UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Méthodes de détection pour la sécurité des systèmes IoT hétérogènes

THÈSE

présentée et soutenue publiquement le 17 janvier 2023

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Adrien Hemmer

Composition du jury

Rapporteurs :	Mohamed Kaâniche	Directeur de Recherche CNRS, LAARS
	Michele Nogueira	Associate Professor, Federal University of Minas Gerais
Examinateur :	Gregory Blanc	Maître de Conférences, Télécom SudParis
Président du jury :	Olivier Perrin	Professeur, Université de Lorraine
Encadrants :	Isabelle Chrisment	Professeure, Université de Lorraine (Loria) (Directrice de thèse)
	Rémi Badonnel	Professeur, Université de Lorraine (Loria) (Co-directeur de thèse)

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Sommaire

Chapitre 1
Introduction générale **1**

1.1 Contexte 1
1.2 Problématique 3
1.3 Contributions 3
1.4 Organisation de la thèse 4

Chapitre 2
Sécurisation et monitorage dans les infrastructures IoT **7**

2.1 Introduction 7
2.2 Systèmes IoT 8
2.2.1 Définition et propriétés 8
2.2.2 Principales attaques 11
2.2.3 Sécurisation des systèmes IoT 13
2.3 Méthodes de détection d'attaques 15
2.3.1 Détection par signature 15
2.3.2 Détection des anomalies par apprentissage 16
2.3.3 Détection des anomalies par utilisation du *process mining* 26
2.4 Application de deux algorithmes de *process mining* 27
2.4.1 Algorithme de l'inductive miner 27
2.4.2 Algorithme du transition system miner 33
2.5 Synthèse 36

Chapitre 3
Approche de détection utilisant le *process mining* pour l'IoT **37**

3.1 Introduction 37
3.2 Architecture 38
3.2.1 Principaux composants et interactions 38

3.2.2 Collecte des données . . . 38
3.2.3 Modélisation des données d'entrée . . . 39
3.2.4 Description des modèles de comportement utilisés par le *process mining* . 40
3.3 Phase d'apprentissage . . . 41
3.3.1 Normalisation des données numériques . . . 41
3.3.2 Partitionnement de données . . . 43
3.3.3 Découpe temporelle . . . 45
3.3.4 Construction des modèles de comportement . . . 46
3.4 Phase de détection . . . 47
3.4.1 Normalisation des données numériques . . . 47
3.4.2 Regroupement par la distance euclidienne . . . 48
3.4.3 Découpe temporelle . . . 49
3.4.4 Méthode d'alignement d'un modèle avec les données . . . 49
3.4.5 Mécanisme de détection d'anomalies . . . 50
3.5 Synthèse . . . 52

Chapitre 4
Évaluation de l'approche de détection utilisant le *process mining* 53

4.1 Introduction . . . 53
4.2 Description des jeux de données . . . 54
4.2.1 Description qualitative des jeux de données . . . 54
4.2.2 Véhicules connectés . . . 55
4.2.3 Industrie 4.0 . . . 56
4.2.4 Robots d'assistance . . . 58
4.3 Performance du *process mining* pour la détection d'attaque . . . 59
4.3.1 Impact des paramètres de configuration sur la détection . . . 59
4.3.2 Évaluation de la phase de détection . . . 62
4.4 Comparaison à d'autres méthodes de détection . . . 70
4.4.1 Description des méthodes . . . 71
4.4.2 Comparaison des performances de détection . . . 73
4.4.3 Conséquences de la présence de perturbations sur la détection . . . 78
4.5 Synthèse . . . 80

