MASTER EN SCIENCES INFORMATIQUES

Fingerprinting de devices IoT à l'aide de l'apprentissage automatique

Maniraguha, Clément

Award date:
2019

Awarding institution:
Universite de Namur

Université de Namur
Faculté d'informatique
Année académique 2018–2019

Fingerprinting de devices IoT à l'aide de l'apprentissage automatique

Clément Maniraguha-Kiruhura

Promoteur : _____ (Signature pour approbation du dépôt - REE art. 40)
Jean-Noël Colin

Mémoire présenté en vue de l'obtention du grade de
Master en Sciences Informatiques.

Résumé

Internet connaît aujourd'hui une extension inédite avec l'émergence des objets connectés. Que ce soit dans un réseau local d'entreprise, ou un dans un réseau privé à la maison, ces nouveaux appareils électroniques d'un genre nouveau ont fait irruption massivement dans le quotidien.
Bien que l'utilisation de dispositifs IoT puisse présenter de nombreux avantages en matière d'efficacité, de confort et de coût, leur utilisation intensive soulève plusieurs problèmes de sécurité et de confidentialité.
Dans ce contexte, l'identification des dispositifs qui évoluent dans un réseau constitue une composante essentielle des outils de gestion de réseau, car elle fournit des informations essentielles en matière de surveillance de réseau.
Dans ce travail, nous nous appuyons sur l'apprentissage automatique pour explorer une méthode d'identification de dispositifs IoT sur base d'analyse de données issues du trafic réseau.

Mots-clés : Empreinte digitale d'appareil, apprentissage automatique, IoT, Internet des objets

Abstract

The Internet is now experiencing an unprecedented extension with the emergence of connected objects. Whether in a local business network, or in a private home network, these new electronic devices of a new kind are massively affecting our everyday lives.
Although the use of IoT devices can have many advantages in terms of efficiency, comfort and cost, their intensive use raises several security and confidentiality issues.
With this in mind, the identification of devices that evolve in a network is an essential component of network management tools because it provides essential information about network monitoring information.
In this work, we rely on machine learning to explore a method of identifying IoT devices based on data analysis from network traffic.

Keywords : Device fingerprinting, Machine Learning, IoT, Smart Devices, Internet of Things

Remerciements

Je tiens tout d'abord à remercier mon promoteur, Mr Jean-Noël Colin, Professeur à l'Université de Namur, pour m'avoir guidé et orienté dans mes travaux, ainsi que Mr Sereysethy Touch pour son aide.

Je tiens également à remercier Mathieu Van Moorsel, mon ancien collègue pour ses conseils éclairés en matière de réseaux informatiques.

Je tiens aussi à remercier ma famille, qui m'a accompagné tout au long de mes études à horaire décalé, plus particulièrement ma compagne pour l'aide et la patience qu'elle m'a accordée.

Je souhaite finalement remercier toute personne ayant contribué d'une manière ou d'une autre à la réalisation du présent mémoire.

A la mémoire de mon frère Thierry,

Table des matières

Liste d’abréviations

DNS : Domain Name System
FNP : Faux négatifs
FPP : Faux positifs
ICMP : Transmission control protocol
IOT : Internet Of Things
M2M : Machine to Machine
MAC : Medium Access Control
MLP : Multi-Layer Perceptron
NIDS : Network Based Intrusion Detection System
NTP : Network Time Protocol
SMOTE : Synthetic Minority Over-sampling Technique
SVM : Support Vector Machine
TCP : Transmission control protocol
TSF : Timing Synchronization Function
WLAN : Wireless Local Area Network

Chapitre 1 : Introduction

Le terme « internet des objets » apparaît en 1999, lorsque Kevin Ashton, collaborateur de Procter & Gamble, l'utilise au cours d'une présentation intitulée "Internet of Things" pour désigner le lien entre la technologique RFID et l'Internet [1].

Aujourd'hui, les objets qui n'étaient pas jusqu'alors connectés et qui nous entourent ont désormais la possibilité de communiquer d'objets à objets, d'objets à personnes ou de personnes à objets, pour rendre notre environnement plus intelligent.
L'IoT est déjà très présent dans nos vies de tous les jours à travers différents objets : bracelet, vêtement, maison, voiture, senseurs corporels, ville intelligente, etc.
Des revues spécialisées rivalisent de superlatifs pour décrire ce que l'on qualifie déjà comme faisant partie de la quatrième révolution industrielle : il y a déjà plus d'objets connectés que d'habitants sur la planète.

Ce marché IoT émergent en plein essor introduit de nouveaux défis et pose des interrogations notamment en termes de sécurité pour les administrateurs réseau. Compte tenu de la grande diversité d'objets connectés et en partie due à une course acharnée des acteurs de l'IoT, il devient donc nécessaire d'être en mesure de sécuriser ces différents objets.

1.1 Définitions et concepts de base

Dans cette section, nous définissons quelques concepts importants pour soutenir la compréhension du présent travail.

1.1.1 Internet des Objets (IoT)

L'Internet des objets connectés fait référence à l'interconnexion entre les réseaux informatiques traditionnels comme l'Internet et des objets, des lieux et des environnements physiques. Les objets connectés peuvent être d'usages aussi divers que variés, dans le domaine de l'e-santé, de la logistique, de l'industrie, de la ville intelligente, de la domotique, sécurité, et bien d'autres encore.
L'Internet des objets est sans commune mesure en partie responsable d'un accroissement exponentiel du volume de données généré sur le réseau.

1.1.2 Dispositif IoT

Un dispositif IoT est un composant matériel doté d'un capteur leur permettant de générer, d'échanger et de consommer des données avec une intervention humaine minimale. Les types de dispositifs s'étendent aux objets, capteurs et éléments de la vie courante qui ne sont normalement pas considérés comme des ordinateurs.

Dans la suite du présent mémoire, nous réfèrerons à ces objets sous l'appellation de dispositifs IoT pour désigner les objets connectés que l'on retrouve facilement dans une maison dite « intelligente » comme un éclairage connecté, un détecteur de mouvement ou un frigo intelligent. A contrario, un smartphone ou une imprimante intelligente ne sont pas repris sous cette appellation.
La nuance se situe au niveau du degré d'interaction humaine requis pour que l'un ou l'autre objet émette ou reçoive des[1] données à traiter.

1.1.3 Domotique

La domotique[1] est l'ensemble des techniques de l'électronique, de physique du bâtiment, d'automatisme, de l'informatique et des télécommunications utilisées dans les bâtiments et permettant de centraliser le contrôle des différents applicatifs de la maison (système de chauffage, volets roulants, porte de garage, portail d'entrée, prises électriques, etc.).

Dans le présent travail, la domotique est discutée à travers le prisme des objets connectés évoluant dans une maison intelligente ou *smart home* en anglais, et connectés à l'Internet depuis un réseau domestique.
Un environnement semblable a été mis place dans le cadre d'un labo impliquant des dispositifs IoT, un Raspberry Pi [2] dans lequel un firmware OpenWrt [3] tourne. Pour ce dernier, à titre d'information, le constructeur Vera[2] s'appuie sur le firmware OpenWrt pour proposer des contrôleurs domotiques.

[1] Domotique, https://fr.wikipedia.org/wiki/Domotique
[2] Vera, https://getvera.com/

1.1.4 Fingerprinting de devices

Le fingerprinting de devices que l'on traduit « empreinte digitale d'appareil » en littérature, est un processus par lequel un périphérique ou le logiciel qu'il exécute est identifié sur base de ses caractéristiques observables de l'extérieur [4].

Historiquement les principales caractéristiques comprennent le système d'exploitation, le protocole, le navigateur, la version du logiciel utilisée. Toutes ces caractéristiques forment une signature unique permettant d'identifier le device.
Dans ce mémoire, le fingerprinting de dispositifs IoT est effectué à partir des statistiques des données de flux réseau générées par ces derniers visant à identifier le type de dispositif IoT.

A cet égard, deux dispositifs IoT t_1 et t_2 sont dits du même type T s'ils partagent le même modèle ainsi que la même version de logiciel. Ce choix se justifie par le fait qu'une modification dans le logiciel peut sensiblement modifier les caractéristiques précédemment établies.

1.1.5 Apprentissage Automatique

L'apprentissage automatique ou machine learning en anglais, est une branche de l'intelligence artificielle qui consiste à créer des algorithmes capables de s'améliore automatiquement avec l'expérience.
Il est intégré de plus en plus dans la plupart des technologies que nous utilisons au quotidien. Qu'il s'agisse de recommandations suggérées par des fournisseurs de contenus comme Netflix, de placement de contenus publicitaires sur mesure sur un site marchand ou bien un filtre anti-spam capable d'apprendre à identifier les e-mails frauduleux à partir d'exemples de spam et des messages normaux.

La machine « apprend » des données préalables et adapte ses réponses. Faire de l'apprentissage automatique suppose d'utiliser des jeux de données de différentes tailles, afin d'identifier des similitudes, corrélations et différences.
En outre, l'apprentissage automatique fait largement appel à des outils et des concepts de la statistique, et fait partie d'une discipline plus vaste appelée « science des données ».

Il existe trois grands types d'apprentissage automatique :

- L'**apprentissage supervisé** a pour but d'établir des règles de comportement à partir d'un ensemble de données contenant des exemples de cas déjà étiquetés. Les données consistent en un ensemble de couples entrées / sorties {(X, Y)}. L'algorithme est formé en mappant les entrées sur les sorties (Y = f (X)). Lorsque vous fournissez une nouvelle entrée, l'algorithme devrait prédire la sortie.
  Autrement dit, étant donné un ensemble de caractéristiques $\{(x^{(1)}, \ldots, x^{(m)}\}$ associés un ensemble de sorties $\{(y^{(1)}, \ldots, y^{(m)}\}$ on veut construire un classifieur qui apprend à prédire y depuis x.
  Il existe deux types de modèles prédictifs : de classification et de régression.

- L'**apprentissage non supervisé**, contrairement à l'apprentissage supervisé, le non supervisé traite le cas où l'on dispose seulement des entrées {X} sans avoir au préalable les sorties. Le but de l'apprentissage non-supervisé est de trouver des formes cachées dans un jeu de données non-étiquetées $\{(x^{(1)}, \ldots, x^{(m)}\}$. Le problème

d'apprentissage non supervisé le plus fréquent est la segmentation (ou clustering) où l'on essaie de séparer les données en groupes (catégorie, classe, cluster…).

- L'**apprentissage par renforcement** est un type d'apprentissage automatique dans lequel un algorithme n'a pas de données d'entraînement au début. L'objectif est qu'un agent évolue dans un environnement et tire des leçons de sa propre expérience. Pour qu'un algorithme d'apprentissage par renforcement fonctionne, l'environnement dans lequel il évolue doit être calculable et avoir une fonction de récompense (ou de pénalité sous la forme de récompense négative) qui évalue la qualité d'un agent.

Le fingerprinting de dispositifs IoT présenté dans ce travail s'appuie sur l'apprentissage supervisé. Plus précisément, il est traité comme un problème de classification supervisée.
A cette fin, six algorithmes de classification ont été utilisés :

- **Classifieur bayésien naïf.**

  Il s'agit d'une méthode de classification qui se base principalement sur le théorème de Bayes. Ce dernier est un classique de la théorie des probabilités. Ce théorème est fondé sur les probabilités conditionnelles.
  Sa caractéristique principale est qu'il émet une hypothèse forte a priori de l'indépendance des caractéristiques considérées, ignorant ainsi les corrélations pouvant exister entre elles. Néanmoins, cet algorithme a l'avantage d'être simple et rapide.

- **Machine à vecteurs de support (SVM).**

  Les SVM sont plus généralement utilisés dans les cas de classification. Ils reposent sur l'idée de trouver un hyperplan qui divise au mieux un jeu de données en deux classes. Les vecteurs de support sont les points de données les plus proches de l'hyperplan.
  Le classifieur SVM tente de trouver l'hyperplan avec une marge maximale séparant les deux classes, le terme « marge » indique la distance minimale entre le plan de séparation et les points de chaque côté.

- **Adaptive Boosting (AdaBoost).**

  Le boosting est une technique qui vise à convertir des règles de prédiction peu performantes en des règles plus performantes. Adaboost (Freund et Schapire (1996)) est un des algorithmes les plus utilisés de la famille du boosting. Il génère un ensemble d'apprenant faibles de façon itérative et les combine avec le vote majoritaire et construit un classifieur très efficace. Chaque apprenant faible est entraîné de sorte à prendre en compte les erreurs de classification de l'apprenant précèdent.

- **Forêts d'arbres décisionnels (Random Forest).**

  Il s'agit une technique d'apprentissage supervisé, inventé par Breiman en 2001, qui combine une technique d'agrégation, le « Bagging » (Breiman, 1996), et une technique particulière d'induction d'arbres de décision. Les forêts aléatoires sont formées par simple assemblage d'arbres de décision multiples, allant généralement de quelques dizaines à des milliers d'arbres. Après avoir entraîné chaque arbre de décision individuellement sur un sous ensemble aléatoire de données selon le principe du bagging, la prédiction de la forêt est alors obtenue en moyennant les prédictions des arbres [5]. En outre, le processus aléatoire dans la construction des arbres permet d'assurer une faible corrélation entre derniers. Les forêts sont également connues pour leur précision et leur capacité à traiter des jeux de données composés de peu d'observations et de nombreuses caractéristiques. Par ailleurs, il est possible de réaliser les calculs en parallèles pour tirer parti des possibilités des machines à processeurs multi-cœurs par exemple.

- **eXtreme Gradient Boosting (XGBoost).**

  Il s'agit une implémentation open source optimisée et parallélisée du Gradient Boosting, créée par Tianqi Chen [6] avant la contribution de nombreux développeurs. Le principe de base du Boosting de Gradient est de combiner les résultats d'un ensemble de modèles plus simple et plus faibles afin de fournir une meilleure prédiction [7]. Tout comme Random Forest, XGBoost utilise des arbres de décision pour résoudre des problèmes de classification et de régression.

- **Perceptron multicouche (ou MLP, pour Multilayer Perceptron).**

  Le perceptron multicouche est un réseau orienté de neurones artificiels composé d'au moins trois couches où l'information voyage dans un seul sens, de la couche d'entrée vers la couche de sortie en passant par la ou les couche(s) cachée(s). Une couche reçoit un vecteur d'entrée et le transforme en vecteur de sortie.

1.2 Problématique

Le nombre croissant d'objets connectés à Internet capables de communiquer entre eux ne cesse d'augmenter à un rythme soutenu. Cette tendance tend à s'accentuer avec la multiplication d'acteurs aussi bien fabricants que fournisseurs.
L'internet des objets (IoT) en s'appuyant sur des réseaux traditionnels auxquels sont connectés les objets dits "intelligents", soulève de nouvelles problématiques autour de la sécurité [8] de ces réseaux et des informations qui y transitent.

Il devient donc nécessaire d'être en mesure de sécuriser ces différents objets. Ainsi, l'identification des dispositifs qui évoluent dans un réseau constitue une composante essentielle des outils de gestion de réseau, car elle fournit des informations importantes permettant notamment de s'assurer de la légitimité du trafic échangé.
Les dispositifs peuvent être identifiés sur base de leur adresse IP ou adresse MAC depuis la couche de liaison du modèle OSI. Cependant, cette approche peut être mise à mal par des attaques bien connues d'usurpation d'adresse IP.

Le spoofing est un type d'attaque dans lequel l'attaquant se fait passer pour quelqu'un d'autre afin d'accéder à des ressources limitées ou de voler des informations. L'IP spoofing [9] en est une variante, une technique de piratage informatique qui consiste à envoyer des paquets IP avec une adresse IP source falsifiée, dans le but de dissimuler l'identité de l'expéditeur ou d'emprunter l'identité d'un autre système informatique. Faisant d'elle la plus grande vulnérabilité de sécurité de l'architecture Internet (TCP / IP).

Le fingerprinting des objets connectés pose un grand défi compte tenu du grand nombre d'hétérogénéité des protocoles [10], des réseaux utilisés et peu de normes consensuelles.
Des approches récentes en matière de fingerprinting sur base d'analyse comportementale de dispositif informatique ont émergé [11]. L'idée fondamentale consiste à scruter le trafic traversant le réseau, en utilisant soit des techniques de mesures actives ou passives, et d'en extraire des patterns uniques suffisamment discriminants pour identifier individuellement les dispositifs présents au sein de ce réseau.

Dans le présent travail, nous explorons une méthode d'identification de dispositifs IoT sur base d'analyse de données issues du trafic réseau et plus particulièrement les informations contenues dans les en-têtes TCP des paquets.
Ensuite, à l'aide de techniques d'apprentissage automatique, nous avons développé des modèles de classification supervisée capable de prédire le type de dispositifs IoT.

1.3 Objectifs de recherche

La principale problématique de ce mémoire est la suivante :

Est-il possible d'identifier le type des dispositifs IoT évoluant dans un réseau au moyen de techniques reposant sur d'apprentissage automatique ?

En vue de mener à bien cet objectif, il est nécessaire d'exécuter les étapes suivantes :

1. Mettre en place un environnement domotique ;

2. Mettre au point un mécanisme de capture de trafic réseau en vue d'une analyse ultérieure ;

3. Développer un script capable de traiter et transformer les traces réseaux en jeu de données exploitables par des algorithmes d'apprentissage automatique ;

4. Implémenter et optimiser divers algorithmes d'apprentissage supervisé capables de classifier les dispositifs IoT ;

5. Procéder à l'évaluation de la performance des différents algorithmes.

1.4 Plan du mémoire

Le premier chapitre est une introduction générale, qui met l'accent sur le contexte du sujet traité, la problématique et les objectifs à atteindre.
Le deuxième chapitre présente un état de l'art sur le fingerprinting de dispositifs IoT. Le troisième chapitre se concentre sur la méthodologie suivie pour mettre en place pour valider la méthode de fingerprinting proposée dans ce travail.
Le quatrième chapitre porte sur l'évaluation de performances. Ce chapitre est composé d'une première section portant sur l'évaluation de notre propre jeu de données et une deuxième section dédiée à l'évaluation de la solution sur un autre jeu de données externes à seule fin de valider la solution proposée.
Le chapitre cinq est consacré à la discussion des solutions proposées pour la problématique décrite à la section 1.3.
La conclusion du travail ainsi que les travaux futurs sont présentées dans le chapitre six.

Chapitre 2 : Etat de l'art

Il existe une grande variété de méthodes de fingerprinting de dispositifs pouvant être principalement classée en deux catégories selon le type de surveillance réseau considéré : la surveillance active ou la surveillance passive. Un récapitulatif de comparaison entre les deux est donné dans le tableau 2.1.

Le principe d'une surveillance active consiste à générer du trafic dans le réseau et à observer les éventuelles réactions au stimulus. En tant que tel, il crée un trafic supplémentaire dans le réseau.
A contrario, dans le cas d'une surveillance passive, il s'agit d'une approche jugée moins intrusive, consistant à capturer le trafic traversant le réseau et à étudier ses propriétés en un ou plusieurs points du réseau. Généralement, cette approche nécessite des outils logiciels de capture ou d'analyse de trafic comme tcpdump[3], libpcap[4], etc.

