Inc., 1993. *Machine Learning*, 16(3):235–240, September 1994.

[SAU17] A. K. Sikder, H. Aksu, and A S. Uluagac. 6thsense: A Context-aware Sensor-based Attack Detector for Smart Devices. *26th USENIX Security Symposium (USENIX Security 17)*, page 19, August 2017.

[SB16] M. Sheikhan and H. Bostani. A hybrid intrusion detection architecture for Internet of things. In *2016 8th International Symposium on Telecommunications (IST)*, pages 601–606, September 2016.

[SB17] M. Sheikhan and H. Bostani. A Security Mechanism for Detecting Intrusions in Internet of Things Using Selected Features Based on MI-BGSA. *International Journal of Information & Communication Technology Research*, 9(2):53–62, October 2017.

[Sha76] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, April 1976. Google-Books-ID: wug9DwAAQBAJ.

[SHHS] A. Sivanathan, A. Hamza, Hassan Habibi, and V. Sivaraman. UNSW Proliferation Dataset.

[SHLG18] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization:. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy*, pages 108–116, Funchal, Madeira, Portugal, 2018. SCITEPRESS - Science and Technology Publications.

[SHM+16] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, January 2016.

[SJS15] L. Sujihelen, C. Jayakumar, and C. S. Singh. Detecting Node Replication Attacks in Wireless Sensor Networks: Survey. *Indian Journal of Science and Technology*, 8(16), July 2015.

[SLSGZ19] S. Saharan, V. Laxmi, M. Singh Gaur, and A. Zemmari. Privacy Preserving Data Offloading Based on Transformation. In A. Zemmari, M. Mosbah, N. Cuppens-Boulahia, and F. Cuppens, editors, *Risks and Security of Internet and Systems*, Lecture Notes in Computer Science, pages 86–92, Cham, 2019. Springer International Publishing.

[SRGCP15] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini. Security, privacy and trust in Internet of Things: The road ahead. *Computer Networks*, 76:146–164, January 2015.

[SSG+17] A. Sivanathan, D. Sherratt, H. H. Gharakheili, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman. Characterizing and classifying IoT traffic in smart cities and campuses. In *2017 IEEE Conference*

on Computer Communications Workshops (INFOCOM WKSHPS), pages 559–564, May 2017.

[SSJ14] S. K. Sahu, S. Sarangi, and S. K. Jena. A detail analysis on intrusion detection datasets. In *2014 IEEE International Advance Computing Conference (IACC)*, pages 1348–1353, February 2014.

[SSTG12] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers & Security*, 31(3):357–374, May 2012.

[STJ14] D. Singh, G. Tripathi, and A. J. Jara. A survey of Internet-of-Things: Future vision, architecture, challenges and services. In *2014 IEEE World Forum on Internet of Things (WF-IoT)*, pages 287–292, March 2014.

[STO+11] J. Song, H. Takakura, Y. Okabe, M. Eto, D. Inoue, and K. Nakao. Statistical analysis of honeypot data and building of Kyoto 2006+ dataset for NIDS evaluation. In *Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security*, BADGERS '11, pages 29–36, Salzburg, Austria, April 2011. Association for Computing Machinery.

[SU14] K. Sonar and H. Upadhyay. A Survey: DDOS Attack on Internet of Things. *International Journal of Engineering Research and Development*, 10(11):58–63, 2014.

[SU16] M. Surendar and A. Umamakeswari. InDReS: An Intrusion Detection and response system for Internet of Things with 6lowpan. In *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pages 1903–1908, March 2016.

[SWS+00] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J.C. Platt. Support Vector Method for Novelty Detection. In S. A. Solla, T. K.

Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 582–588. MIT Press, 2000.

[Tan03] M. Tanase. IP Spoofing: An Introduction | Symantec Connect Community, March 2003.

[Tax02] D. M. J. Tax. One-class classification: Concept learning in the absence of counter-examples. page 1, 2002.

[tcp17] tcpdump. Tcpdump/Libpcap public repository, 2017.