Chapitre 5
Adaptation ensembliste de l'approche de détection 82

5.1 Introduction . . . 82
5.2 Architecture utilisant l'apprentissage ensembliste . . . 83

5.2.1 Modélisation des données d'entrée pour les systèmes complexes 83
5.2.2 Adaptation de l'architecture précédente à l'apprentissage ensembliste . . . 85
5.3 Construction d'un graphe de dépendance 86
5.3.1 Contexte 86
5.3.2 Coefficient de Pearson 87
5.3.3 Construction manuelle par un expert 88
5.4 Apprentissage ensembliste 88
5.4.1 Choix des méthodes de détection 88
5.4.2 Définition des poids 89
5.5 Mécanisme de retour adaptatif 91
5.5.1 Fenêtre glissante temporelle 91
5.5.2 Adaptation de la taille de la fenêtre glissante 92
5.6 Synthèse 93

Chapitre 6
Évaluation de l'approche ensembliste **94**

6.1 Introduction 94
6.2 Description des jeux de données 95
6.3 Évaluation des performances de l'architecture 96
6.3.1 Construction du graphe de dépendances 97
6.3.2 Apprentissage ensembliste 98
6.3.3 Mécanisme de retour adaptatif 100
6.4 Influence des perturbations sur les performances de détection 103
6.4.1 Données bruitées 103
6.4.2 Perte de données 105
6.5 Synthèse 108

Chapitre 7
Développement et intégration du prototype **110**

7.1 Introduction 110
7.2 Implémentation de la solution de détection 111
7.2.1 Transformation des données d'entrée 112
7.2.2 Application du *process mining* 113
7.2.3 Adaptation de la solution à l'apprentissage ensembliste 115
7.3 Conteneurisation de la solution de détection 116
7.3.1 Explicitation des entrées et des sorties de la phase d'apprentissage 116
7.3.2 Explicitation des entrées et des sorties de la phase de détection 119

7.3.3 Choix des paramètres de pré-traitement 121
7.4 Intégration de l'approche à la plateforme SecureIoT 123
7.4.1 Présentation de la plateforme SecureIoT 123
7.4.2 Déploiement d'une API REST 126
7.5 Synthèse 128

Chapitre 8
Conclusions et perspectives **129**

8.1 Bilan des travaux réalisés 129
8.2 Publications 131
8.3 Perspectives de recherche 132

Glossaire **133**

Table des figures **136**

Liste des tableaux **140**

Bibliographie **141**

Chapitre 1

Introduction générale

Sommaire

1.1 Contexte 1
1.2 Problématique 3
1.3 Contributions 3
1.4 Organisation de la thèse 4

1.1 Contexte

Depuis plusieurs années, l'Internet des Objets (IoT) s'est grandement développé dans un nombre important de domaines d'applications, tels que ceux relatifs aux villes intelligentes, à l'industrie 4.0, à l'énergie, aux transports, à la santé, ou encore à l'agriculture. Ce paradigme repose sur l'interconnexion d'objets connectés à la frange de l'Internet, qui permettent de collecter, traiter des informations sur les environnements physiques, pour acquérir de nouvelles connaissances sur ceux-ci, et améliorer leur fonctionnement en prenant les décisions les plus pertinentes possibles [27]. Par exemple, une ville intelligente, composée d'une multitude de capteurs de différente nature, peut avoir parmi ses objectifs de prévenir la création des bouchons routiers. Pour atteindre cet objectif, nous pouvons utiliser, entre autres, les téléphones des conducteurs, les caméras de surveillance et les panneaux de circulation connectés. Ces appareils sont amenés à utiliser des protocoles de communication variés. De plus, certains de ces objets, notamment les téléphones des conducteurs, peuvent rejoindre ou quitter le réseau à tout moment, ce qui complexifie le fonctionnement du système. Aussi, la taille et la dynamique croissantes de ces systèmes, construits à partir d'un ensemble d'objets connectés hétérogènes, les amènent à être une cible particulièrement vulnérable aux attaques de sécurité. Leurs ressources limitées et certains choix de conception des systèmes IoT contribuent également à ces problèmes de sécurité [25, 91]. En plus de cela, les cyber-attaques gagnent en maturité et peuvent tirer profit de la complexité de ces systèmes.