Mise à part ces catégories, Kohno et al. [12] introduit une nouvelle approche dite de semi-passive. Dans ce cas, une surveillance active est entreprise uniquement si, en premier lieu, la machine cible a initié une connexion.

Type de surveillance	**Avantages**	**Risques/enjeux**
Active	• Résultats plus précis • Plus rapide	• Perturbation introduite par les paquets sondes • Détectable notamment par systèmes de détection d'intrusion (NIDS) • Impact sur bande passante • Besoin de collaboration de la cible
Passive	• Non-intrusive et ne change rien à l'état du réseau • Utilisable à tout moment • Non détectable • Préserve la bande passante • Permet analyse hors ligne	• Nécessité d'avoir des outils logiciels adaptés • Nécessite de collecter suffisamment de paquets • Problème de confidentialité des données • Collaboration de la cible no nécessaire

Tableau 2.1: Comparatif entre surveillance active/passive.

[3] tcpdump est un analyseur de paquets en ligne de commande https://www.tcpdump.org/
[4] libpcap fournit des fonctions pour la capture de paquets https://www.tcpdump.org/

L'état de l'art qui suit est consacré aux techniques de fingerprinting de type de dispositifs basés sur les deux types de surveillance.

2.1 Fingerprinting du système d'exploitation (OS)

Il s'agit d'un processus d'identification, sur base de technique active ou passive, du système d'exploitation d'une machine distante en analysant les paquets provenant de cette dernière. Actuellement, les outils Nmap [13], Xprobe2 [14], et P0f [15] sont les plus populaires en matière de reconnaissance de système d'exploitation.

Nmap ("Network Mapper") est un outil open source d'exploration réseau et d'audit de sécurité. Il repère les hôtes actives sur un réseau et peut reporter quels ports sont ouverts sur ces hôtes. Nmap [13], dispose également d'un module d'identification du système d'exploitation lequel, sur base des réponses particulières qu'il obtient des requêtes particulières envoyées, essaye de reconnaitre la version d'un système d'exploitation à l'aide de sa base de connaissance interne.

Sivanathan et al [16], ont conduit des tests visant à déterminer la faisabilité d'identification du type d'un dispositif IoT en sondant ses ports ouverts.
Les auteurs ont utilisé Nmap [13] pour scanner les ports de 19 dispositifs IoT provenant de leur banc d'essai, dans le but de constituer une base de connaissance de combinaisons de numéro de ports de dispositifs IoT formant ainsi leur signature.
De cette base de connaissance, il en ressort que 42 numéros de ports sont uniques, que quelques dispositifs du même vendeur partagent une même combinaison et que plusieurs numéros de ports sont exclusivement utilisés par un seul dispositif.
Sur base de ces analyses, Sivanathan et al [16] ont mis au point un script permettant d'identifier les dispositifs IoT en utilisant un arbre binaire dont chaque nœud représente un numéro de port classé selon un ordre décroissant de popularité. Ainsi le nœud parent est le port 80 suivi du 8080, 53 et ainsi de suite.

De manière itératif, le script va scanner les différents ports d'un dispositif IoT candidat pour trouver un chemin unique formant une combinaison connue, depuis le nœud racine jusqu'aux feuilles.

Les auteurs expliquent que ce mécanisme d'arbre binaire, leur ont permis d'éviter une tâche fastidieuse de scanner systématiquement les ports allant de 1 à 65535.
Cependant, cette étude se limite uniquement au protocole TCP orienté connexion, à l'inverse du protocole UDP dit « sans connexion » en raison de son manque de collaboration. Dans le sens, où la plupart du temps les ports UDP ouverts ne répondent pas aux sollicitudes d'un scan actif.

P0f [15] est un outil permettant également de faire la détection de systèmes d'exploitation, mais contrairement à Nmap [13], il opère de manière passive. Il analyse les trames transitant sur le réseau et les compare avec une base de connaissance des caractéristiques de systèmes exploitations qui lui sont déjà connues afin de trouver une correspondance.

2.2 Fingerprinting par biais d'horloge (Clock Skew)

Cette technique de fingerprinting repose, non plus sur les caractéristiques logicielles, mais sur de petites divergences qui peuvent intrinsèquement subsister dans le matériel.
Chaque machine possède une horloge interne qui tend à dévier très légèrement du temps universel dû à divers facteurs telle que : l'implémentation de l'horloge au niveau logiciel, les imperfections matérielles, la différence dans la longueur de câbles, etc [17]. Cette dérive est généralement donnée en partie par million (ppm) correspondant à un rapport d'un millionième (10^{-6}) de seconde.

Dans une étude, Moon et al. [18] propose un algorithme basé sur la programmation linéaire mesurant la dérive de l'horloge à partir des paquets réseau. Plus tard, Kohno et al. [12] sont les premiers à introduire l'idée d'un fingerprinting d'une machine distante en exploitant précisément cette mesure. Les auteurs démontrent que les biais d'horloge peuvent être distingués parmi différentes machines physiques tout en restant stables dans le temps, et dès lors, peuvent être utilisés comme indice de discrimination pour caractériser une machine précise.

Kohno et al. [12] mesurent l'estimation du biais d'horloge en observant activement ou passivement des timestamps disponibles dans les paquets TCP et ICMP échangés avec la machine cible.

Dans la méthode active, les auteurs envoient à la machine distante plusieurs ICMP timestamp requests (ICMP Type 13 Code 0) [19], en retour, celle-ci y répond (ICMP Type 14 Code 0) [19] avec un paquet contenant notamment un champ nommé « Transmit », un timestamp correspondant à l'heure de transmission ayant une résolution temporelle de l'ordre de la milliseconde. La principale limite à cet exercice est que la cible ne doit pas être derrière un NAT ou un pare-feu filtrant les ICMP.

Dans le cas de la méthode passive ou semi-passive, le timestamp est récupéré tout simplement en surveillant les paquets TCP au niveau de l'option Timestamp [20] (lorsqu'elle est activée), qui transitent lorsque la machine cible communique avec les autres hôtes du réseau.

Cependant, certains systèmes d'exploitation n'incluent pas le champ d'option TCP Timestamp dans les paquets initiant une connexion (SYN). Dans ce cas précis, Kohno et al. [12] proposent d'utiliser la méthode semi-passive consistant à modifier le paquet SYN-ACK (accusant réception) et y inclure l'option TCP Timestamp avant l'envoie. En ce moment, l'option serait présente le restant des échanges.

Dans une autre étude, Jana et al. [21] ont couvert l'utilisation du biais d'horloge dans le but de pouvoir détecter des points d'accès non autorisés dans un environnement de réseau sans fil. Au lieu de se servir du timestamp des paquets TCP et ICMP, les auteurs exploitent la Time Synchronization Function (TSF) de la spécification IEE 802.11, qui fournit un timestamp présent dans chaque trame balise (beacon frame) ou trame de réponse de sondage (probe response) d'un point accès, dans le but d'estimer son biais d'horloge.
Les auteurs estiment qu'entre 50-100 paquets sont suffisants pour l'estimation avec une précision de l'ordre de 99%.

2.3 Fingerprinting sur base de requêtes de sonde 802.11

Une station émet des trames de type requête de sonde (probe request) quand elle a besoin d'obtenir des informations d'une autre station par exemple pour déterminer quels sont les points d'accès à sa portée dans un réseau sans fil.

Etant donné que l'algorithme utilisé pour scanner les points d'accès n'est pas explicitement défini dans la norme 802.11, le choix de son implémentation est laissé libre cours aux développeurs de pilotes de périphériques sans fil.

Cette subtilité est la base de l'étude de Franklin et al. [4] qui montrent que les caractéristiques temporelles des trames de requête de sonde (probe request) peuvent être exploitées pour caractériser un pilote de la carte réseau d'un périphérique sans fil. Les auteurs se basent sur la distribution des temps inter-arrivée des trames (TIAT) illustré dans la figure 2.1.

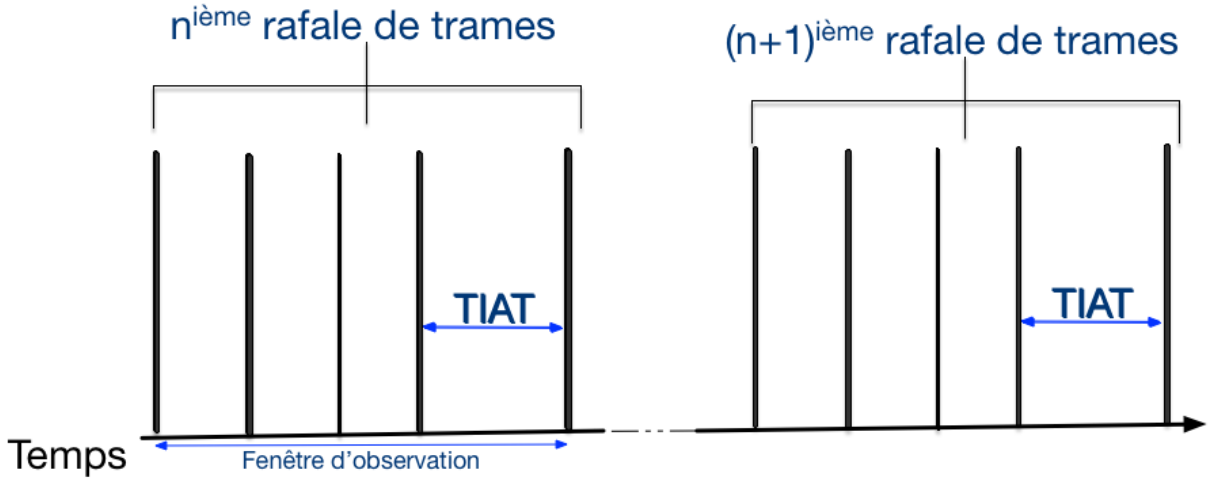

Figure 2.1 : Séquence de transmission de trames regroupées en rafale avec le temps inter-arrivée des trames (TIAT).

La technique utilise un processus en deux étapes. La première consiste en la réalisation de capture de trames de requête de sonde (probe request) de façon passive. S'ensuit l'étape de la génération de signature en se basant sur le delta de temps d'arrivé entre les requêtes de sonde. Les signatures sont obtenues par un procédé de discrétisation en classes d'amplitudes égales délimitées empiriquement.
Deux attributs sont soigneusement sélectionnés pour créer une signature, à savoir la taille de classe et la moyenne effective de chacune des classes.

Pour identifier un nouveau pilote, les auteurs utilisent un algorithme reposant sur la distance métrique qui compare la nouvelle signature avec celles de la base de connaissances des signatures.
Les auteurs conduisent 3 tests de la méthode sur des traces de trames récoltées sur 17 périphériques sans fil suivant différents scénarios. L'évaluation de la méthode est de 96, 84 et 77% sur 3 tests respectifs.
Cependant, la méthode comporte quelques limitations :

- l'impossibilité de distinguer différentes versions d'un même pilote ;

- l’existence d’une couche d'abstraction matérielle [22] peut avoir un impact la diversité de pilotes par conséquent sur qualité du fingerprinting ;
- de même que sa précision est sensible aux conditions du réseau.

Les auteurs notent aussi qu’une éventuelle standardisation de la spécification du taux auquel les trames de requête de sonde sont transmises rendrait la technique inefficace. Enfin, certains appareils proposent la possibilité de désactiver l’envoi de ces trames, ce qui constitue un obstacle de plus.

Une autre étude [23] s’est inspirée de cette technique pour démontrer la faisabilité de suivre l’emplacement d’une station (STA) au fil du temps malgré la mise en place de la fonctionnalité des adresses réseau (MAC) aléatoires, en caractérisant précisément les trames requêtes de sonde que l’on peut collecter au moment de la période de découverte de service actif de stations.

2.4 Fingerprinting de type stimulus-réponse des périphériques sans fil

Bratus et al. [24] proposent une méthode active pour le fingerprinting des périphériques sans fil 802.11. L’idée principale consiste à envoyer à une station cible une série de trames mal formées, inutiles ou dénuées de tout formalisme et d’observer la réaction au stimulus. L’hypothèse étant que la nature des réponses varie suffisamment d’une station à une autre servant ainsi d’attribut pour le fingerprinting.

Avant ça, dans une étude empirique Gopinath et al. [25] mettent en évidence le fait que les périphériques WLAN de différents constructeurs sur le marché présentent une hétérogénéité au niveau de l'implémentation de la sous-couche de contrôle d'accès au support (c.-à-d. Medium Access Control ou MAC). Les auteurs soulignent également la possibilité d’exploiter cette hétérogénéité pour le fingerprinting du constructeur/modèle d’un périphérique sans fil.

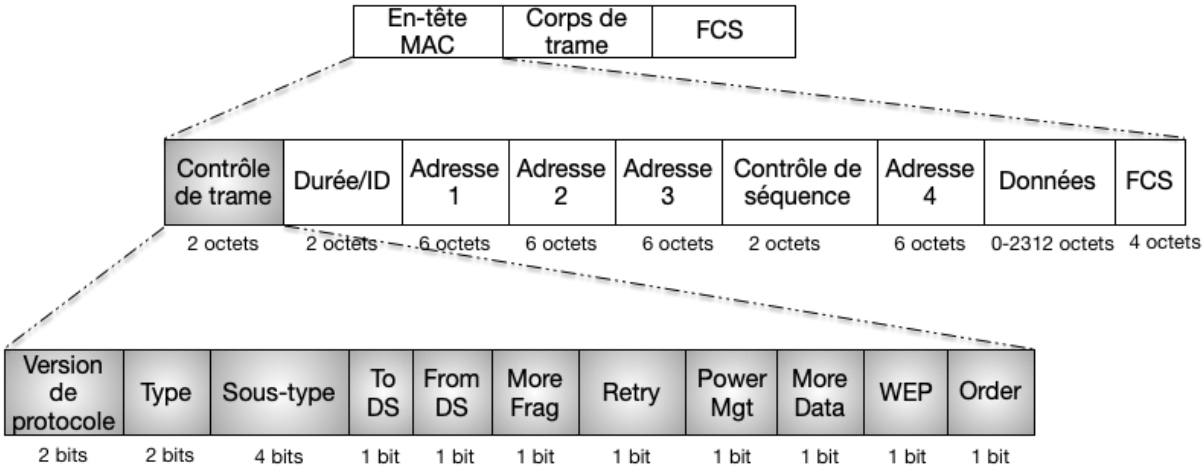

Figure 2.2 : Le champ « contrôle de trame » compte 8 bits de contrôle représentant 256 possibilités de configuration

C’est dans cette optique que Bratus et al. [24] explorent toutes les 256 combinaisons (cf. Figure 2.2) possibles de caractéristiques dérivées des sous-champs du champ "contrôle de trame (c.-à-d. Frame Control)" de l'en-tête de la sous-couche de contrôle d'accès au support.

Un exemple de test de stimulus est d'envoyer une trame de requête de sonde contenant le bit FromDS ou ToDs fixé à 1 alors même que ces deux bits sont fixés à 0.

Pour générer et injecter des trames non standard et mal formées sur des périphérique cible, les auteurs ont développé un outil appelé BAFFLE. Les réactions aux stimulus forment alors des caractéristiques pour le fingerprinting de périphérique sans fil. La méthode est utilisée pour identifier un point d'accès ou une station client.

2.5 Fingerprinting sur base de l'en-tête de paquet TCP

Miettinen et al. [26] proposent un système nommé « IOT Sentinel » qui identifie les types de dispositifs IoT et exécute un plan d'action approprié pour restreindre ou autoriser leurs communications au sein d'un réseau. De sorte que, des éventuels dispositifs, vecteurs d'attaque, ne soient utilisés pour compromettre l'ensemble du réseau.

Le système s'appuie sur l'algorithme de classification "Random Forest" pour identifier le type de dispositif. Selon les auteurs, deux dispositifs sont dits du même type s'ils partagent le même modèle et la même version de logiciel.

Lorsqu'un nouveau dispositif est introduit dans le réseau pour la première fois, qu'une nouvelle adresse MAC est découverte, que ce dernier entame sa phase d'installation et de configuration (premiers instants de communication avec la passerelle), dans ce cas, le système engage un processus de capture de paquets à l'aide de tcpdump avec un filtrage par adresse MAC du nouveau dispositif.
En sortie pour chaque paquet capturé, 23 caractéristiques sont extraites à plusieurs niveaux du modèle OSI, le résultat est un ensemble de vecteurs de caractéristiques de dimension 23 X 12 paquets = 276. Ces dernières sont résumées dans le tableau 2.1

Sur base des analyses préliminaires, les auteurs ont fixé le nombre de vecteurs à 12, le jugeant de taille suffisamment long pour discriminer les types de dispositifs et suffisamment court pour que des paquets uniques le remplissent. Si, d'aventure, la capture d'un dispositif ne générait pas suffisamment de paquets pour le volume requis, le vecteur est complété avec des zéros. Il est à noter que les paquets consécutifs comportant des champs identiques en sont exclus.

Couche du modèle OSI	**Caractéristique**
Couche liaison (2)	ARP / LLC
Couche réseau (4)	IP / ICMP / ICMPv6 / EAPoL
Couche transport (2)	TCP / UDP
Couche application (8)	HTTP / HTTPS / DHCP / BOOTP / SSDP / DNS / MDNS / NTP
Couche réseau :IP options (2)	Padding / RouterAlert
Couche réseau : contenu du paquet (2)	Taille paquet / contenu
Couche réseau : adresse IP (1)	Nombre d'adresse IP destination
Couche réseau: ports (2)	Port source / destination

Tableau 2.2: Liste des caractéristiques extraites pour l'identification d'un dispositif.

L’ensemble de vecteurs de caractéristiques, F′, servent de base pour entraîner l'algorithme de classification dans le processus d’identification des dispositifs et ce, en suivant une approche à deux volets.

Une première opération consiste à construire un classifieur multi-classe par dispositif selon une stratégie "un contre tous (One-vs-All) [27].”
Elle consiste à entraîner C classifieurs binaires (C étant le nombre de dispositifs à disposition). Le c-ième de ces classifieurs utilise toutes les observations de la classe c comme exemple positifs, et toutes les autres comme exemple négatifs. Ainsi, chaque classifieur apprend à distinguer une classe de toutes les autres. Le classement est donné par le classifieur avec le plus grand score de probabilité.

Les auteurs fixent un seuil de discrimination à 0.2 au-delà duquel, une prédiction sur F′ par un des classifieurs est considérée acceptable et rejetée dans le cas contraire.
Dans le cas, où plusieurs classifieurs prédisent un résultat positif, le système entame une deuxième opération consistant à discriminer définitivement les candidats. Pour ce faire, le système utilise un algorithme basé sur la distance de *Damerau–Levenshtein* pour décider de la classe.