[TDB12] P. Tsankov, M. T. Dashti, and D. Basin. SecFuzz: Fuzz-testing Security Protocols. In *Proceedings of the 7th International Workshop on Automation of Software Test*, AST '12, pages 1–7, Piscataway, NJ, USA, June 2012. IEEE Press.

[Thi17] V. L. L. Thing. IEEE 802.11 Network Anomaly Detection and Attack Classification: A Deep Learning Approach. In *2017 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6, March 2017.

[TKG+17] G. Tuna, D. G. Kogias, V. C. Gungor, C. Gezer, E. Taşkın, and E. Ayday. A survey on information security threats and solutions for Machine to Machine (M2m) communications. *Journal of Parallel and Distributed Computing*, 109:142–154, November 2017.

[Tre96] J. Tretmans. *Conformance Testing with Labelled Transition Systems: Implementation Relations and Test Generation*, volume 29 of *Computer Networks*. 1996.

[Vap13] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Science & Business Media, June 2013. Google-Books-ID: EqgACAAAQBAJ.

[VF14] O. Vermesan and P. Friess. Internet of Things Applications - From Research and Innovation to Market Deployment Book. *River Publishers*, June 2014.

[VHS11] A. R. Vasudevan, E. Harshini, and S. Selvakumar. SSENet-2011: A Network Intrusion Detection System dataset and its comparison with KDD CUP 99 dataset. In *2011 Second Asian Himalayas International Conference on Internet (AH-ICI)*, pages 1–5, November 2011.

[WAB+17] C. Y. Wu, A. Ahmed, A. Beutel, A. J. Smola, and H. Jing. Recurrent Recommender Networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, WSDM '17, pages 495–503, Cambridge, United Kingdom, February 2017. Association for Computing Machinery.

[WF02] I. H. Witten and E. Frank. Data mining: practical machine learning tools and techniques with Java implementations. *ACM SIGMOD Record*, 31(1):76–77, March 2002.

[WJ17] L. Wang and R. Jones. Big Data Analytics for Network Intrusion Detection: A Survey. *International Journal of Networks and Communications*, 7(1):24–31, 2017.

[WOM14] O. A. Wahab, H. Otrok, and A. Mourad. A cooperative watchdog model based on Dempster–Shafer for detecting misbehaving vehicles. *Computer Communications*, 41:43–54, March 2014.

[WRV13] L. Wallgren, S. Raza, and T. Voigt. Routing Attacks and Countermeasures in the RPL-Based Internet of Things. *International Journal of Distributed Sensor Networks*, 9(8):794326, August 2013.

[WS84] Q. R. Wang and C. Y. Suen. Analysis and Design of a Decision Tree Based on Entropy Reduction and Its Application to Large Character Set

Recognition. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 4:406–417, 1984.

[XL05] C. Xiang and S. M. Lim. Design of Multiple-Level Hybrid Classifier for Intrusion Detection System. In *2005 IEEE Workshop on Machine Learning for Signal Processing*, pages 117–122, September 2005.

[YWY+17] Y. Yang, L. Wu, G. Yin, L. Li, and H. Zhao. A Survey on Security and Privacy Issues in Internet-of-Things. *IEEE Internet of Things Journal*, 4(5):1250–1258, October 2017.

[ZCW+14] Z. Zhang, M. C. Y. Cho, C. Wang, C. Hsu, C. Chen, and S. Shieh. IoT Security: Ongoing Challenges and Research Opportunities. In *2014 IEEE 7th International Conference on Service-Oriented Computing and Applications*, pages 230–234, November 2014.

[ZFS17] V. Zlomislić, K. Fertalj, and V. Sruk. Denial of service attacks, defences and research challenges. *Cluster Computing*, 20(1):661–671, March 2017.

[ZMKdA17] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga. A survey of intrusion detection in Internet of Things. *Journal of Network and Computer Applications*, 84:25–37, April 2017.

[ZYX+08] Z. Zeng, H. B. Yu, H. R. Xu, Y. Q. Xie, and J. Gao. Fast training Support Vector Machines using parallel sequential minimal optimization. volume 1, pages 997–1001, November 2008.