La faiblesse des systèmes IoT ne concerne pas uniquement les objets connectés, mais peut avoir un impact sur l'ensemble des équipements reliés au réseau. Leurs ressources peuvent en effet être utilisées pour réaliser des attaques qui ne ciblent pas uniquement les objets. C'est notamment le cas des botnets constitués d'objets infectés qui permettent de réaliser des attaques de déni de service distribué. Un exemple emblématique est l'une des attaques réalisées par un botnet d'objets infectés par le malware Mirai en octobre 2016. Cette attaque, réalisée à partir de caméras

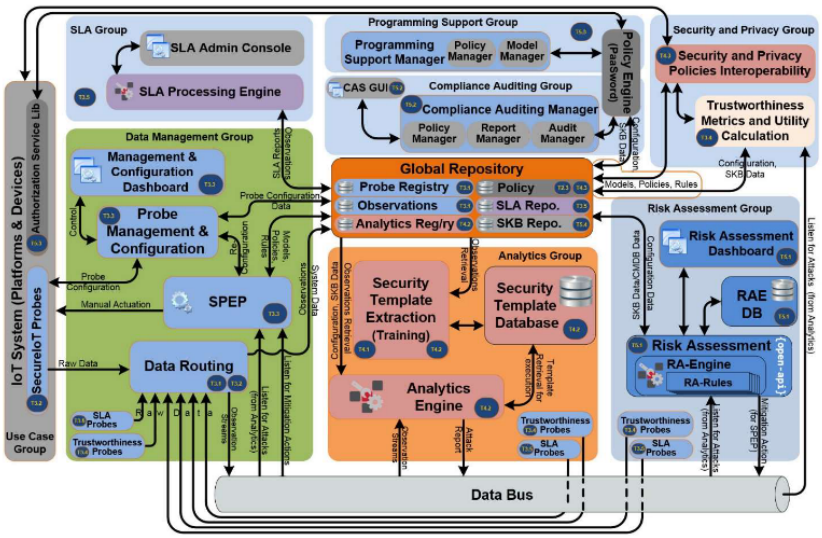


FIGURE 1.1 – Plateforme de sécurité utilisée dans le projet européen SecureIoT [2]

connectées à l'encontre du service DNS de l'entreprise DYN, fut responsable de l'indisponibilité, pendant plusieurs heures, de diverses plateformes et services majeurs fournis par l'Internet [91]. En France, environ 10 millions d'attaques par déni de service distribué, provenant la plupart du temps d'équipements IoT, ont été recensées pour l'année 2021 [13]. Dans un contexte de hausse permanente des menaces cyber, cette thèse contribue à améliorer la sécurisation des systèmes IoT et s'est inscrite dans le cadre du projet européen SecureIoT [2], dont le but était de construire une plateforme de sécurité pour les objets connectés qui garantisse la modularité, l'extensibilité, la configurabilité ainsi que le passage à l'échelle. Cette plateforme repose une évaluation de la confiance des éléments (objets connectés) d'un système IoT, qui s'appuie sur cinq métriques [127] :

— la sécurité qui correspond au niveau de protection contre, notamment, les accès non autorisés dont bénéficie chaque objet connecté,

— la confidentialité qui définit quelles informations concernant chacun des objets peuvent être transmises, collectées ou traitées et par quelle entité,

— la résilience qui est la capacité d'un élément d'un système à fonctionner de manière acceptable face à une perturbation,

— la fiabilité qui caractérise la capacité d'un système à fonctionner normalement dans des conditions spécifiques pendant une période de temps,

— la sûreté qui évalue les possibilités pour chacun des équipements de fonctionner sans mettre en danger les utilisateurs proches ou d'avoir de graves conséquences sur l'environnement dans lequel il évolue.