Un jeu de données de 540 éléments issues de la capture de 27 types de dispositifs est utilisé. La méthode croisée "10-fold cross-validation" avec k=10 a été adoptée pour évaluer les performances des modèles construits. L’algorithme de classification obtient un taux de classification de 95% pour 17 dispositifs et 5% pour le reste. La moyenne étant de 81%.

Bien que le système IoT Sentinel n'ait besoin en moyenne que de 21 paquets pour procéder au fingerprinting, il n’opère que dans un cadre très limité, lorsqu’il un dispositif est introduit pour la première fois dans le réseau.

Afin de remédier à cette lacune, B. Bezawada et al. [28] proposent un système complémentaire nommé « IotSense » qui procède à l’identification comportementale des dispositifs IoT à partir de leur activité au sein du réseau.
Chaque dispositif se voit attribué un profile comportemental, de sorte à détecter d’éventuelles déviations au comportement initial du dispositif, en raison d’activités malveillantes par exemple.

En vue de caractériser les différents comportements normaux des dispositifs, les auteurs ont capturé entre 1000 et 10000 paquets émanant du trafic réseau de 14 dispositifs IoT à différents moments clés représentatifs des états d’un dispositif sur une période donnée. En premier lieu, lors de sa configuration initiale. Deuxièmement, lorsqu’il est à l’état stable, les auteurs ont interagi avec via son application. Finalement, durant sa période d’inactivité sans l'intervention d'une tierce partie.

En analysant les en-têtes Ethernet, IP et de transport, 20 caractéristiques sont extraites par intervalles de 5 paquets consécutifs. Le résultat est un ensemble de vecteurs de caractéristiques de dimension 20 X 5 paquets = 100. Cet ensemble est ensuite utilisé pour entraîner un modèle par la classification supervisée pouvant être utilisé pour identifier le type d’un dispositif ou sa catégorie. Les dispositifs sont classés dans 9 catégories selon leurs similitudes. Typiquement, deux caméras connectées issues de deux modèles/vendeurs différents seront regroupées sous la catégorie caméra.

Les auteurs ont utilisé les classifieurs k-nearest-neighbors, Decision trees, Gradient boosting et Majority voting depuis l'outil *Scikit-learn* [29].
Lors de la formation d'un classificateur pour un type de dispositif, la stratégie utilisée est "un contre tous (One-vs-All)" [27] : le principe consiste à transformer le problème à k classes en k classifieurs binaires. Le classement est donné par le classifieur avec le score le plus élevé.

Par ailleurs, la méthode de la validation croisée "5-fold cross-validation" a été utilisée pour l'évaluation. Dans cette méthode, les données étiquetées sont divisées en k parties égales (généralement 5 ou 10), et k modèles différents sont entraînés : chaque modèle « met de côté » une partie différente et s'entraîne sur les k-1 parties restantes [30].
L'expérience pour l'identification du type de dispositif est évalué à taux moyen compris entre 93% et 99% et une précision moyenne de de 99%. Tandis que, l'expérience visant à identifier un dispositif selon sa catégorie est évaluée quant à elle à un taux moyen de variant entre 99.7% et 100%

2.6 Fingerprinting sur base de statistiques du flux de trafic réseau

La problématique de l'utilisation de l'analyse des données du trafic réseau a déjà fait l'objet de nombreuses études [31] principalement dans un contexte de sécurité des systèmes informatiques. L'idée étant que le trafic généré par chaque classe d'applications possède une propriété statistique unique et les techniques d'apprentissage automatique conviennent assez bien à l'exploration de tels patterns.
Un exemple de ce type est une étude décrivant un système supervisé permettant de détecter de logiciels malveillants inconnus en utilisant la classification du trafic réseau [32].

L'étude présente une approche de l'analyse de données basée sur 4 niveaux d'observations de trafics réseau :

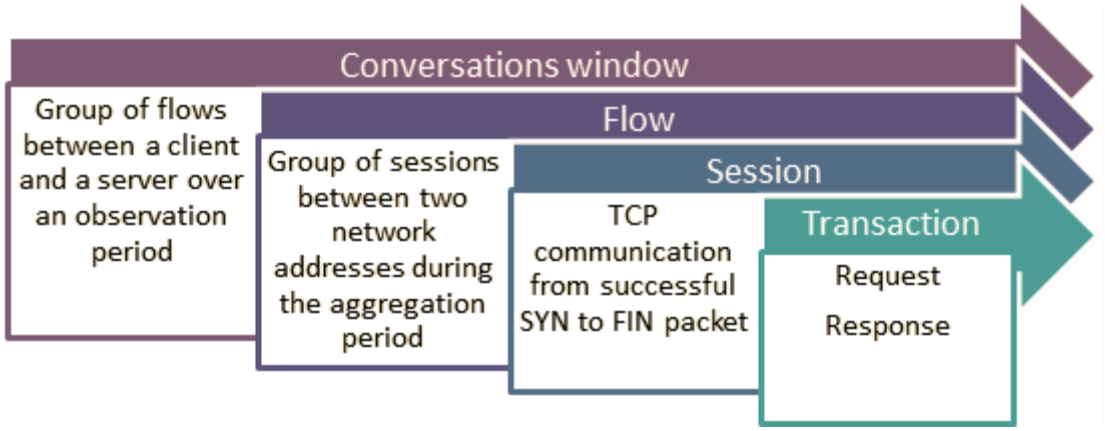

FIGURE 2.1: Types de flux de données encapsulés [33]

- Transaction - Représentant d'une interaction entre un client et un serveur. Il s'agit d'une communication à double sens : le client envoie une demande au serveur et le serveur traite la demande et renvoie une réponse au client.

- Session - Un quadruplet unique composé d'adresses IP sources et de destination et de numéros de port.
  Une session TCP commence par une négociation réussie et se termine par un délai d'attente ou un paquet avec l'indicateur RST ou FIN de l'un des périphériques.

Une session UDP consiste en tous les paquets envoyés d'un client à un serveur et d'un serveur à un client jusqu'à ce qu'une durée d'inactivité de communication définie soit atteinte.

- Flux - Un groupe de sessions entre deux adresses réseau (paire IP) au cours de la période d'agrégation. Un nouveau flux commence si le délai entre la fin d'une session (le dernier paquet) et le début d'une nouvelle session (premier paquet) est supérieur au délai d'inactivité défini. La nouvelle session fait alors partie du nouveau flux.

- Fenêtres de conversation - Groupe de flux entre un client et un serveur sur une période d'observation. Une conversation peut être définie entre deux adresses réseau (paire IP) ou un groupe de ressources réseau (par exemple entre deux systèmes autonomes).

Les auteurs implémentent par ailleurs un outil d'extraction de caractéristiques comportementales du trafic réseau basé sur ces 4 niveaux d'observations décrits plus haut. L'outil reçoit en entrée des fichiers de captures réseaux au format PCAP pour produire en sortie un ensemble de vecteurs de caractéristiques au format de fichiers CSV.

En tout 972 [33] caractéristiques sont extraites à travers différents protocoles et couches réseau. Le tableau 2.2 présente quelques exemples de caractéristiques tirées de différents protocoles et couches du trafic réseau.

Niveau d'observation	Protocole	Caractéristiques
Transaction	HTTP	Nom d'hôte Référent Cookie User Agent Type de contenu
	SSL	Nom du serveur Version SSL Date d'expiration du certificat
	DNS	Query name Alexa 1M rank Nombre de noms canoniques Drapeaux de réponseTTL
Session	TCP	Port de destination Taille de paquet Nombre de paquets avec le drapeau PUSH Nombre de paquets hors usage
Flux	TCP	Nombre de paquets keep-alive dans un flux Temps inter-arrivé Nombre de port réutilisant des paquets
	IP	Destination IP Géolocalisation IP Numéro de système autonome IP
Fenêtres de conversation	UDP	Rapport entre les paquets envoyés et reçus
	DNS	Nombre de réponses de domaine inexistantes
		Nombre de sessions en flux Quantité totale de données transmises

Tableau 2.3 : Quelques exemples de caractéristiques tirées de différents protocoles et couches du trafic réseau, tandis que la liste complète est donnée dans [33].

Meidan et al. [34] sont les premiers à démontrer la faisabilité d'identifier des dispositifs IoT sur base de traces réseaux en utilisant l'apprentissage automatique. Les auteurs présentent « ProfilIot » un système qui, dans une première étape, analyse les sessions TCP pour distinguer le trafic réseau généré par les objets non-IoT et IoT et, dans une seconde étape, procède à leur identification.

Leur travail décrit un environnement expérimental dans lequel des données de trafic réseau ont été collectées à partir de 13 périphériques différents (dont 9 étaient des dispositifs IoT) connectés à un point d'accès Wi-Fi.

La capture de trafic a été enregistrée sous forme de paquets dans des fichiers pcap[5] pour une analyse ultérieure. Le but étant de déterminer si ce trafic appartient à un ordinateur personnel, un smartphone ou un dispositifs IoT (connu au sein du réseau).

Les données collectées sont ensuite transformées en utilisant un outil d'extracteur de caractéristiques [32], qui accepte un fichier pcap en entrée et fournit en sortie, un ensemble de caractéristiques sous forme de sessions TCP. Une session est identifiée par un quadruplet unique (adresse source, port source, adresse cible, port cible).

À l'aide d'un apprentissage supervisé, ils ont formé un méta-classificateur qui prédit la probabilité qu'une session donnée provienne d'un dispositif appartenant à l'ensemble des dispositifs IoT connus. Suite à la définition d'un seuil, le classificateur était également en mesure de prédire si cette session était issue d'un dispositif IoT ou non.

L'algorithme de classification de dispositifs IoT obtient un taux de classification de plus de 99%.

Dans une autre étude [35], un mécanisme similaire a été utilisé pour la détection de dispositifs IoT non autorisés dans le réseau à l'aide de techniques d'apprentissage automatique. La méthode proposée s'inscrit dans un contexte d'application de règles de sécurité concernant les types de dispositifs IoT autorisée à se connecter au réseau.

Pour chaque flux de données de trafic réseau provenant d'un dispositif, le défi consiste à identifier son type et ainsi déterminer s'il fait partie de la liste blanche autorisé à communiquer.

Sur une période de plusieurs mois, les auteurs ont capturé et enregistré au format PCAP le trafic réseau provenant de 17 dispositifs IoT distincts. Ensuite, ils ont utilisé l'outil extracteur de caractéristiques * pour transformer chaque session TCP en un vecteur de caractéristiques. En tout, plus de 300 caractéristiques ont été utilisées pour entraîner et évaluer un classifieur multi-classe par type de dispositif.
Le modèle d'apprentissage automatique utilisé dans cette étude est le modèle Random Forest.

Dans un premier temps, les résultats présentent en moyenne un taux de 94% de sessions correctement classés comme inconnues et un taux de 97% de sessions correctement classifiées comme faisant parties de la liste blanche.

Ensuite pour améliorer ces résultats, les auteurs ont mis en œuvre une étape supplémentaire dans le processus de classification utilisant le vote à la majorité sur une séquence de 20 sessions consécutives déjà classifiées. Le résultat montre une amélioration de l'ordre de 2%.

[5] pcap (« packet capture ») est une interface de programmation permettant de capturer un trafic réseau. https://www.tcpdump.org/

2.7 Caractéristiques du trafic des dispositifs IoT

Comprendre la nature et les caractéristiques du trafic généré par les dispositifs IoT est une étape cruciale pour la mise en œuvre d'une gestion efficace de la politique réseau et des ressources dans une infrastructure IoT.

Cependant, des études se concentrant exclusivement sur la caractérisation du trafic IoT en sont encore à leurs balbutiements.

Défi auquel Sivanathan et al [36] ont tenté de répondre en analysant de manière empirique le trafic réseau dans des conditions simulant un environnement de ville intelligente et de campus intelligent afin de dégager les caractéristiques et patterns comportementaux des dispositifs IoT.

Pour ce faire, ils ont collecté le trafic réseau de toute une gamme hétérogène de dispositifs (30), aussi bien IoT (28) que non-IoT (2), pendant une période continue s'étalant sur plusieurs mois. Le trafic IoT comprend aussi bien le trafic généré par les dispositifs de manière autonome que le trafic généré suite aux interactions des utilisateurs avec les dispositifs.

Les données brutes collectées se composent des informations d'en-tête et de charge utile de données de paquet TCP.

Les auteurs s'intéressent en premier lieu à la distribution de 4 caractéristiques de flux trafic : le débit, durée, ratio et la durée d'inactivité des flux trafic. Il est expliqué que pour chacune des caractéristiques on trouve des disparités qui exhibent un pattern distinct.

Puis, Sivanathan et al [36] expliquent que les dispositifs IoT utilisent chacun moins de 10 ports distincts pour communiquer, que certains dispositifs utilisent des numéros de port non standard. En outre, certains d'entre eux émanant du même fabricant partagent certains numéros de port. De même que, en matière de requêtes DNS, certains noms de domaine sont invoqués par des dispositifs du même constructeur.

Les auteurs pointent également qu'en ce qui concerne le protocole NTP, certains dispositifs présentent un pattern identifiable au niveau d'intervalle d'envoi de requête NTP.

Enfin, ils remarquent que 17 sur les 28 dispositifs IoT du banc d'essai, utilisent TLS/SSL pour communiquer. En outre, au niveau de la liste des suites cryptographiques [37] émis lors de l'établissement d'une connexion TLS.

Par ailleurs, les auteurs se sont appuyés sur ces patterns pour mettre en place un algorithme multi-étapes basé sur l'apprentissage automatique utilisant des combinaisons de ces caractéristiques pour aider à classer les dispositifs IoT avec haute précision de l'ordre de 99%.

2.8 Résumé

Les techniques abordées dans ce chapitre sont mises en œuvre en se basant sur les méthodes actives ou passives le fingerprinting. Dans les méthodes actives, un stimulus est envoyé à la cible pour ensuite analyser les réponses ou absence de réponse.
A l'inverse, les méthodes passives poursuivent le même but mais n'introduisent pas de trafic supplémentaire.

A cet égard, dans le présent travail, nous avons choisi la méthode passive pour procéder au fingerprinting des dispositifs IoT. De par l'hétérogénéité des logiciels et du matériel de ces derniers, la méthode active nous semble moins opportune parce que souvent elle requiert la collaboration de la cible visée en plus d'interférer avec le trafic légitime.

[26, 28, 34, 35, 36] traitent de fingerprinting des dispositifs IoT tout comme le présent travail. Un tableau comparatif des techniques passives de fingerprinting des dispositifs IoT est présenté dans le tableau 2.3.

Malgré le fait que la précision moyenne de prédiction soit de 81% chez Miettinen et al. [26], pour 1/3 des dispositifs cette performance baisse pour avoisiner une précision de 50% seulement. Cette contre-performance est due à une confusion possible entre dispositifs du même fournisseur. De plus, cette étude ne se limite exclusivement que sur l'étape de la configuration du dispositif, au moment de son introduction dans le réseau.

	Miettinen et al. [26]	Bezawada et al. [28]	Meidan et al. [34] (ProfilIot)	Meidan et al. [35]	Sivanathan et al [36]
Nombre de caractéristiques	23	20	-	> 300	12
Données brutes disponibles publiquement	Oui	Non	Non	Non	Oui
caractéristiques sur base d'en-tête paquet (T) / sur base de flux de trafic réseau (F)	T	T	F	F	F
Nombre de dispositifs testés	IoT(27)	14(IoT)	IoT (9) Non-IoT(4)	IoT (17)	IoT (28)
Précision de prédiction (%)	81	99	99	99	99

Tableau 2.4 : Comparaison des techniques de fingerprinting de dispositif IoT de l'état de l'art

Meidan et al. [34] (ProfilIot) affiche une précision de 99% dans leur résultat, cependant leur expérience ne porte que sur 9 dispositifs IoT. Par ailleurs, ils utilisent un peu plus de 300 caractéristiques pour entraîner leur modèle de classification, ce qui peut représenter un coût important en temps d'apprentissage des algorithmes utilisés.

Chapitre 3 : Méthodologie

Dans ce chapitre, nous décrivons l'environnement et la procédure mis en place pour valider la méthode de fingerprinting des dispositifs IoT proposée dans ce travail.

3.1 Labo

Nous avons déployé une infrastructure d'objets connectés afin de simuler un environnement domotique (cf. la figure 4.1) composé de cinq dispositifs IoT :

- une Caméra TP-Link NC200 ;
- un Capteur d'ouverture Domos Wi-Fi ;
- une prise connectée TP-Link HS100 Wi-Fi ;
- une prise connectée Chacon ;
- et un pont Philips Hue associé à deux ampoules Philips Hue.

Les 5 dispositifs se connectent à Internet grâce à une box Raspberry Pi 3 b+ [2] dans laquelle nous avons installé OpenWrt [3] pour s'en servir comme routeur. Les caractéristiques détaillées du Raspberry Pi 3 B+ peuvent être trouvées dans le tableau 3.1.

L'avantage d'OpenWrt, au-delà de fournir les fonctionnalités attendues d'un routeur grand public (gestion du Wi-Fi, DHCP, etc.), est de proposer une interface Web "LuCI" [38] à partir de laquelle il est possible d'à peu près tout configurer et administrer. La figure 3.2 illustre un aperçu des interfaces Web.

De plus, OpenWrt [3] offre un gestionnaire de paquets opkg [39] pour installer ou maintenir des logiciels disponibles à partir de ses dépôts embarqués et permet ainsi d'étendre ses fonctionnalités. A cet égard, nous avons installé quelques paquets opkg [39], à savoir tcpdump pour visualiser/capturer le trafic de notre réseau domotique, block-mount et kmod-usb-core pour le montage et la prise en charge du périphérique de stockage USB où seront sauvegardées les traces de captures.

Enfin, OpenWrt est livré avec DnsMasq [40], un petit serveur DNS qui intègre un serveur DHCP très léger et simple à configurer, rendant aisé l'assignation d'adresses IP internes. Cette configuration nous a permis d'automatiser la collecte de trafic réseau en provenance et à destination des dispositifs IoT connectés au réseau du Raspberry Pi. Les sous-sections suivantes décrivent le déroulement de la capture de paquets et leur traitement.