L'accès à certaines ressources ou services est ainsi conditionné par les valeurs associées à ces métriques. En effet, on va par exemple demander à un objet voulant interagir avec un élément critique pouvant avoir un impact important, de présenter une sécurité élevée. La plateforme de sécurité, illustrée par la figure 1.1, se fonde en particulier sur un monitorage et une analyse des données produites par les objects connectés [2]. Elle s'inspire de modèles existants, comme l'Industrial Internet Security Framework (IISF) [141] et l'OpenFog Reference Architecture (ORA) [117], qui possèdent également des modules d'analyse des données, et viennent en complément des autres mécanismes de sécurité (comme l'authentification des objets). La figure

précédemment mentionnée décrit les principaux modules de la plateforme et leurs relations. Les objets connectés envoient leurs informations au module de gestion des données ("Data Management Group") qui se charge de les mettre à disposition des autres modules dans un répertoire global ("Global Repository"). Les travaux de cette thèse portent sur le module d'analyse ("Analysis Group"), qui interagit avec le répertoire global et le module de gestion des risques ("Risk Assessment Group"). Concrètement, les données émises par les objets connectés sont collectées et formatées par le module de gestion des données. Une fois les données formatées, elles sont placées dans le répertoire global où elles sont récupérées pour effectuer des analyses. Ces analyses nous permettent notamment d'identifier des anomalies et caractériser des attaques. Les alertes éventuelles sont transmises au module de gestion des risques. Les autres modules, comme celui du service support à la programmation ("Programming Support Group"), de vérification de la conformité ("Compliance Auditing Group"), ou bien encore des accords sur les niveaux de service ("SLA Group") viennent en soutien pour la mise en place de la plateforme de sécurité, et contribuent à l'intégration des objets connectés, ainsi qu'à la définition des politiques d'accès.

1.2 Problématique

Cette thèse vise à définir de nouvelles méthodes de détection pour la sécurité des systèmes IoT hétérogènes. Ces méthodes doivent permettre à la fois de modéliser efficacement le comportement des systèmes IoT et de mettre en évidence les comportements anormaux d'objets connectés. Plus spécifiquement, elles doivent répondre aux défis suivants :

— hétérogénéité des données et des protocoles : la grande diversité introduite par les objets connectés exige que nos méthodes puissent prendre en compte des données et des protocoles hétérogènes. En particulier, des traitements sont nécessaires pour s'assurer qu'elles soient correctement intégrées et qu'elles permettent d'obtenir les meilleurs résultats de détection sur des attaques qui peuvent elles-mêmes utiliser cette hétérogénéité pour se camoufler,

— détection au plus tôt des attaques cyber : la détection des attaques doit avoir lieu le plus tôt possible pour permettre de réduire leurs impacts potentiels sur les systèmes IoT. En particulier, les premières phases de reconnaissance qui peuvent être opérées par les attaquants doivent être identifiées rapidement. Les méthodes doivent être aussi adaptatives, et capables d'ajuster leur sensibilité en fonction du contexte.

— aide à la prise de décision : les modèles de comportement ainsi que les alertes doivent être autant que possible précis pour permettre une meilleure compréhension par l'administrateur ou l'opérateur, et afin de permettre une exploitation efficace pour la sélection des contre-mesures qui seront appliquées.

1.3 Contributions

Les contributions de cette thèse s'articulent en trois contributions majeures. Pour notre première contribution, nous avons formalisé et implémenté une méthode de détection reposant sur le *process mining* [77] pour la caractérisation d'anomalies dans des données hétérogènes issues de systèmes IoT. Lors de la phase d'apprentissage, cette méthode génère des modèles de comportement sous la forme de réseaux de petri, qui prennent en compte l'évolution des systèmes, et permettent de mieux comprendre leur fonctionnement. La détection des anomalies se fait en comparant les modèles de comportement normal avec d'autres jeux de données provenant de ces systèmes. Ce choix de l'utilisation du *process mining* intervient à la suite de l'état de l'art sur la

sécurisation et le monitoring dans les infrastructures IoT. Nous avons ensuite comparé les performances de notre méthode à celles d'autres méthodes courantes de détection des anomalies [76], comme notamment la méthode One Class SVM (OCSVM) et la méthode One Class Random Forest (OCRF). Nous avons également testé la robustesse de notre solution, lorsque les données sont bruitées ou sont caractérisées par des pertes.

Notre seconde contribution porte sur l'adaptation ensembliste de notre approche de détection. Les performances des méthodes de détection peuvent être très variables en fonction de la nature des données, notamment sur des systèmes IoT hétérogènes. Nous avons donc proposé d'intégrer plusieurs méthodes de détection en suivant une approche ensembliste pour détecter les attaques cyber [75]. L'utilisation conjointe de ces méthodes a pour but d'améliorer les performances de détection de notre solution pour des données hétérogènes. Cette contribution comporte également un mécanisme adaptatif permettant d'ajuster la réactivité de la détection en fonction des résultats de certaines méthodes. Ainsi, cette solution est capable d'être automatiquement plus sensible aux variations sur les données pour réduire globalement le temps de détection.

Afin de concrétiser notre approche et mener à bien les différentes expériences, notre troisième contribution a porté sur la réalisation d'un prototype de la solution de détection, qui a été intégré dans la plateforme de sécurité du projet SecureIoT. Ce prototypage a été mis en oeuvre en prenant en compte des contraintes de modularité et d'intégration. Il permet d'analyser les données collectées depuis un système IoT hétérogène, et de générer des alertes transmises à un module de gestion des risques. Son fonctionnement a été évalué à l'aide de multiples jeux de données issus de différents systèmes IoT hétérogènes.

1.4 Organisation de la thèse

Le plan du manuscrit est organisé selon les contributions évoquées précédemment, et est composé d'un total de 8 chapitres, dont les principaux sont décrits avec leurs relations sur la figure 1.2. Il débute par un état de l'art sur la sécurisation et le monitorage dans les infrastructures IoT, indiqué en jaune clair sur la figure. Ce chapitre aborde les propriétés de tels systèmes, ainsi que les principales attaques auxquelles ils sont confrontés. Il évoque ensuite les différentes solutions de sécurisation, avec un focus sur les grandes catégories de méthodes de détection d'attaques, et fournit des éléments détaillés sur les algorithmes de *process mining*, qui seront ensuite utilisés dans le cadre de nos travaux.

Le premier chapitre de contribution à proprement parler, intitulé approche de détection utilisant le *process mining* pour l'IoT, apparaît en bleu clair, et formalise notre première solution consistant à exploiter et adapter le *process mining* pour détecter des anomalies dans les systèmes IoT hétérogènes. La difficulté de mise en oeuvre vient du fait que le *process mining* peut rapidement être confronté à une explosion d'états avec ces systèmes. Nous avons proposé d'intégrer des méthodes de pré-traitement, qui permettent un partitionnement préalable des données, et réduisent ce phénomène. Le *process mining* présente dès lors deux avantages majeurs. Il prend en considération l'enchaînement des données, et non pas juste le positionnement des points de données dans l'espace vectoriel défini par leurs attributs. Il permet de créer des modèles de systèmes IoT plus facilement compréhensibles. Nous montrons ensuite dans le second chapitre présenté en bleu clair comment la méthode de détection est appliquée à des jeux de données hétérogènes issus de systèmes IoT hétérogènes. Nous décrivons en particulier le fonctionnement de la phase d'apprentissage et de la phase de détection de notre méthode, en l'appliquant sur les données fournies par les partenaires industriels du projet SecureIoT. Nous comparons notamment les performances avec celles d'autres méthodes couramment utilisées dans la détection des anomalies.