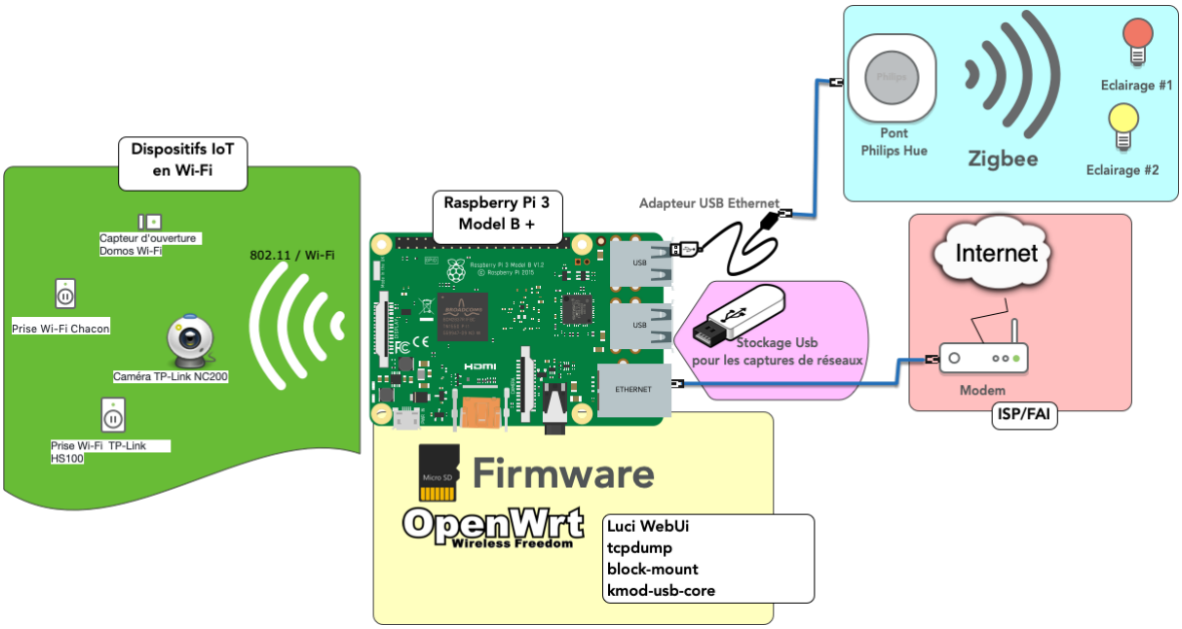

Figure 3.1 : Architecture de l'environnement domotique du labo

Processeur		**Mémoire**	
Marque du processeur	Broadcom	Mémoire vive (RAM)	1 GB
Nom de code processeur	Modèle B+	Mémoire extensible	Non
Nombre de noyaux du processeur	Quad Core (4)	Nombre de canaux de mémoire	Canal unique (1)
Fréquence d'horloge	1,4 GHz	Type de mémoire	DDR2
Vitesse turbo	1,4 GHz	Connexion sans fil	
Mémoire cache	512 kB	Standard Wi-Fi/Bluetooth versio 4.2	Wireless AC, Wi-Fi G, Wireless N
Connexions filaires			
Connexion casque audio Oui			
Connexion casque audio	mini-jack 3,5 mm		
Nombre de connexions casque audio	1		
Nombre de ports TRS Audio Jack 3,5 mm femelles	1		
Version port Micro USB Type-B (forme 2.0) femelle	2.0		
Nombre de ports Micro USB Type-B (forme 2.0) femelles	1		
Port USB Type-A standard version femelle	3.0		
Nombre de ports USB Type-A standard Femelle	4		
Nombre de ports HDMI	1		
Version femelle port HDMI Type A	1.4		

Nombre de ports HDMI type-A 1.4b femelles	1	
Nombre de ports Ethernet	1	
Débit Ethernet	Gigabit Ethernet (1000 Mbps)	

Tableau 3.1 : Caractéristiques matérielles du Raspberry Pi 3 B+

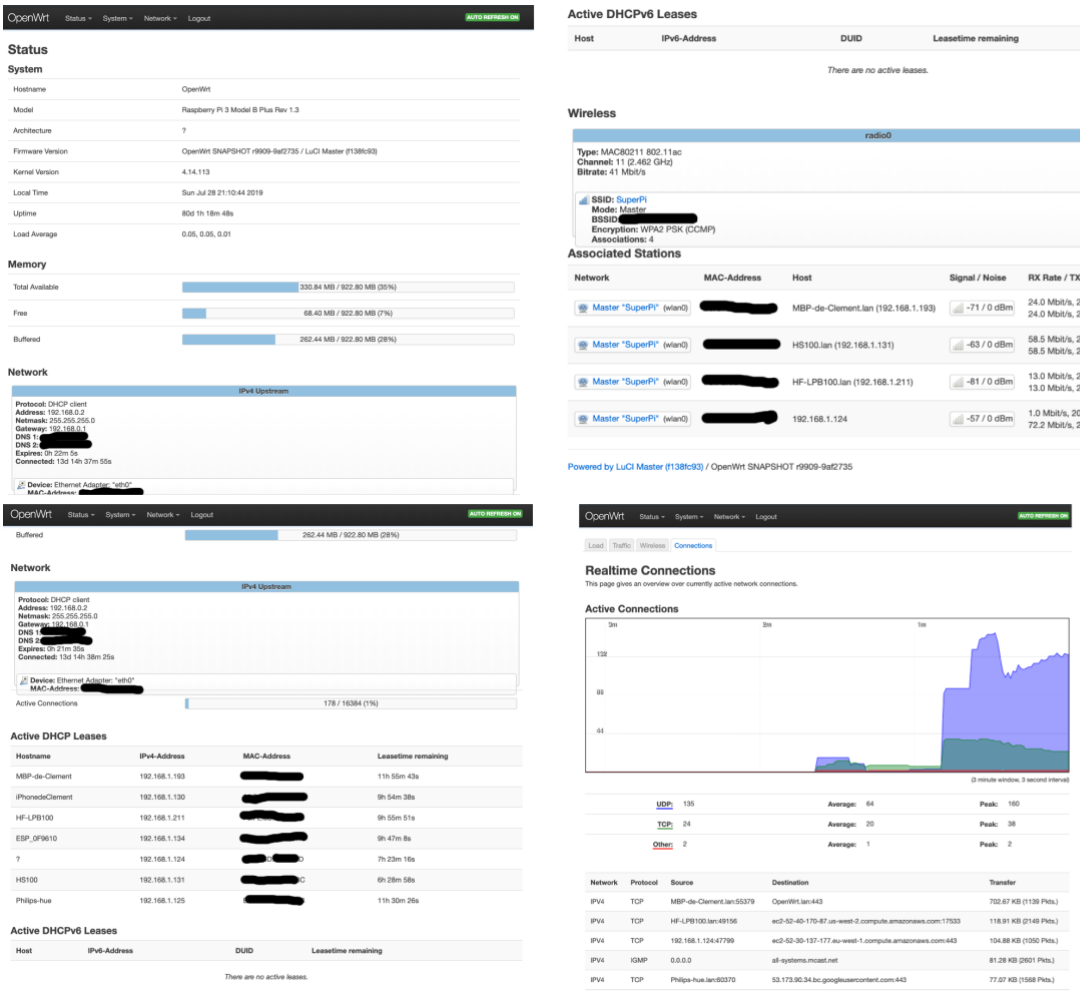

Figure 3.2 : Aperçu des interfaces Web LuCI

3.2 Collecte des données

La capture du trafic réseau est un processus relativement aisé qui peut être réalisé en plaçant un outil tel que tcpdump ou t-shark sur un hôte par lequel le trafic sur le réseau est routé. Dans notre cas, tout le trafic réseau entrant et sortant du réseau local a été collecté à l'aide de l'outil tcpdump s'exécutant sur OpenWrt [3].

Cependant, avant de mettre en place un système automatisé basé sur OpenWrt, nous avons entamé une phase exploratoire dans la mesure où, comme mentionné plus haut, les dispositifs IoT présentent une hétérogénéité aussi bien fonctionnelle (statique ou mobile, batterie ou sous tension, etc.) que technique (logiciels, réseaux, caractéristiques physiques, etc.), si bien que cette hétérogénéité s'observe également au niveau de l' activité réseau.
Pour cette raison, nous avons dans un premier temps observé et collecté (« manuellement ») le trafic réseau de chaque dispositif IoT séparément à l'aide de l'analyseur de paquet Wireshark.

Durant cette phase d'observation, toutes les traces ont été collectées pendant une durée moyenne de 4 à 5 minutes depuis un ordinateur (MacOs « Mojave ») connecté au même réseau que le dispositif. La figure 3.3 décrit la configuration réseau mise en place pour permettre d'observer le trafic réseau transitant entre le pont Philips Hue et L'extérieur. Tandis que la figure 3.4 montre une configuration pour surveiller les dispositifs nécessitant du Wi-Fi.

L'étude des données a permis d'établir les constatations suivantes :

- Tous les dispositifs, excepté le capteur d'ouverture de porte, génèrent du trafic fréquemment à intervalle régulier quand il n'y a aucune interaction (humaine). Par exemple, les requêtes de type NTP ;

- La distribution de volume de paquets par dispositif IoT présente généralement des variations de la même ampleur quand il n'y pas d'interactions avec les tiers. Et par ordre décroissant d'importance, le volume se répartit comme suit : le pont Philips Hue, la prise Chacon, la caméra Tp-Link et la prise Tp-Link ;

- Toutefois, lorsque la caméra réagit à un événement, par exemple en mode détection de mouvements, dès lors son activité réseau est bien plus conséquent ;

- Le capteur d'ouverture de porte ne génère du trafic que pour notifier de la séparation de son élément magnétique.

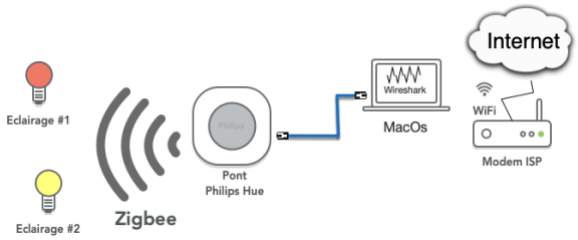

Figure 3.3 : Dans cette topologie réseau, le pont Hue communique avec le monde extérieur grâce à l'ordinateur qui joue le rôle de routeur wifi. De ce fait, tout le trafic réseau des échanges qui s'opèrent entre le pont Hue et le monde extérieur sur l'interface Wi-Fi peut être capturé par l'outil Wireshark

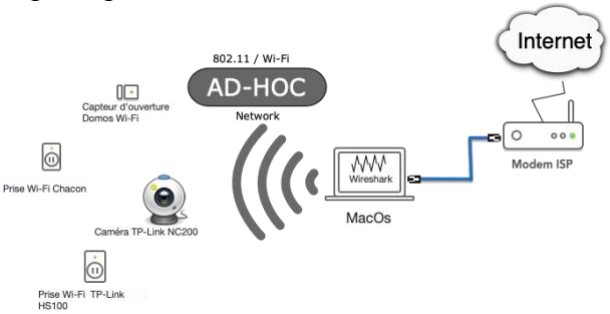

Figure 3.4 : L'ordinateur est configuré pour partager la connexion Internet du câble Ethernet via un réseau Wi-Fi ad hoc.

Ces constatations font échos aux travaux de Sivanathan et al [36] que nous avons évoqués plus haut. La figure 3.5 illustre la distribution de paquets des dispositifs IoT du labo prise sur une durée d'une heure et 5 minutes. On peut notamment remarquer l'absence d'activité réseau en ce qui concerne le capteur d'ouverture de porte. En outre, sans interaction aucune avec les

dispositifs, le pont Hue génère plus de trafic que la caméra Tp-Link. Cependant, si on interagit avec ce dernier, alors son activité réseau est multiplié par un facteur de 5 par rapport au reste (cf. figure 3.5 graphiques b, b', d et d').

Après la phase d'exploration, nous avons mis en place un labo qui simule un environnement domotique que nous avons présenté en introduction de ce chapitre. Ce labo poursuit une double finalité :
1) Collecter de façon passive de traces réseaux de dispositifs IoT en temps réel ;
2) Créer un environnement semblable à celui d'une maison intelligente.

Pour ce faire, nous avons collecté les traces du 12 mai au 28 Juillet grâce une tâche cron sur OpenWrt exécutant un script tcpdump toutes les deux heures et 5 minutes et ce une heure durant. Le script filtre le trafic capté sur base des adresses MAC des dispositifs IoT de sorte à exclure d'autres paquets (ordinateur ou smartphone) transmis sur le réseau.

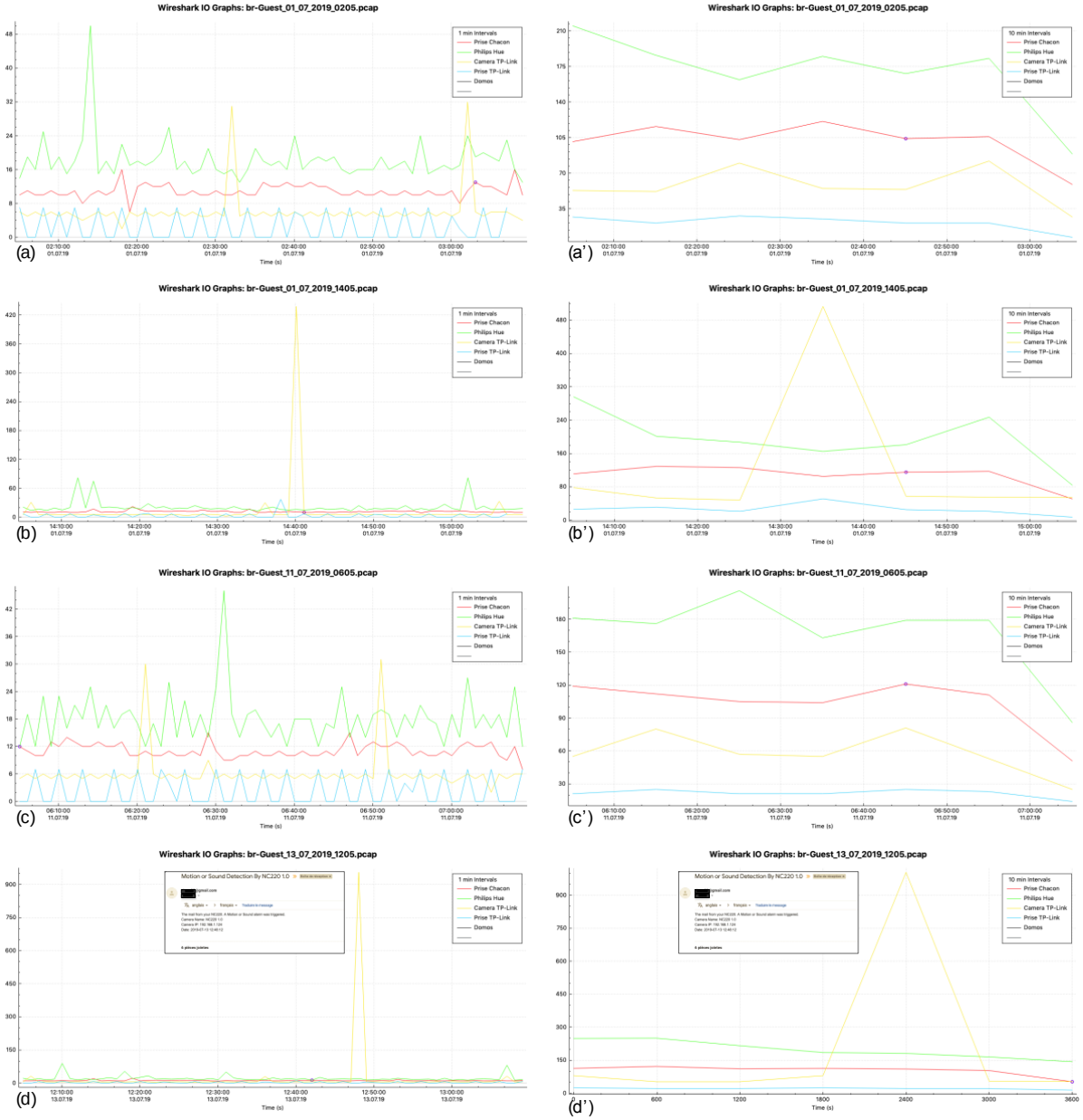

Figure 3.5 : Distribution de paquets des dispositifs IoT. A gauche par intervalle d'une 1 minute (a, b, c, d) et à droite par intervalle de 10 minutes (a', b', c', d')

Notons que pour collecter des paquets issues d'une interaction humaine, nous avons mis en places des moyens détournés :

- les deux prises Wi-Fi ont été configurées en mode « Simulation de présence » de sorte à s'éteindre /s'allumer de manière aléatoire dans des certaines tranches horaires ;

- la caméra a été configurée en mode «détecteur de son » de sorte à envoyer une notification par mail;

- le capteur d'ouverture de porte est moins élaboré et ne génère que très peu de trafic. Pour cette raison, les seules traces collectées ont été provoquées manuellement ;

- deux ampoules Philips ont installés dans des pièces de vie, de sorte que les paquets générés par le pont Hue incluent ceux issues de l'interaction avec les ampoules.

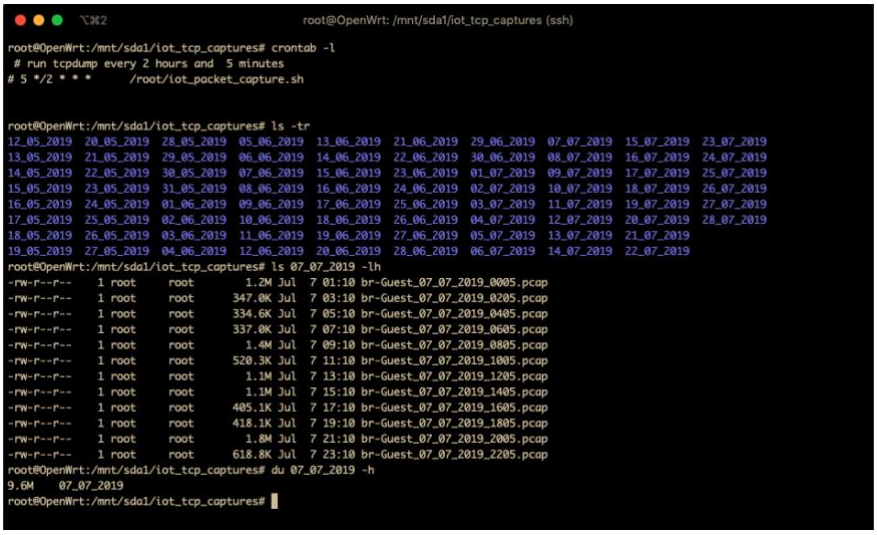

Figure 3.6 : Par jour des paquets sont capturés et sauvegardés au format PCAP dans un dossier qui a pour nom la date du jour. Chaque dossier en compte une douzaine.

Les traces résultantes ont été stockées sous forme de fichiers pcap sur un disque externe USB monté sur le Raspberry Pi, soit 590 mégaoctets de données brutes utilisées par la suite pour le traitement et l'extraction des caractéristiques que nous développons dans la section suivante.

3.3 Traitement de données

Dans la section précédente, nous avons décrit le processus de collecte de traces. Une fois que nous avons toutes les traces, nous devons les convertir dans un format exploitable par les algorithmes d'apprentissage automatique.
Pour ce faire, un script python a été implémenté pour permettre l'extraction de caractéristiques en prenant en entrée un fichier pcap et en produisant un fichier « .txt » représentant un jeu de données. La dissection de paquet réseau est assurée par Scapy [41], une librairie python permettant entre autres la manipulation et l'exploration de paquet TCP/IP.

La méthode proposée pour le fingerprinting de dispositifs IoT se base sur les caractéristiques statistiques sur les flux sortants. Un flux de réseau peut être défini comme un ou plusieurs paquets voyageant entre deux adresses d'ordinateur en utilisant un protocole particulier (TCP,

UDP, ICMP, ...) et, le cas échéant, une paire de ports particulière (définie pour chaque extrémité du flux) [42]. Presque toutes les « connexions » réseau contiennent deux flux minimum. Un flux de client à serveur et un autre de serveur à client.

Dans le présent travail, un flux de réseau est défini comme un groupe de séquences de paquets voyageant entre deux hôtes et partageant le même 5-tuple<Adresse IP source, Adresse IP destination, Port Source, Port Destination, Protocole> et ce, dans un intervalle de temps donné. A noter que seuls les protocoles TCP et UDP sont considérés. Tel qu'illustré à la Figure 3.7, un flux est délimité par le 5-tuple cité plus haut et est compris dans un intervalle prédéfinie sinon il s'agit d'un nouveau flux.
Sur base des flux, 47 caractéristiques sont extraites. Ce processus se déroule en même temps que la constitution d'un flux lors du traitement du fichier pcap. La liste complète des caractéristiques est reportée dans le tableau 3.1.

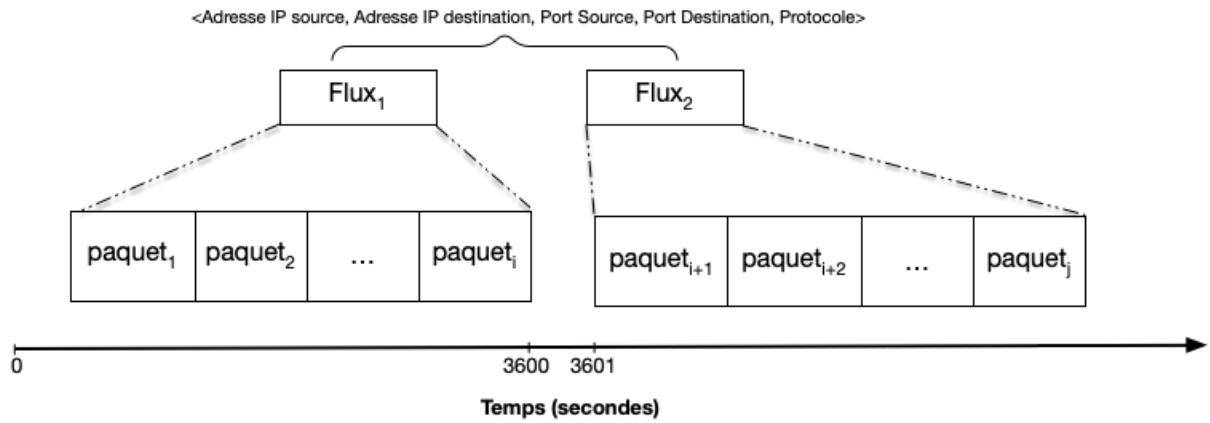

Figure 3.7 : Représentation de flux

Caractéristique	Mesure	Description
label	-	Etiquète
prot	-	int protocole 17:UDP 6:TCP
push_flag	-	Nombre de flags PUSH
ack_flag_nodata	-	Nombre de paquets keep-alive
flow_duration	-	Durée du flux
avg_bits_per_sec	-	Débit moyen
avg_window	moyenne	Taille de la fenêtre
avg_arr_time	moyenne	Temps inter-arrivée : Soit T la durée de temps qui sépare deux arrivées de paquets consécutives
std_arr_time	-	Temps inter-arrivée
min_arr_time	minimum	Temps interarrivée
max_arr_time	maximum	Temps interarrivée
avg_pkt_len	moyenne	Taille du paquet
std_pkt_len	L'écart-type	Taille du paquet
min_bytes	minimum	Taille du paquet
max_bytes	maximum	Taille du paquet

tot_pkt	-	Nombre de paquets
tot_bytes	somme	Taille du paquet
ssdp	-	Nombre de paquets de type SSDP (Simple Service Discovery Protocol)
https	-	nombre de paquets de type https
http	-	nombre de paquets de type http
dns	-	nombre de paquets de type DNS
avg_dns_response_ttl	moyenne	DNS time-to-live
std_dns_response_ttl	L'écart-type	DNS time-to-live
min_dns_response_ttl	minimum	DNS time-to-live
max_dns_response_ttl	maximum	DNS time-to-live
avg_dns_response_answers_rrs	moyenne	DNS Resource records (RRS)
std_dns_response_answers_rrs	L'écart-type	DNS Resource records (RRS)
min_dns_response_answers_rrs	minimum	DNS Resource records (RRS)
max_dns_response_answers_rrs	maximum	DNS Resource records (RRS)
avg_dns_response_additional_rrs	moyenne	DNS Addtional Record(s)
std_dns_response_additional_rrs	L'écart-type	DNS Addtional Record(s)
min_dns_response_additional_rrs	minimum	DNS Addtional Record(s)
max_dns_response_additional_rrs	maximum	DNS Addtional Record(s)
avg_dns_response_authority_rrs	moyenne	DNS authoritative nameservers
std_dns_response_authority_rrs	L'écart-type	DNS authoritative nameservers
min_dns_response_authority_rrs	minimum	DNS authoritative nameservers
max_dns_response_authority_rrs	maximum	DNS authoritative nameservers
avg_dns_response_type_a	moyenne	Resource records TYPE A
std_dns_response_type_a	L'écart-type	Resource records TYPE A
min_dns_response_type_a	minimum	Resource records TYPE A
max_dns_response_type_a	maximum	Resource records TYPE A
avg_dns_response_type_cname	moyenne	Resource records TYPE CNAME (canonical name)
std_dns_response_type_cname	L'écart-type	Resource records TYPE CNAME (canonical name)
min_dns_response_type_cname	minimum	Resource records TYPE CNAME (canonical name)
max_dns_response_type_cname	maximum	Resource records TYPE CNAME (canonical name)

Tableau 3.1 : Caractéristiques de flux utilisées pour le fingerprinting des dispositifs IoT

Les 47 caractéristiques adoptées dans un premier temps sont le résultat d'un processus de recherche visant à sélectionner les variables les plus pertinentes pour discriminer les flux issus du trafic des différents dispositifs IoT.
Certaines de ces caractéristiques sont courantes comme les adresses sources/destinations, les protocoles, les ports, et les protocoles. D'autres, basées sur des statistiques sommaires de flux sont inspirées de la littérature [43] [34].
Dans la section suivante nous allons discuter de la sélection de caractéristiques effectuée dans le présent travail.

3.4 Sélection de caractéristiques

La sélection est guidée intuitivement par le caractère intrinsèque des dispositifs IoT qui est basé sur la notion de la connectivité suggérant ainsi que le trafic émis par ces derniers peut renfermer des caractéristiques discriminantes. Typiquement, la plupart des dispositifs IoT échangent régulièrement du trafic avec des serveurs bien souvent identifiables par leurs noms de domaine correspondant à leurs fabricants/fournisseurs. De plus, ces échanges peuvent survenir de manière périodique comme, par exemple, l'usage du protocole NTP à destination des services d'horodatage, ou bien des requêtes DNS à l'initiative des dispositifs IoT. Beaucoup de dispositifs IoT présentent un schéma reconnaissable dans l'utilisation de certains protocoles TCP/IP [36].

[34], présenté dans l'état de l'art, introduit ProfilIot, un système analysant les sessions TCP pour distinguer, dans un premier temps, les dispositifs IoT et non-IoT et dans un second temps identifier le type de dispositif.
Cependant, leur technique a nécessité pas moins de 300 caractéristiques extraites du trafic réseau pour entraîner les algorithmes l'apprentissage automatique, ce qui suscite des interrogations quant à la mise à l'échelle d'une telle solution si un plus grand nombre de dispositifs ou un jeu de données plus conséquent devait être utilisé.
Néanmoins, notre sélection caractéristique est en parti inspirée de la liste de 300 caractéristiques utilisées dans [34], plus spécifiquement les statistiques sommaires de flux sortants.

D'autres caractéristiques sont historiquement utilisées dans le fingerprinting du système d'exploitation, comme la taille de fenêtre TCP utilisée pour annoncer combien d'octets supplémentaires de données que l'expéditeur du paquet est prêt à accepter.
Dans le contexte des objets connectés, l'intuition derrière cette caractéristique est que la taille de fenêtre TCP dépend de la mémoire du dispositif IoT et la rapidité de son traitement. Bezawada et al. [28] démontrent que la petite taille des dispositifs IoT ou la contrainte en en ressources contribue à la variabilité de cette caractéristique. Pour exemple, les ampoules connectées ont tendance à avoir la taille de fenêtre plus petite, et à contrario, les caméras auront une taille de fenêtre plus variables et plus grandes.

Une fois la liste des caractéristiques adoptée, la question qui se pose ensuite est de savoir dans quelle mesure il serait possible de réduire la dimension de données.

3.5 Réduction des dimensions des données

La taille des données peut être mesurée selon deux dimensions : le nombre de caractéristiques p et le nombre d'observations n sous forme d'une matrice de dimension n x P.
Au terme du processus d'extractions de caractéristiques, notre jeu de sous forme de fichier « txt » comprend 46764 observations et 47 caractéristiques, soit 6.6 méga octets.

Les méthodes de réduction de la dimensionnalité sont généralement classées dans deux catégories [44] [45] :

- Une réduction basée sur une **sélection de caractéristiques** qui consiste à choisir un sous-ensemble optimal de caractéristiques pertinentes, à partir d'un ensemble de caractéristiques original, selon un critère de performance

- Une réduction basée sur une **transformation des données** qui consiste à remplacer l'ensemble initial des données par un nouvel ensemble réduit, construit à partir de l'ensemble initial de caractéristiques.

Dans la catégorie sélection de caractéristiques, on distingue trois méthodes principales : « filter », « wrapper » et « embedded ». Dans le présent travail, c'est cette dernière qui a été utilisée.
Les méthodes « embedded » (appelées aussi méthodes intégrées) intègrent directement la sélection dans le processus d'apprentissage. Elles combinent les qualités des méthodes filter et wrapper. Les techniques intégrées les plus emblématiques peut être trouvées dans les algorithmes de type SVM, dans les arbres de décisions, ou AdaBoost.

En pratique, l'utilitaire *sklearn.feature_selection.SelectFromModel* de la bibliothèque *Scikit-learn* [29] a été utilisé en conjonction avec l'algorithme Random Forest pour fournir un classement de l'importance de la sélection des caractéristiques. Le classement est donné dans le tableau 3.2.
Ce traitement a permis non seulement de réduire le nombre de caractéristiques, mais aussi de réduire l'espace de stockage ainsi que le temps nécessaire pour entraîner un modèle tout en conservant la précision, une des métriques d'évaluation des performances d'un modèle comme nous allons le voir dans la section suivante.

Caractéristique	**Importance (score)**
std_arr_time	0.174313
avg_pkt_len	0.160242
avg_bits_per_sec	0.115903
max_arr_time	0.106001
Prot	0.057308
min_arr_time	0.055174
Ssdp	0.046119
flow_duration	0.039683
min_bytes	0.034971
tot_pkt	0.031956
push_flag	0.029253
std_pkt_len	0.027312
max_bytes	0.024368
tot_bytes	0.017413
ack_flag_nodata	0.011619
avg_window	0.009712
avg_arr_time	0.007780

Tableau 3.2 : 17 Caractéristiques sélectionnées pour l'apprentissage automatique classée par ordre d'importance de poids

3. 6 Classification

Après l'étape d'extraction de caractéristiques sur base de fichiers PCAP et de leur transformation en jeu de données. Celui-ci a été ensuite traitée à l'aide de la librairie Scikit-learn pour élaborer des modèles capables de prédire/identifier le type de dispositif IoT grâce à la technique apprentissage automatique.

Il existe de multiples algorithmes de classification différents adaptés à un problème tel que celui-ci. Nombreux d'entre eux supportent intrinsèquement les données multiclasses (par exemple les arbres de décision, les voisins les plus proches, les bayésiens naïfs, ou encore la régression logistique multinomiale). Et pour d'autres comme le SVM qui ne supporte que deux classes par définition, il existe malgré tout plusieurs méthodes d'adaptation des SVM aux problèmes multiclasses [46].
Scikit-learn comprend un large éventail d'algorithmes d'apprentissage automatique supervisé et non supervisé. Dans le présent travail, six différents algorithmes de classification ont été utilisées : Random Forest, SVM, AdaBoost, XGBoost, Gaussian Naïve Bayes et Perceptron multicouche.

Pour ce faire, les algorithmes ont été exécutés dans une application Web appelé Jupyter Notebook [47] choisie pour son interface conviviale, sa facilité de rapidement présenter des graphiques dans des documents interactifs « cahiers », sa mise en page permettant de combiner à la fois du code python et du texte, etc. La Figure 3.8 schématise une vue d'ensemble de la méthodologie mise en place partant de la capture du trafic réseau jusqu'à l'élaboration de modèles de prédiction.

Comme susmentionné, l'approche proposée dans ce papier est basée sur l'apprentissage supervisée multiclasses dans le sens où nous traitons le fingerprinting des dispositifs IoT comme un problème de classification supervisé. Notre jeu de données contient un ensemble de valeurs où chaque valeur est associée à une variable (ou caractéristique) et à une observation. Les données sont étiquetées de 0 à 4, Chacon Smart Connector WiFi, Bridge Hue, Tp-Link plug, Tp-Link Camera et Domos Door Sensor WiFi respectivement en vue de leur classification, afin d'établir une base d'apprentissage pour le traitement ultérieur des données.

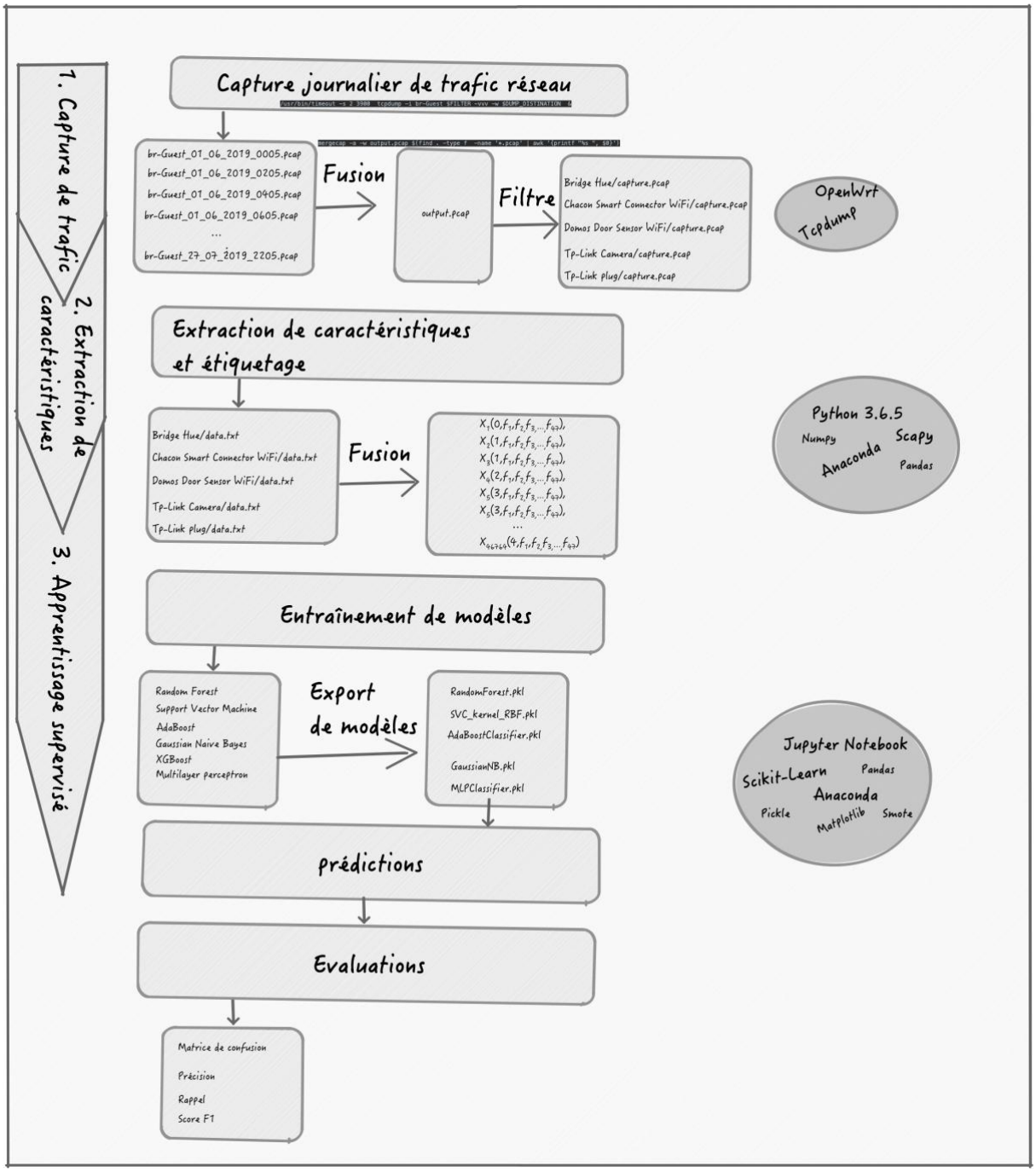

Figure 3.8 : Vue globale de la Méthodologie

3.7 Prétraitement de données

Le prétraitement et le nettoyage des données sont des tâches importantes qui doivent intervenir avant d'utiliser un jeu de données à des fins d'apprentissage automatique. La phase de prétraitement des données consiste en plusieurs opérations incluant le traitement des données manquantes, la correction des données erronées et la transformation des données.

A ce propos, nos premières tentatives d'entraînement de modèles de classification lors de la phase d'apprentissage se sont heurtées à deux problèmes liés à la qualité de données qu'il a fallu traiter.
Premièrement, en regardant la distribution des observations de notre jeu de données, nous remarquons que les données sont dans des ordres de grandeurs différents (cf. la Figure 3.9). Les distributions de chaque caractéristique varient considérablement, ce qui aura une incidence sur nos résultats.

Il existe deux moyens courants pour mettre toutes les variables à la même échelle : la transformation min-max et la normalisation.