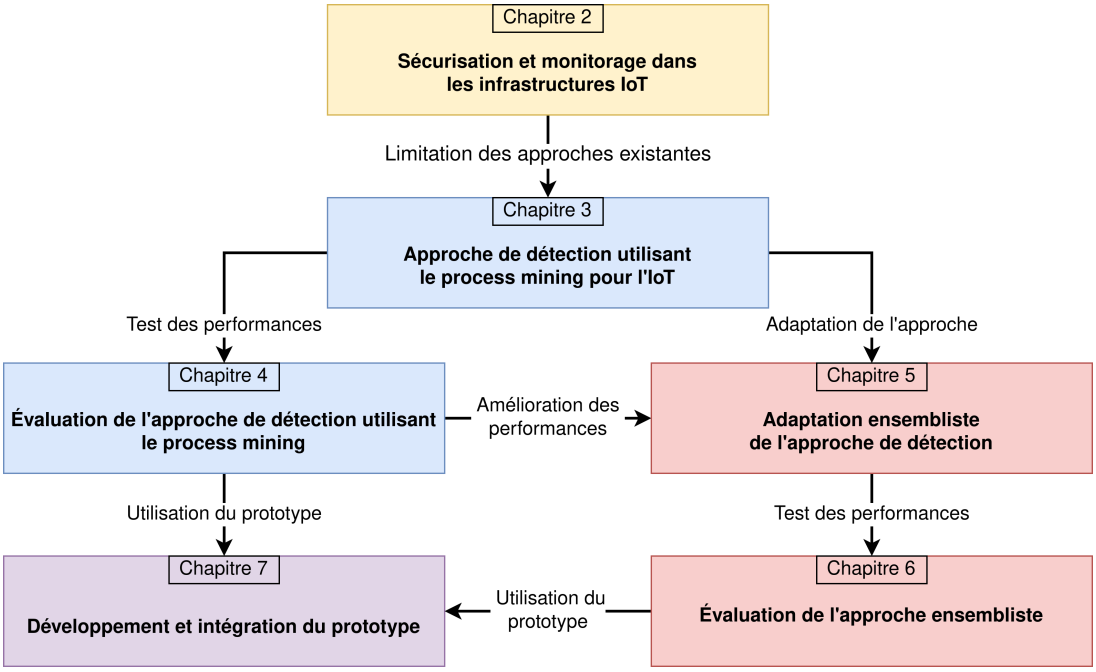


FIGURE 1.2 – Principaux chapitres du manuscrit et leurs relations

Enfin, nous testons la robustesse de notre solution au bruit et à la perte de données.

Nous poursuivons le manuscrit avec une amélioration ensembliste de notre solution, correspondant à notre seconde contribution, et représentée par les chapitres en rouge clair. Nous décrivons tout d'abord notre méthode ensembliste dont l'objectif est d'améliorer le temps de détection en intégrant plusieurs méthodes de détection. Elle permet de détecter les premiers signes d'une attaque. Nous y détaillons quatre stratégies d'agrégation des scores de détection permettant sa mise en oeuvre. Nous détaillons aussi le fonctionnement d'un mécanisme de retour adaptatif qui s'appuie sur l'utilisation d'une fenêtre glissante, et permet d'automatiquement ajuster la réactivité de la détection en fonction des résultats partiels. La décision de modifier la taille de cette fenêtre est typiquement prise lors de la rencontre d'une alerte de sécurité caractérisant les prémisses d'une attaque ou ses premières phases. Notre méthode exploite un graphe de dépendances, qui spécifie les liens entre les différentes sources de données à partir desquelles est réalisée la détection. Il permet de déterminer les autres sources de données sur lesquelles la sensibilité de détection va être accrue. Nous développons ensuite dans un chapitre d'évaluation des performances, les résultats obtenus à l'aide des quatre stratégies d'agrégation utilisées pour notre approche ensembliste. Nous évaluons aussi l'impact du changement de la taille de fenêtre glissante sur le temps de détection lors d'attaques complexes composées de plusieurs étapes. Finalement, sur le même modèle qu'avec la première méthode, nous avons testé la robustesse de l'adaptation ensembliste face au bruit et à la perte de données.