La transformation min-max (en anglais, min-max scaling, ou parfois normalization) est assez simple : les valeurs sont décalées et recalibrées afin qu'elles se situent toutes entre 0 et 1.
La normalisation (en anglais, standardization) est assez différente : elle consiste tout d'abord à soustraire la valeur moyenne (de sorte que les valeurs normalisées auront toujours une moyenne nulle), puis à diviser par l'écart-type afin que la distribution qui en résulte ait une variance égale à 1.

```
Entrée [6]: #Aperçus de 5 dernières lignes des données
            data.tail()
```

Out[6]:

	label	min_bytes	max_arr_time	avg_bits_per_sec	std_pkt_len	prot	bootp	flow_duration	avg_pkt_len	push_flag	http	max_bytes	tot_pkt	ntp	ack_fl
46759	4	54	0.122951	84615.156134	475.781840	6	0	0.234379	309.875000	3	0	1514	8	0	
46760	4	54	0.088665	30065.817180	122.543076	6	0	0.228565	143.166667	3	0	396	6	0	
46761	4	54	0.115196	87006.958897	475.781840	6	0	0.227936	309.875000	3	0	1514	8	0	
46762	4	54	0.094968	29651.238265	122.543076	6	0	0.231761	143.166667	3	0	396	6	0	
46763	4	54	0.141824	86412.486199	475.781840	6	0	0.229504	309.875000	3	0	1514	8	0	

Figure 3.9 :la méthode *tail*() dans scikit-learn permet d'afficher les 5 dernières des données. Il est aisé de constater, par exemple, que la moyenne de paquets (avg_pkt_len) est très éloignée de la durée du flux (flow_duration) au niveau d'ordre de grandeur.

La bibliothèque scikit-learn un transformateur appelé MinMaxScaler qui met à l'échelle une caractéristique depuis sa plage originale jusqu'à l'intervalle [min, max]. Les valeurs par défaut sont min=0 et max=1 en l'absence de spécification. Dans notre cas la fourchette se situe entre -1 et 1.

```
Entrée [3]: from sklearn.model_selection import train_test_split
            X = np.array(data.loc[:, data.columns != 'label'])
            y = np.array(data.loc[:, data.columns == 'label'])
            ## Partage du jeu de données entre jeux d'apprentissage et de test
            X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

            from sklearn.preprocessing import MinMaxScaler

            scaling = MinMaxScaler(feature_range=(-1, 1)).fit(X_train)
            X_train = scaling.transform(X_train)
            X_test = scaling.transform(X_test)
```

Figure 3.9 : Standardisation du jeu d'apprentissage et de test avec MinMaxScaler

Notons au passage que le jeu de données a été divisé en deux sous-ensembles : un ensemble d'entraînement X_train et un ensemble de test X_test à hauteur 30% de données. De sorte à entraîner les algorithmes sur le premier sous-ensemble de données et ensuite mesurer leur performance sur le second. Toutes les transformations de prétraitement de données opérées ont été appliquées aux deux sous-ensembles.

Deuxièmement, nous pouvons remarquer dans l'histogramme de la figure 3.10 que le jeu de données présente un déséquilibre des classes. Les classes « Domos » et « Prise Chacon » sont largement sous représentées ou si nous voyons le verre à moitié plein, la classe « Pont Hue » est surreprésentée.

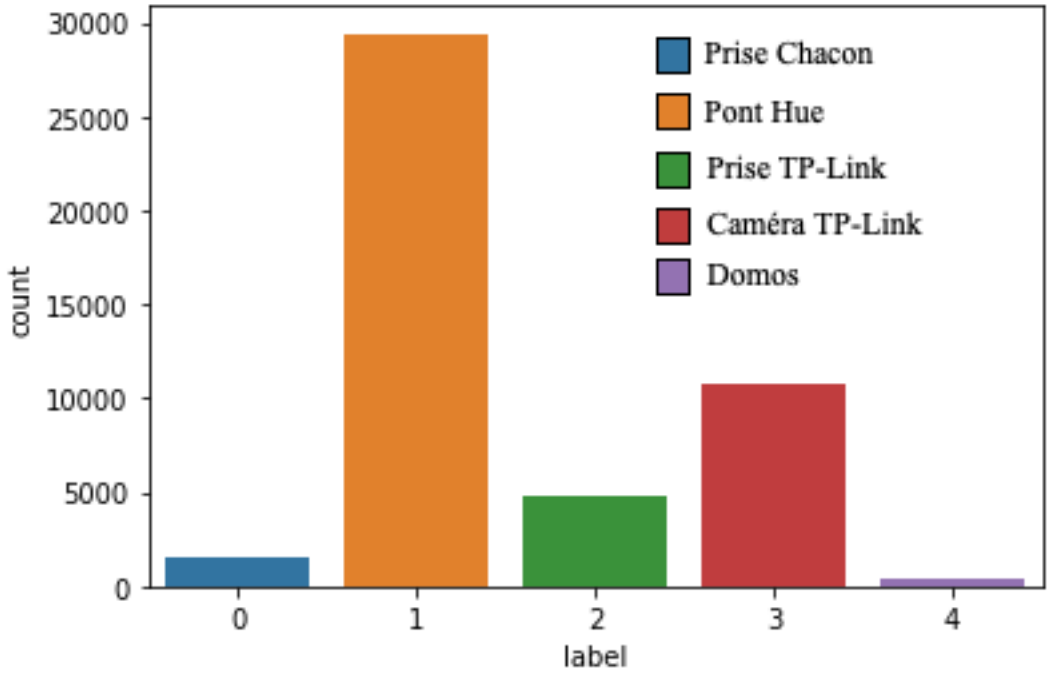

Figure 3.10 : Distributions des classes du jeu de données

Pour remédier à ce problème de déséquilibre de classes, une technique consiste à rééquilibrer le jeu de données avec une technique d'échantillonnage qu'on le peut subdiviser en deux catégories :

- **Sur-échantillonnage**, qui consiste à générer intelligemment des points de données synthétiques pour les classes minoritaires.
- **Sous-échantillonnage**, consiste à supprimer aléatoirement de la base d'apprentissage des échantillons de la ou les classes surreprésentées. Cette stratégie, plus évidente et simple mais elle risque de supprimer des échantillons importants.

C'est la méthode SMOTE [48] de sur-échantillonnage qui a été utilisée pour corriger le problème de déséquilibre de classes. SMOTE génère une instance synthétique X_{new} en sélectionnant au hasard deux échantillons (X_{zi},X_i) de la même classe et en interpolant un point entre ces échantillons. La figure 3.11 illustre ce mécanisme tandis que la figure 3.12 présentent le résultat de l'application de cette méthode au jeu de données du labo.

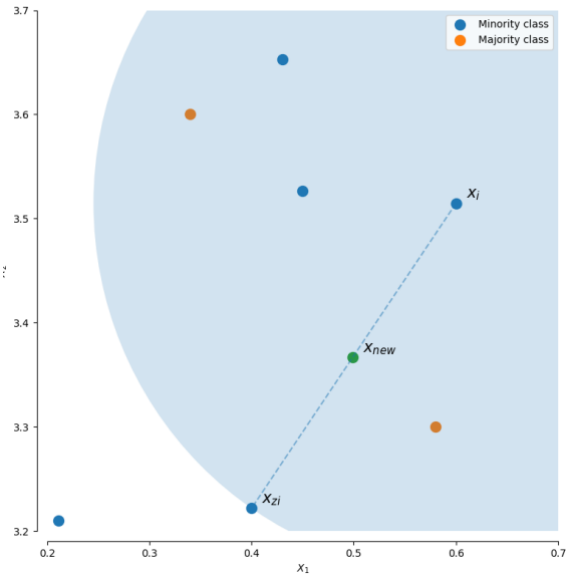

Figure 3.11 : Illustration de la génération d'échantillons dans l'algorithme de sur-échantillonnage [49]

```
Entrée [9]: print("Dimension de la matrice  X  du jeu d'entraînement : {}".format(X_train.shape))
            print("Dimension de la matrice  X du jeu de test avant le sur-échantillonnage: {}\n".format(X_test.shape))
            for index, item in enumerate(DEVICES_LABO):
                print("Nombre d'éléments étiquetés  \"{}\" avant le sur-échantillonnage: {}".format(item,sum(y_train==index)))

Dimension de la matrice  X  du jeu d'entraînement : (32734, 17)
Dimension de la matrice  X du jeu de test avant le sur-échantillonnage: (14030, 17)

Nombre d'éléments étiquetés  "Chacon Smart Connector WiFi" avant le sur-échantillonnage: [1032]
Nombre d'éléments étiquetés  "Bridge Hue" avant le sur-échantillonnage: [20582]
Nombre d'éléments étiquetés  "Tp-Link plug" avant le sur-échantillonnage: [3311]
Nombre d'éléments étiquetés  "Tp-Link Camera" avant le sur-échantillonnage: [7542]
Nombre d'éléments étiquetés  "Domos Door Sensor WiFi" avant le sur-échantillonnage: [267]

Entrée [10]: from imblearn.over_sampling import SMOTE
             sm = SMOTE(random_state=2)

             X_train_res, y_train_res = sm.fit_sample(X_train, y_train.ravel())

             print('Dimension de la matrice du jeu d\'entraînement, après le sur-échantillonnage : {}'.format(X_train_res.shape))
             print('Dimension de la matrice du jeu de test, après le sur-échantillonnage : {}\n'.format(X_test.shape))

             for index, item in enumerate(DEVICE_SAMPLE):
                 print("Nombre d'éléments étiquetés  \"{}\" après le sur-échantillonnage: {}".format(item,sum(y_train_res==index)))

Dimension de la matrice du jeu d'entraînement, après le sur-échantillonnage : (102910, 17)
Dimension de la matrice du jeu de test, après le sur-échantillonnage : (14030, 17)

Nombre d'éléments étiquetés  "Chacon Smart Connector WiFi" après le sur-échantillonnage: 20582
Nombre d'éléments étiquetés  "Bridge Hue" après le sur-échantillonnage: 20582
Nombre d'éléments étiquetés  "Tp-Link plug" après le sur-échantillonnage: 20582
Nombre d'éléments étiquetés  "Tp-Link Camera" après le sur-échantillonnage: 20582
Nombre d'éléments étiquetés  "Domos Door Sensor WiFi" après le sur-échantillonnage: 20582
```

Figure 3.12 : Distributions après l'application de l'algorithme de sur-échantillonnage. Chaque classe compte désormais le même nombre (20582) d'observations. Les classes minoritaires se sont alignées à la classe majoritaire.

Les performances des différents algorithmes sont mesurées sur l'ensemble de tests. Les métriques utilisées sont la matrice de confusion, la précision, le rappel et le score F1. Ces métriques sont décrites dans la prochaine section.

3.8 Métriques

3.8.1 Matrice de confusion

Une matrice de confusion sert à évaluer la qualité d'une classification. Chaque ligne représente la classe réelle, et chaque colonne représente la classe prédite.
On distingue quatre situations :

- **V**rai **P**ositif : instance positive, classifiée positive.
- **F**aux **P**ositif : instance négative, classifiée positive.
- **V**rai **N**égatif : instance négative, classifiée négative.
- **F**aux **N**égatif : instance positive, classifiée négative.

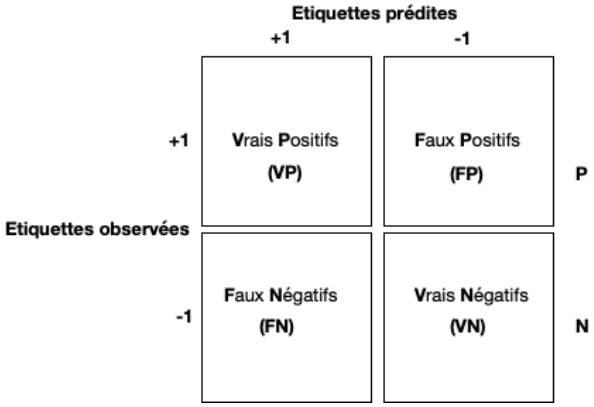

Figure 3.13 : Matrice de confusion

Dans une matrice de confusion, les nombres en diagonales (du coin supérieur gauche au coin inférieur droit) sont les valeurs des échantillons correctement classifiées. L'utilisation des codes couleur dans matrice donne la possibilité de lire rapidement un certain nombre d'occurrences dans un test.
Trois mesures d'évaluation sont dérivées de cette matrice pour mesurer la performance du classifieur. Il s'agit de Rappel, précision et score F1.

3.8.2 Rappel (recall)

Le taux de vrais positifs, appelé rappel ou sensibilité désigne le taux d'étiquettes positives correctement prédites par le classifieur. Le rappel est égal à 0 lorsqu'aucun positif n'a été détecté comme point positif, tandis qu'il est égal à 1 lorsque tous les positifs ont été détectés comme points positifs. Mathématiquement, le rappel est défini comme suit :

$$\text{Rappel} = \frac{\textbf{V}\text{rais } \textbf{P}\text{ositifs}}{\textbf{V}\text{rais } \textbf{P}\text{ositifs} + \textbf{F}\text{aux } \textbf{N}\text{égatifs}}$$

3. 8.3 Précision

La précision mesure la proportion des échantillons prédits comme étant positifs et qui le sont effectivement. Elle est égale à 0 lorsque tous les positifs ont été détectés négatifs. Sinon elle est égale à 1 si tous les négatifs ont été détectés négatives.

$$\text{Précision} = \frac{\textbf{V}\text{rais } \textbf{P}\text{ositifs}}{\textbf{V}\text{rais } \textbf{P}\text{ositifs} + \textbf{F}\text{aux } \textbf{P}\text{ositifs}}$$

3.8.4 Score F1

Communément appelé f1-score ou f-mesure, il s'agit d'une mesure qui considère à la fois précision et le rappel pour calculer un certain score. Le score F1 peut être interprété comme une moyenne pondérée des valeurs de précision (P) et de rappel (R). Lorsque la précision est égale au rappel, on obtient une égalité : Précision = Rappel = f1-score.
Pour le choix d'un algorithme, on cherchera à choisir celui dont le score f1 présente un indice plus élevé. Son calcul est donné comme suit :

$$\text{f1-score} = 2\,\frac{P * R}{P + R}$$

Chapitre 4 : Evaluation

Le présent chapitre présente l'évaluation des performances des différents algorithmes de classification mis en place selon les critères définis précédemment.

La première partie évalue les performances à partir du jeu de données issu des traces de trafic réseau collectées dans les conditions exposées précédemment dans le chapitre 3.
La deuxième partie présente le résultat de l'application de notre solution sur un jeu de données externe issu des fichiers de capture de trafic réseau disponible publiquement [36].

4.1 Evaluation sur notre jeu de données

Le jeu de données utilisé dans le cadre du labo a été collecté à partir du trafic réseau provenant de 4 dispositifs IoT différents : une Caméra TP-Link, un capteur d'ouverture de porte/fenêtre, une prise connectée TP-Link, une prise connectée Chacon et un pont Philips Hue gérant deux éclairages.
Comme nous l'avons exposé dans le chapitre 3, 47 caractéristiques ont été, dans un premier temps, extraites des fichiers de captures réseau pour constituer notre socle de caractéristiques exploitables par un algorithme d'apprentissage.
Dans un second temps, nous avons réduit ce nombre à 17 caractéristiques tout en conservant la même pertinence à l'aide de la méthode intégrée de l'algorithme de forêt d'arbres décisionnels (Random Forest).
Au final, notre jeu de données comprend 46764 observations et 18 caractéristiques dont une désigne l'étiquette (label). Il a, par la suite, été divisé en deux sous-ensembles (jeu d'entraînement et de test) durant la phase d'apprentissage supervisé. Une fois les modèles entraînés sur le jeu d'entraînement, nous avons vérifié leur performance sur le jeu de test à l'aide des métriques décrites plus haut inclues dans la bibliothèque Scikit-learn.

En ce qui concerne l'apprentissage supervisé, nous avons soumis le jeu de données à six modèles de classification multiclasses : Random Forest, SVM, AdaBoost, XGBoost, Gaussian Naïve Bayes et Perceptron multicouche.
Il en résulte que le modèle Random Forest s'est avéré être le plus performant au vue des résultats de métriques. De plus, son temps d'apprentissage est assez rapide comparé aux autres.

Il est très important de prendre en considération la distribution des classes (étiquettes) dans l'interprétation des métriques. En effet, notre jeu de test présente un déséquilibre au niveau

de la répartition entre classes comme il est reporté dans la colonne « support » du tableau 4.1 correspondant au nombre d'échantillons par classe. On peut remarquer que les étiquettes « Domos Door Sensor WiFi » et « Chacon Smart Connector WiFi » représentent 0.73 % te 3.19 % du jeu de test.

Ainsi, dans ce même tableau, on peut remarquer que l'étiquette « Chacon Smart Connector WiFi » présente une précision de 0.28 ce qui peut sembler très peu alors même que d'après la matrice de confusion 439 échantillons de cette étiquette ont été correctement classés sur un total 447.
L'explication se trouve au niveau de l'étiquette « Bridge Hue », où 1127 de faux positifs ont été prédits et de ce fait, ont contribué à faire baisser mécaniquement la précision de classification de l'étiquette « Chacon Smart Connector WiFi » :
$précision = \frac{\text{VP}}{VP+FP} = \frac{439}{439+1127+10+1+1}$ =0.28

C'est pour cette raison que les métriques du tableau 4.1 doivent être interpréter en prenant en compte toutes les mesures en même temps : précision, rappel et distribution de classes.

Afin, une métrique couramment utilisée avec des jeux de données déséquilibrés dans un contexte de multiclasses est la moyenne micro (micro avg) qui rend compte de la moyenne des valeurs de précision ou rappel mais en prenant compte du déséquilibre des classes. Tandis que la moyenne macro calcule tout simplement la moyenne de ces valeurs.

	precision	recall	f1-score	support
Chacon Smart Connector WiFi	0,28	0,98	0,43	447
Bridge Hue	0,96	0,87	0,91	8832
Tp-Link plug	0,99	0,80	0,89	1430
Tp-Link Camera	0,91	0,90	0,91	3219
Domos Door Sensor WiFi	0,87	0,95	0,91	102
micro avg	0,88	0,88	0,88	14030
macro avg	0,80	0,90	0,81	14030
weighted avg	0,93	0,88	0,90	14030

Tableau 4.1 : Rapport de classification du modèle Random Forest

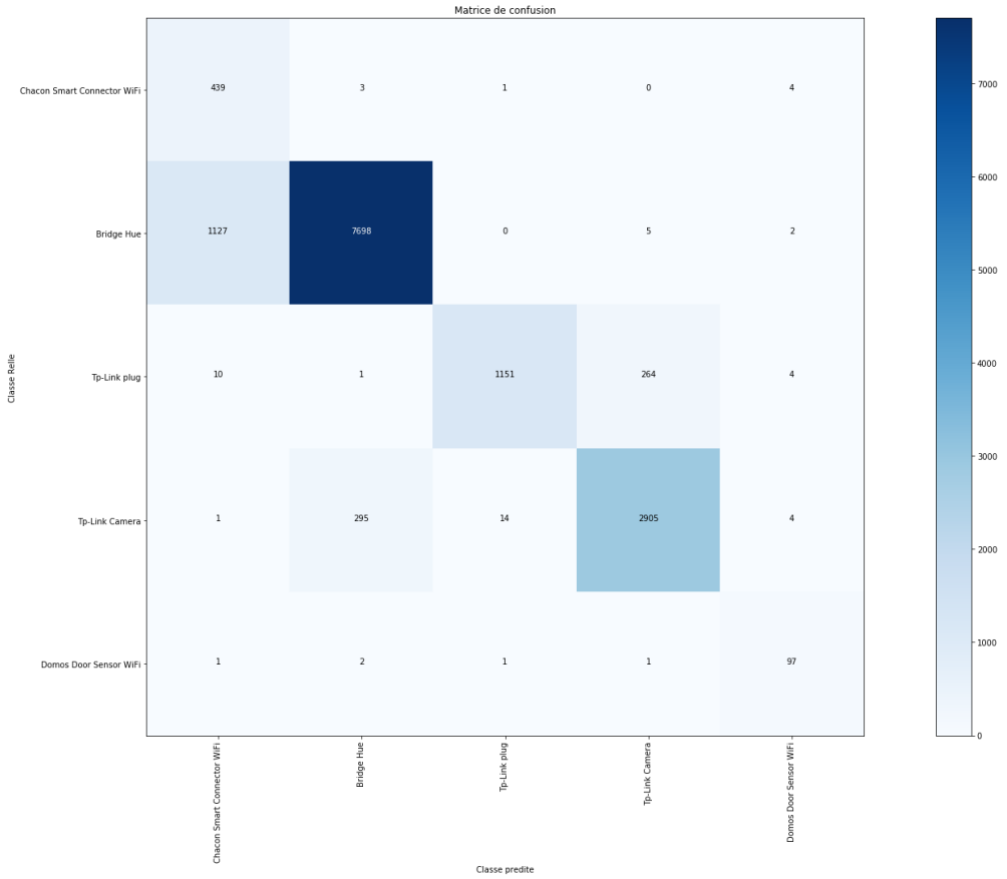

Figure 3.14 : Matrice de confusion du modèle Random Forest

La figure3.15 nous permet d'observer les mesures F1-score des différents algorithmes de classification utilisés sur le jeu de test. On remarque que le Random Forest est celui qui performe le mieux suivi de près par l'algorithme XGboost.

	Random Forest	SVM	AdaBoost	XGBoost	Gaussian Naïve	Perceptron multicouche
F1-Score max	0,91	0,87	0,80	0,91	0,23	0,30
F1-Score min	0,43	0,51	0,16	0,42	0,02	0,11
F1-Score micro avg	0,88	0,80	0,69	0,86	0,12	0,64
F1-Score macro avg	0,81	0,71	0,45	0,79	0,14	0,41

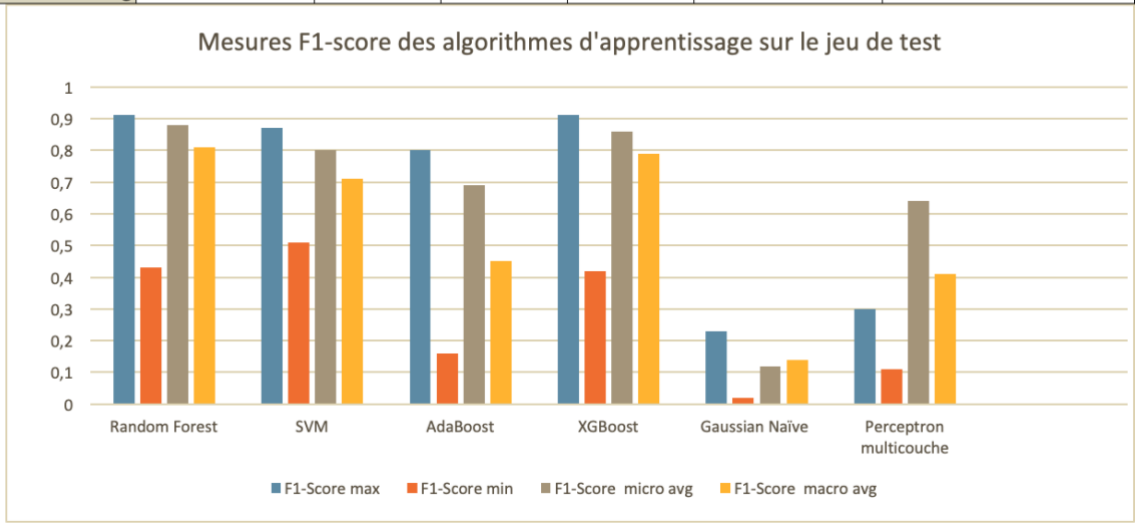

Figure 3.15 : Performance des algorithmes d'apprentissage

Les modèles ont été entraînés sans optimisation au niveau des paramètres, exception faite pour le modèle SVM, pour lequel nous avons utilisé une technique appelée recherche par quadrillage (grid search), qui consiste à effectuer une recherche exhaustive pour trouver les meilleurs paramètres parmi un ensemble de valeurs. Cependant, cette technique bien que performante, présente un coût considérable en temps de calcul.

	precision	recall	f1-score	support
Chacon Smart Connector WiFi	0.97	0.45	0.62	447
Bridge Hue	0.78	0.99	0.87	8832
Tp-Link plug	0.86	0.78	0.82	1430
Tp-Link Camera	0.97	0.35	0.51	3219
Domos Door Sensor WiFi	0.92	0.65	0.76	102
micro avg	0.80	0.80	0.80	14030
macro avg	0.90	0.64	0.71	14030
weighted avg	0.84	0.80	0.77	14030

Figure 3.16 : Rapport de classification du modèle SVM avec un noyau RBF (pour Radial Basis Function) en utilisant la technique de recherche par quadrillage.

	precision	recall	f1-score	support
Chacon Smart Connector WiFi	0.25	0.55	0.34	447
Bridge Hue	0.94	0.69	0.80	8832
Tp-Link plug	0.37	0.21	0.26	1430
Tp-Link Camera	0.54	0.94	0.69	3219
Domos Door Sensor WiFi	0.13	0.22	0.16	102
micro avg	0.69	0.69	0.69	14030
macro avg	0.45	0.52	0.45	14030
weighted avg	0.76	0.69	0.70	14030

Figure 3.17 : Rapport de classification du modèle Adaboost

	precision	recall	f1-score	support
Chacon Smart Connector WiFi	0.31	0.19	0.23	447
Bridge Hue	0.94	0.12	0.21	8832
Tp-Link plug	0.09	0.14	0.11	1430
Tp-Link Camera	0.83	0.05	0.10	3219
Domos Door Sensor WiFi	0.01	0.98	0.02	102
micro avg	0.12	0.12	0.12	14030
macro avg	0.44	0.30	0.14	14030
weighted avg	0.80	0.12	0.18	14030

Figure 3.18 : Rapport de classification du modèle bayésien naïf (Gaussian Naive Bayes)

	precision	recall	f1-score	support
Chacon Smart Connector WiFi	0.27	0.96	0.42	447
Bridge Hue	0.96	0.86	0.91	8832
Tp-Link plug	0.95	0.79	0.86	1430
Tp-Link Camera	0.90	0.89	0.89	3219
Domos Door Sensor WiFi	0.82	0.95	0.88	102
micro avg	0.86	0.86	0.86	14030
macro avg	0.78	0.89	0.79	14030
weighted avg	0.92	0.86	0.88	14030

Figure 3.19 : Rapport de classification du modèle XGBoost pour eXtreme Gradient Boosting

	precision	recall	f1-score	support
Chacon Smart Connector WiFi	0.36	0.26	0.30	447
Bridge Hue	0.73	0.84	0.78	8832
Tp-Link plug	0.68	0.57	0.62	1430
Tp-Link Camera	0.40	0.16	0.23	3219
Domos Door Sensor WiFi	0.06	0.67	0.11	102
micro avg	0.64	0.64	0.64	14030
macro avg	0.45	0.50	0.41	14030
weighted avg	0.63	0.64	0.62	14030

Figure 3.20 : Rapport de classification du modèle Perceptron multicouche (MLP pour Multilayer Perceptron)

4. 2 Evaluation sur un jeu de données externe

[36] que nous avons présenté dans l'état de l'art, ont mis à la disposition de la communauté de chercheurs les traces de trafic réseau utilisées dans le cadre de leur étude. Il s'agit de fichiers de capture réseaux au format PCAP et CSV rendus téléchargeables [50].

Nous avons récupéré 3 fichiers au format PCAP représentant 3 jours de captures auxquels nous avons appliqué, par la suite, la même transformation décrite la chapitre 3 consistant à extraire des caractéristiques à l'aide d'un script python.
L'étiquetage de données a été facilité par la liste d'adresses MAC fournie avec les données. En sortie, nous disposons d'un jeu de données comptant 62631 observations et, comme pour notre propre jeu de données, 18 caractéristiques dont une désigne l'étiquette (label).

Ce jeu de données est intéressant à deux égards. Premièrement, il est constitué d'une variété de dispositifs comprenant aussi bien de l'IoT que du non IoT appartenant à différentes catégories (cf. la figure 3.21). Deuxièmement, les conditions de collecte de trafic réseaux sont similaires aux nôtres dans le sens où les auteurs ont mis en place une expérience simulant un environnement dit « intelligent » dans lequel des traces sont collectées automatiquement et sauvegardées au format pcap à l'aide de tcpdump sur une période de plus de 3 semaines.

Category	Device
Hubs	Smart Things
	Amazon Echo
Cameras	Netatmo Welcome
	TP-Link Day Night Cloud camera
	Samsung SmartCam
	Dropcam
	Insteon Camera
	Withings Smart Baby Monitor
Switches & Triggers	Belkin Wemo switch
	TP-Link Smart plug
	iHome
	Belkin wemo motion sensor
Air quality sensors	NEST Protect smoke alarm
	Netatmo weather station
Healthcare devices	Withings Smart scale
	Blipcare Blood Pressure meter
	Withings Aura smart sleep sensor
Light Bulbs	LiFX Smart Bulb
Electronics	Triby Speaker
	PIX-STAR Photo-frame
	HP Printer

Figure 3.21 : Liste des périphériques IoT

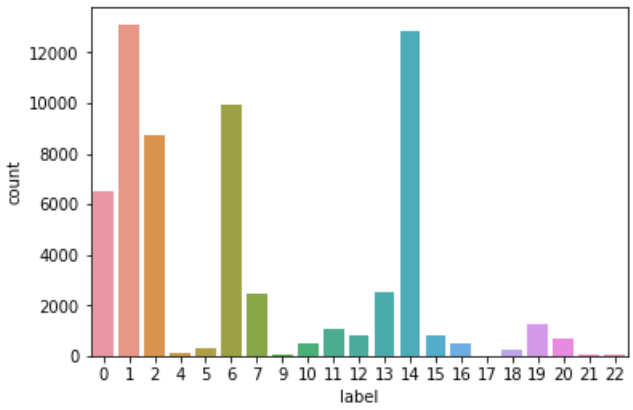

Figure 3.22 : Distribution des classes

Le jeu de données externe présente également un déséquilibre au niveau des classes (étiquettes) comme l'illustre la figure 3.22. Nous avons donc appliqué le même traitement notamment la méthode de sur-échantillonnage précédemment utilisé et ensuite le jeu de données a été divisé en deux sous-ensembles : un ensemble d'entraînement X_train et un ensemble de test X_test à hauteur 30% de données.

Tout comme sur notre propre jeu de données, nous avons entraîné les algorithmes sur le premier sous-ensemble de données et ensuite évalué leur performance sur le second.

Les modèles Random Forest et XGBoost donnent de meilleurs résultats et par rapport aux modèles issus de notre propre jeu de données, ils sont sensiblement plus précis.

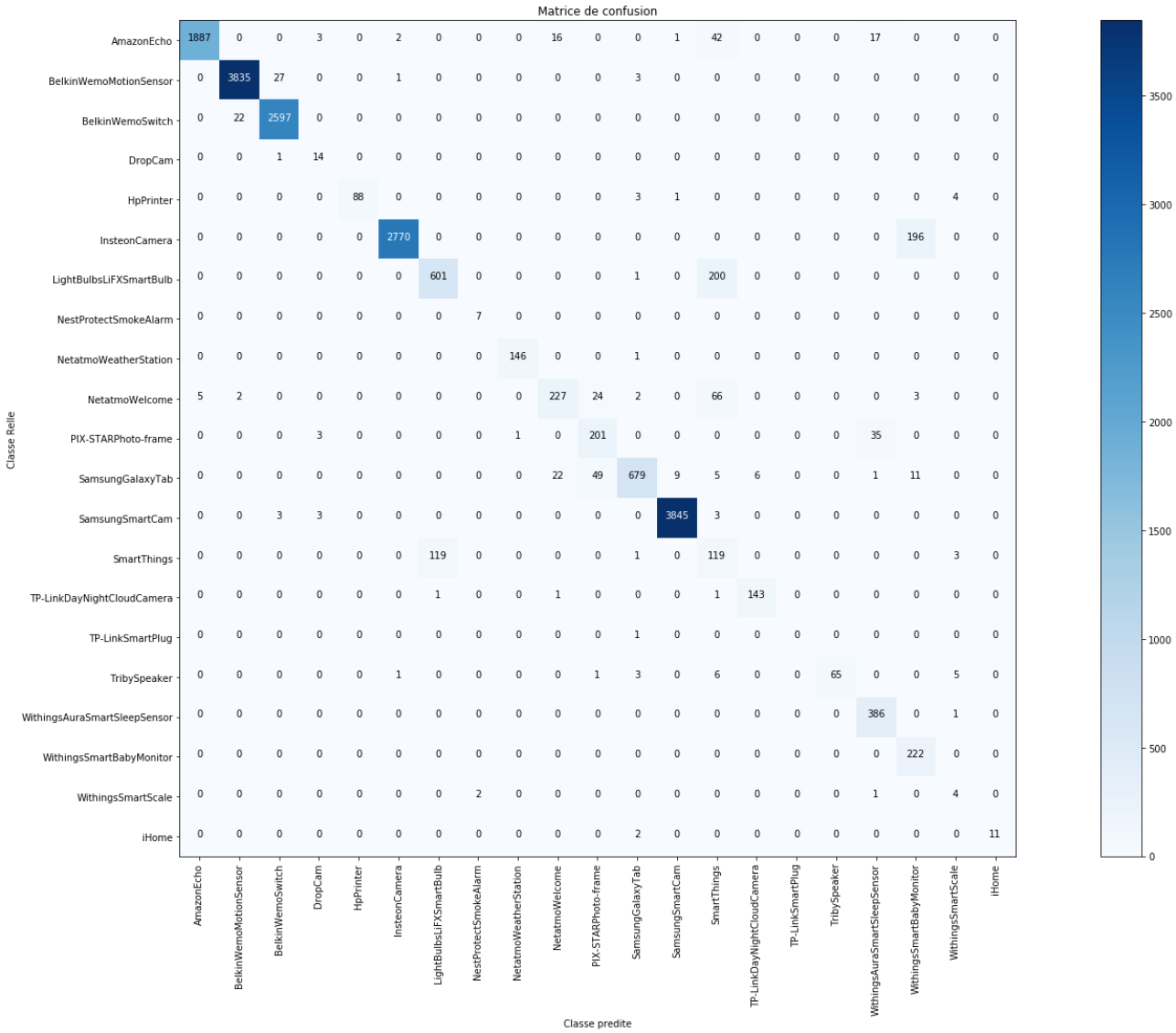

Figure 3.23 : Matrice de confusion du modèle Random Forest

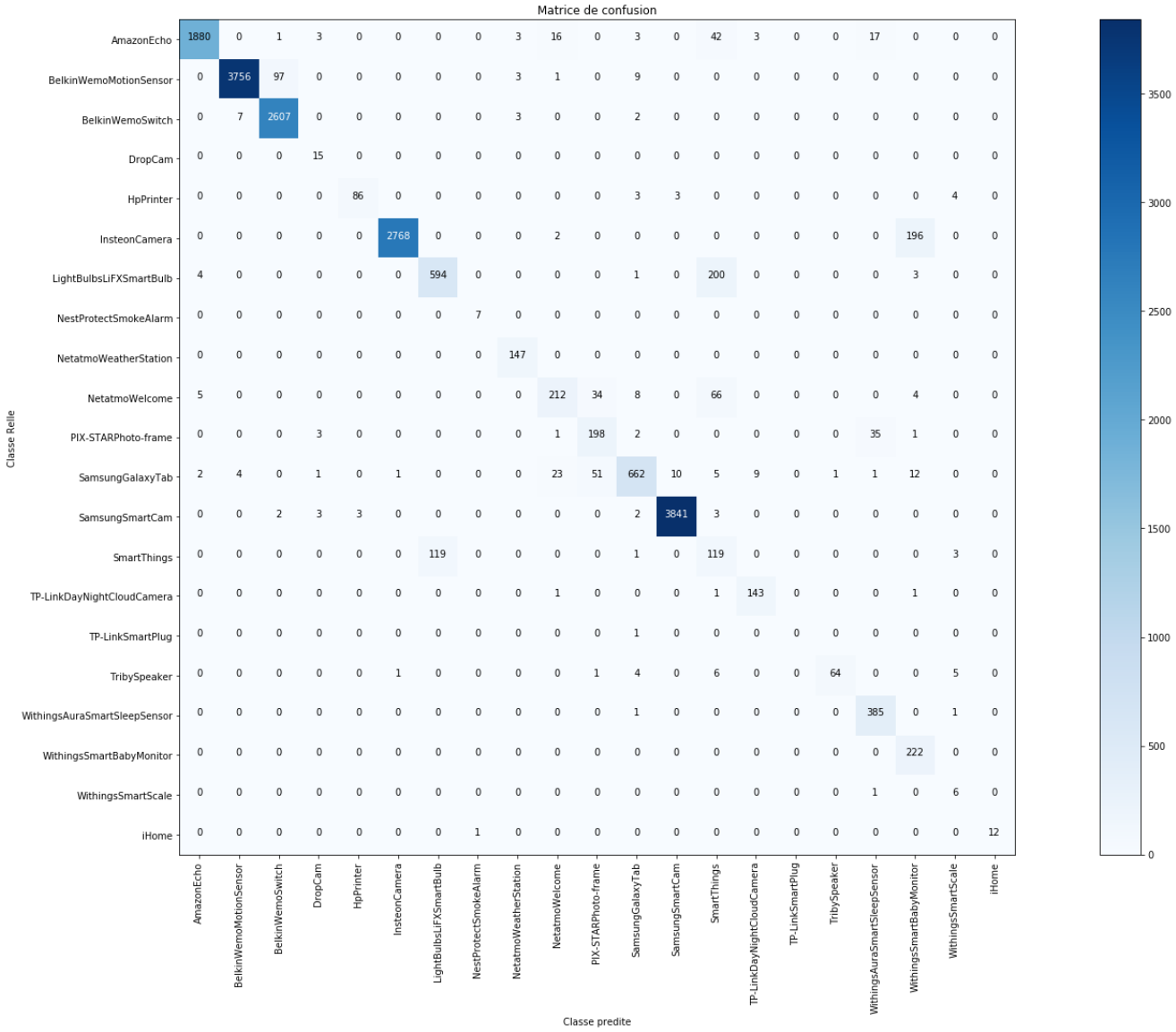

Figure 3.24 : Matrice de confusion du modèle XGBoost

	precision	recall	f1-score	support
AmazonEcho	1.00	0.96	0.98	1968
BelkinWemoMotionSensor	0.99	0.99	0.99	3866
BelkinWemoSwitch	0.99	0.99	0.99	2619
DropCam	0.61	0.93	0.74	15
HpPrinter	1.00	0.92	0.96	96
InsteonCamera	1.00	0.93	0.97	2966
LightBulbsLiFXSmartBulb	0.83	0.75	0.79	802
NestProtectSmokeAlarm	0.78	1.00	0.88	7
NetatmoWeatherStation	0.99	0.99	0.99	147
NetatmoWelcome	0.85	0.69	0.76	329
PIX-STARPhoto-frame	0.73	0.84	0.78	240
SamsungGalaxyTab	0.98	0.87	0.92	782
SamsungSmartCam	1.00	1.00	1.00	3854
SmartThings	0.27	0.49	0.35	242
TP-LinkDayNightCloudCamera	0.96	0.98	0.97	146
TP-LinkSmartPlug	0.00	0.00	0.00	1
TribySpeaker	1.00	0.80	0.89	81
WithingsAuraSmartSleepSensor	0.88	1.00	0.93	387
WithingsSmartBabyMonitor	0.51	1.00	0.68	222
WithingsSmartScale	0.24	0.57	0.33	7
iHome	1.00	0.85	0.92	13
micro avg	0.95	0.95	0.95	18790
macro avg	0.79	0.84	0.80	18790
weighted avg	0.96	0.95	0.95	18790