Enfin, dans le chapitre indiqué en violet et intitulé développement et intégration du prototype, nous décrivons les travaux de prototypage de notre solution, qui ont notamment servi pour les expérimentations et les démonstrations. Nous détaillons la conception et l'algorithmique sous-jacente du prototype s'appuyant sur des bibliothèques dédiées, sa conteneurisation pour un usage modulaire en utilisant l'environnement docker, et la création d'une interface REST permettant une intégration facilitée dans la plateforme du projet SecureIoT. Nous terminons le manuscrit en faisant un bilan des différentes contributions relatives à ces travaux. Nous mettons également en

avant plusieurs perspectives concernant l'intégration de nouvelles méthodes de détection à notre solution ensembliste, l'automatisation de la sélection de contre-mesures, ainsi que l'exploitation de bases de connaissances additionnelles en cybersécurité, dans le cadre de démarches de *threat intelligence*.

Chapitre 2

Sécurisation et monitorage dans les infrastructures IoT

Sommaire

2.1 Introduction 7
2.2 Systèmes IoT 8
2.2.1 Définition et propriétés 8
2.2.2 Principales attaques 11
2.2.3 Sécurisation des systèmes IoT 13
2.3 Méthodes de détection d'attaques 15
2.3.1 Détection par signature 15
2.3.2 Détection des anomalies par apprentissage 16
2.3.3 Détection des anomalies par utilisation du *process mining* 26
2.4 Application de deux algorithmes de *process mining* 27
2.4.1 Algorithme de l'inductive miner 27
2.4.2 Algorithme du transition system miner 33
2.5 Synthèse 36

2.1 Introduction

Les systèmes et applications utilisant des objets connectés introduisent de nouvelles vulnérabilités dans les systèmes d'information. Dans ce chapitre, nous définissons ce que sont les systèmes IoT et comment leurs propriétés peuvent expliquer les principales attaques dont ils sont victimes. Nous adoptons le classement proposé par l'Open Web Application Security Project (OWASP) spécialement pour l'IoT [7], qui donne les 10 vecteurs d'attaques les plus couramment utilisés. Nous présentons différentes attaques qui leur sont associées. Nous discutons de la distinction entre les attaques externes, où un individu essaye d'obtenir les mêmes droits qu'un utilisateur autorisé, des attaques internes, où l'attaquant a déjà accès à au moins une partie du système. Nous résumons ensuite les travaux de recherche effectués pour réduire l'impact des différents types d'attaques. Nous justifions l'importance des méthodes de détection en complément des autres solutions de sécurité. Nous décrivons ensuite plusieurs méthodes de détection des anomalies. Finalement, nous expliquons plus en détail le principe de fonctionnement du *process mining* avant d'illustrer par un exemple le fonctionnement de deux de ses algorithmes, l'*inductive miner* et le *transition system miner*

2.2 Systèmes IoT

L'utilisation d'objets connectés augmente la surface d'attaque des systèmes d'information en y ajoutant de nouveaux points d'entrée dont peuvent se servir des entités malveillantes. Il est donc nécessaire de protéger au mieux ces éléments d'autant plus qu'ils introduisent souvent de nouvelles vulnérabilités dans les systèmes. Au cours du projet européen SecureIoT, nous avons été amenés à travailler à la fois sur des systèmes IoT avec le robot d'assistance, mais également sur des systèmes IIoT (Industrial Internet of Things) avec l'industrie 4.0. Dans la littérature, le choix de la terminologie employée est lié au contexte d'application. Ainsi, une solution de sécurité dans un contexte industriel est référencée dans la catégorie IIoT même si la solution est aussi utilisable pour des systèmes IoT classiques. Dans cette section, nous définissons ce que sont les systèmes IoT et décrivons quelles sont leurs caractéristiques dans le milieu industriel. Puis, nous expliquons quels sont les principaux problèmes de sécurité de ces systèmes et les illustrons avec des exemples d'attaques. Finalement, nous examinons les solutions de sécurité applicables aux systèmes IoT.