Figure 3.25 : Rapport de classification du modèle Random Forest

	precision	recall	f1-score	support
AmazonEcho	0.99	0.96	0.97	1968
BelkinWemoMotionSensor	1.00	0.97	0.98	3866
BelkinWemoSwitch	0.96	1.00	0.98	2619
DropCam	0.60	1.00	0.75	15
HpPrinter	0.97	0.90	0.93	96
InsteonCamera	1.00	0.93	0.97	2966
LightBulbsLiFXSmartBulb	0.83	0.74	0.78	802
NestProtectSmokeAlarm	0.88	1.00	0.93	7
NetatmoWeatherStation	0.94	1.00	0.97	147
NetatmoWelcome	0.83	0.64	0.72	329
PIX-STARPhoto-frame	0.70	0.82	0.76	240
SamsungGalaxyTab	0.95	0.85	0.89	782
SamsungSmartCam	1.00	1.00	1.00	3854
SmartThings	0.27	0.49	0.35	242
TP-LinkDayNightCloudCamera	0.92	0.98	0.95	146
TP-LinkSmartPlug	0.00	0.00	0.00	1
TribySpeaker	0.98	0.79	0.88	81
WithingsAuraSmartSleepSensor	0.88	0.99	0.93	387
WithingsSmartBabyMonitor	0.51	1.00	0.67	222
WithingsSmartScale	0.32	0.86	0.46	7
iHome	1.00	0.92	0.96	13
micro avg	0.94	0.94	0.94	18790
macro avg	0.79	0.85	0.80	18790
weighted avg	0.96	0.94	0.95	18790

Figure 3.26 : Rapport de classification du modèle XGBoost

Chapitre 5 : Discussion

Les performances présentées dans le chapitre précèdent, montrent que la solution proposée pour le fingerprinting de dispositifs IoT est viable. Les algorithmes de classifications par arbres de décision se sont avérés particulièrement efficaces, et ce, avec le paramétrage par défaut des hyper-paramètres des modèles d'apprentissage.

D'une manière générale, les algorithmes explorés dans le cadre du ce travail ont été utilisés dans leur configurations par défaut. Seul le modèle de Machine à vecteurs de support a été entraîné en combinaison avec la recherche par quadrillage (Grid Search). Bien que les performances de ce dernier se soient améliorées, en contrepartie, le temps de calcul s'est avéré plus conséquent.

5. 1 Jeu de données

Le jeu de données utilisé dans le cadre du labo a été collecté, sur une période de 11 semaines, à partir du trafic réseau provenant de quatre dispositifs IoT différents : une Caméra TP-Link, un capteur d'ouverture de porte/fenêtre, une prise connectée TP-Link, une prise connectée Chacon et un pont Philips Hue. Les trois premiers dispositifs ont été fournis par l'université de Namur dans le cadre de ce mémoire.

Les traces ont été sauvegardées au format PCAP pour une analyse ultérieure et sont mises à disposition en ligne[6] ainsi que les modules python ayant servis à l'extraction de caractéristiques.

Un autre jeu de données provenant d'une source externe [50] a servi dans la construction de modèles de classification et le résultat nous a conforté dans l'idée de la faisabilité de méthode de fingerprinting de dispositifs IoT proposée. En effet, ce jeu de données est issu des traces de réseau de 21 dispositifs, aussi bien IoT et non IoT, de différentes catégories et représentatifs d'objets connectés d'un réseau domestique.

5. 2 Apprentissage automatique

Nombreuses études [31] ont démontré le rôle important qu'occupe l'apprentissage automatique en matière de découverte et l'exploration du trafic réseau. Des nouvelles approches, basées notamment sur la classification des flux sur base des caractéristiques statistiques, ont remplacées des méthodes traditionnelles moins efficaces qui se basent sur le numéro de port ou l'inspection de la charge utile (payload) de paquet.

De plus, les méthodes basées sur les statistiques de flux reposent sur des informations extraites de façon passive des en-têtes de paquets TCP : par exemple, la taille de paquets, temps inter-arrivée, la taille de fenêtre TCP, le port source et destination.

Etant données le grand nombre et de l'hétérogénéité des dispositifs IoT, l'approche statistique basée sur des techniques d'apprentissage automatique de classification semble bien indiquée pour le fingerprinting de dispositifs IoT.

[6] https://bitbucket.org/elsanto/fingerprinting-device-iot

5. 3 Contribution

La méthode proposée se base sur des statistiques des données de flux réseau générés par les dispositifs IoT pour procéder à leur identification. Seul le flux unidirectionnel est considéré dans ce travail, en comparaison de l'état de l'art [34], [35], [36]. De plus, que le trafic soit chiffré ou non n'a pas d'importance, étant donné que parmi les caractéristiques exploitées, le contenu de la charge utile n'est pas pris en compte en tant que tel.

5. 4 Limitation

Le nombre limité des dispositifs IoT utilisé dans le cadre du labo n'est pas assez important pour assurer une bonne représentativité des objets connectés que l'on retrouve dans un réseau domestique. Bien que cette limitation ait été atténuée par l'utilisation de données externes même si nous sommes dépendants des conditions dans lesquelles celles-ci ont été produites.

Par ailleurs, le trafic réseau collecté ne représente que six heures par jour. Et les traces ont essentiellement été capturées sans interaction humaine, c'est-à-dire de machine à machine (M2M).
En outre, un changement dans le firmware d'un dispositif IoT via une mise à jour par exemple, impliquerait de réentraîner tous les modèles de classification. Il en va de même, si un nouveau dispositif devait être introduite dans le parc des dispositifs IoT sous surveillance.

Chapitre 6 Conclusion

L'objectif principal de ce travail était de proposer une méthode de fingerprinting de dispositifs IoT en analysant les données de trafic réseau.
Les traces de trafic ont été collectées et analysées pour trouver les modèles de trafic sous-jacents. A cette fin, une infrastructure d'objets connectés simulant une maison intelligente a été déployée pour permettre de collecter le trafic réseau en conditions réelles d'utilisation sur une période de 11 semaines.

Au cours de la phase exploratoire du trafic réseau, nous n'avons eu cesse de faire le lien avec la problématique de la sécurité et de la confidentialité des objets connectés. Même si ces questions n'ont pas été développées dans ce travail, le fingerprinting de devices s'inscrit, avant tout, dans une perspective plus large en matière de sécurité et confidentialité des données.

Pour ne citer qu'un exemple, nous avons été surpris de constater que la prise Wi-Fi Chacon [51] dialoguait avec plusieurs serveurs dont un basé en Chine. Qu'il existe également une interface cachée d'administration [52] protégée par un mot de passe par défaut que l'on retrouve facilement en ligne. Cette interface, accessible depuis l'adresse IP du dispositif, se présente par défaut en langue asiatique avec la possibilité de basculer en anglais, des informations sensibles y sont disponibles et il est possible, par exemple, de mettre à jour ou changer le firmware du produit en téléchargement.
A aucun moment, le manuel d'utilisation ne fait mention de cette porte dérobée. Même si, son accès suppose que l'utilisateur soit déjà connecté au même réseau Wi-Fi.

Gérard Berry [53], membre de l'Académie des sciences et lauréat de la médaille d'or 2014 du CNRS, a qualifié les objets connectés de « passoires » en termes de sécurité. Une déclaration qui fait écho à de nombreuses littératures qui traitent cette problématique.

Par rapport à l'objectif principal, nous avons proposé des caractéristiques que nous avons jugées suffisamment pertinentes pour discriminer les flux issus du trafic des différents dispositifs IoT. Le nombre des caractéristiques a ensuite été réduit, tout en conservant les performances.
De nombreuses contraintes liées aux jeux de données ont été alors soulevées et une solution a été proposée pour y faire face. Différents algorithmes d'apprentissage machine ont ensuite été utilisés pour classifier le trafic réseau.
A cet égard, les algorithmes de classifications par arbres de décision se sont particulièrement distinguées.
Enfin, nous avons appliqué notre solution de fingerprinting de dispositifs IoT sur un jeu de données externe provenant d'un parc d'objets connectés plus important et plus varié. Et nous avons obtenu des résultats similaires.

6.1 Travaux futurs

En l'état actuel, la solution proposée est limitée à la phase de constitution de modèles capables de distinguer le type de dispositifs IoT sur base de données formatées pour l'apprentissage automatique.
Afin de parvenir à une solution pratique, quelques étapes sont nécessaires :

- **Automatisation de l'extraction de caractéristiques** : incorporer le module python d'extraction dans le processus de capture de trafic réseau déjà mis en place.

- **Mesurer les performances avec des données non encore vues** : Une des pistes envisagées serait ajouter une nouvelle étiquette pour classer des données provenant d'un dispositif non encore étiqueté.

- **Modèles de classification :** Il y a certainement moyen améliorer les modèles de classification que nous avons mis en place, à commencer par les différents réglages au niveau des hyper-paramètres.

Bibliographie

[1] Rfid Journal, That 'Internet of Things' Thing, https://www.rfidjournal.com/articles/view?4986

[2] Raspberry https://www.raspberrypi.org/

[3] OpenWrt https://openwrt.org/

[4] Jason Franklin, Damon McCoy, Parisa Tabriz, Vicentiu Neagoe, Jamie Van Randwyk, and Douglas Sicker. Passive data link layer 802.11 wireless device driver fingerprinting. In Proceedings Usenix Security 06, August 2006

[5] Clarence CHIO, David FREEMAN Machine Learning et sécurité - Protéger les systèmes avec des données et des algorithmes - collection O'Reilly, collection O'Reilly page 119-121

[6] Chen, Tianqi and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System." KDD (2016)

[7] Clarence CHIO, David FREEMAN Machine Learning et sécurité - Protéger les systèmes avec des données et des algorithmes - collection O'Reilly, collection O'Reilly page 121-124

[8] Xavier Aghina, « La sécurité de l'IOT », https://www.it-expertise.com/la-securite-de-liot/

[9] Usurpation d'adresse IP. Wikipédia, l'encyclopédie libre. Page consultée le 13 juillet, 2019 à partir de http://fr.wikipedia.org/w/index.php?title=Usurpation_d%27adresse_IP&oldid=144073767.

[10] Ronak Sutaria, Raghunath Govindachari (May 01, 2013) "Understanding The Internet Of Things", https://www.electronicdesign.com/iot/understanding-internet-things

[11] Christoph Neumann, Olivier Heen, Stéphane Onno, "An empirical study of passive 802.11 Device Fingerprinting", 2012 32nd International Conference on Distributed Computing Systems Workshops

[12] T. Kohno, A. Broido, and K. Claffy, "Remote physical device fingerprinting," Transactions on Dependable and Secure Computing (TDSC), vol. 2, no. 2, pp. 93–108, April 2005

[13] "Nmap ("Network Mapper")," https://nmap.org/

[14] "Xprobe," http://xprobe.sourceforge.net/

[15] "P0f," http://lcamtuf.coredump.cx/p0f.shtml

[16] A. Sivanathan, H. Habibi Gharakheili, V. Sivaraman, "Can We Classify an IoT Device Using TCP Port Scan?", IEEE ICIAfS, Colombo, Sri Lanka, Dec 2018

[17] "clock skew," https://en.wikipedia.org/wiki/Clock_skew

[18] S. Moon, P. Skelly, and D. Towsley. Estimation and removal of clock skew from network delay measurements. In Proceedings of Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies., volume 1 of INFOCOM'99, pages 227–234, 1999

[19] "rfc792," https://tools.ietf.org/html/rfc792

[20] "rfc1323," https://tools.ietf.org/html/rfc1323

[21] S. Jana and S. K. Kasera, "On fast and accurate detection of unauthorizedwireless access points using clock skews," inProceedings of the 14thACM International Conference on Mobile Computing and Networking,ser. MobiCom '08. ACM, 2008, pp. 104–115

[22] "couche d'abstraction matérielle," https://fr.wikipedia.org/wiki/Couche_d%27abstraction_mat%C3%A9rielle

[23] Célestin Matte, Mathieu Cunche, Franck Rousseau, Mathy Vanhoef. Defeating MAC Address Randomization Through Timing Attacks. ACM WiSec 2016, Jul 2016, Darmstadt, Germany. ff10.1145/2939918.2939930ff. ffhal-01330476f

[24] Sergey Bratus, Cory Cornelius, David Kotz, and Daniel Peebles. Active behavioral fingerprinting of wireless devices. In Proceedings of ACM WiSec'08, March 2008

[25] K. N. Gopinath, P. Bhagwat, and K. Gopinath. An empirical analysis of heterogeneity in ieee 802.11 mac protocol implementations and its implications. In Proceedings of the 1st ACM international workshop on wireless network testbeds, experimental evaluation & characterization (WiNTECH'06), pages 80–87, 2006

[26] M. Miettinen, S. Marchal, I. Hafeez, T. Frassetto, N. Asokan, A. R. Sadeghi, and S. Tarkoma. IoT Sentinel demo : automated device-type identification for security enforcement in IoT. In IEEE 37th International Conference on Distributed Computing Systems (ICDCS), pages 2511–2514, June 2017

[27] Adnan, Md Nasim & Islam, Md. (2015). One-Vs-All Binarization Technique in the Context of Random Forest

[28] B. Bezawada, M. Bachani, J. Peterson, H. Shirazi, I. Ray, and I. Ray, "Iotsense: Behavioral fingerprinting of iot devices," arXiv preprint arXiv:1804.03852, 2018

[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research 12 (2011), 2825–2830

[30] Clarence CHIO, David FREEMAN Machine Learning et sécurité - Protéger les systèmes avec des données et des algorithmes - collection O'Reilly, collection O'Reilly page 139-142

[31] Y. WANG and S. Z. YU, "Supervised Learning Real-time Traffic Classifiers," Journal of Networks, vol. 4, no. 7, pp. 622-629, 2009

[32] Dmitri Bekerman, Bracha Shapira, Lior Rokach, and Ariel Bar. 2015. Unknown malware detection using network traffic classification. In Conf. on Communications and Network Security (CNS). IEEE, 134–142

[33] Bekerman et al (2015) Network Traffic Features Set
http://www.ise.bgu.ac.il/dima/network_traffic_features_set.pdf

[34] Yair Meidan, Michael Bohadana, Asaf Shabtai, Juan David Guarnizo, Martin Ochoa, Nils Ole Tippenhauer, and Yuval Elovici. 2017. ProfilIoT: A Machine Learning Approach for IoT Device Identification Based on Network Traffic Analysis. In SAC 2017: The 32nd ACM Symposium On Applied Computing

[35] Y. Meidan, M. Bohadana, A. Shabtai, M. Ochoa, N. O. Tippenhauer, J. D. Guarnizo, and Y. Elovici, "Detection of unauthorized iot devices using machine learning techniques," arXiv preprint arXiv:1709.04647, 2017

[36] A. Sivanathan et al., "Characterizing and classifying IoT traffic in smart cities and campuses," in 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 559–564 (2017)

[37] Suite cryptographique. (2017, novembre 24). Wikipédia, l'encyclopédie libre. Page consultée le 13:33, novembre 24, 2017 à partir de
http://fr.wikipedia.org/w/index.php?title=Suite_cryptographique&oldid=142891726

[38] LuCI https://openwrt.org/docs/guide-user/luci/luci.essentials

[39] opkg https://openwrt.org/docs/guide-user/additional-software/opkg

[40], DnsMasq https://openwrt.org/docs/guide-user/base-system/dhcp.dnsmasq

[41] Scapy https://scapy.net/

[42] A. Moore, et al., "Crogan, Discriminators for use in flow-based classification," Queen Mary and Westfield College, Department of Computer Science, Technical Report ,2005

[43] Nguyen, T. T., et Armitage, G. (2008). « A survey of techniques for internet traffic classification using machine learning ». IEEE Communications Surveys & Tutorials, 10(4), 56-76

[44] Y. Bennani, E. Viennet, and S. Guérif. Réduction des dimensions des données en apprentissage artificiel. Revue des Nouvelles Technologies de l'Information (RNTI-A2), pages 135--163, March 2008

[45] Chouaib, Hassan & Tabbone, Salvatore & Terrades, Oriol & Cloppet, Florence & Vincent, Nicole. (2008). Sélection de caractéristiques à partir d'un algorithme génétique et d'une combinaison de classifieurs Adaboost

[46] Sangeetha, R. and B. Kalpana. "Identifying Efficient Kernel Function in Multiclass Support Vector Machines." (2011)

[47] jupyter notebook https://jupyter.org/

[48] N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," Journal of Artificial Intelligence Research, vol. 16, pp. 321-357, 2002

[49] Guillaume Lemaitre, Illustration of the sample generation in the over-sampling algorithm https://imbalanced-learn.readthedocs.io/en/stable/auto_examples/over-sampling/plot_illustration_generation_sample.html

[50] IOT TRAFFIC TRACES https://iotanalytics.unsw.edu.au/iottraces

[51] Chacon, https://www.chacon.be

[52] blogmotion, [Test] Prise électrique WiFi Chacon 802.11 b/g/n, http://blogmotion.fr/feedback/chacon-prise-wifi-53012-15282

[53] Science et Avenir, Tendance au CES, "les objets connectés sont des passoires côté sécurité" avertit Gérard Berry https://www.sciencesetavenir.fr/high-tech/conso/tendance-au-ces-les-objets-connectes-sont-des-passoires-cote-securite-dit-gerard-berry_109382