2.2.1 Définition et propriétés

D'après [10], un système IoT est composé de trois éléments principaux comme indiqué dans la figure 2.1. Le premier est un ensemble hétérogène d'objets intelligents qui collaborent entre eux, via Internet [118], afin de fournir divers services. Le second élément est l'application IoT qui va chercher à améliorer le rendement en utilisant sa connaissance de l'état du système. Elle est composée d'un ensemble de services qui analysent les données remontées par les objets connectés pour prendre des décisions, le plus souvent, à l'aide d'algorithmes d'apprentissage. Finalement, le dernier élément est la présence d'une interface graphique qui doit permettre à des utilisateurs de pouvoir avoir une visibilité sur l'état du système, et d'interagir avec celui-ci afin d'éventuellement prendre des décisions manuellement.

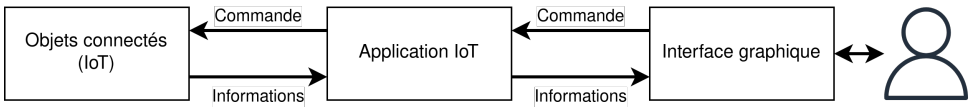


Figure 2.1 – Composition d'un systèmes IoT

Dans [34], les auteurs énumèrent six caractéristiques propres aux systèmes IoT, figure 2.2, et pouvant compromettre leur sécurité. La première d'entre elles, symbolisée par l'utilisation de smartphones, est le flou concernant le périmètre des systèmes IoT avec les changements induits par le remplacement d'appareils ou le déplacement des utilisateurs. La seconde est l'hétérogénéité importante des protocoles, des plateformes et des appareils IoT en général, ainsi que des moyens de communication, avec la possibilité d'utiliser, en autres, le Bluetooth, la 4G, ou bien encore le wifi (cf. figure 2.2).

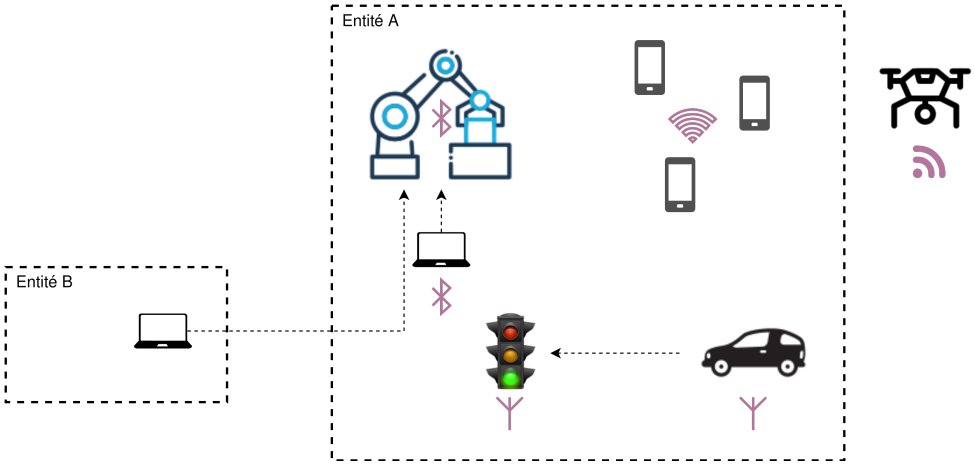


FIGURE 2.2 – Caractéristiques propres aux systèmes IoT

La troisième caractéristique est l'autonomie laissée à certains systèmes IoT avec la possibilité pour un objet de directement contrôler d'autres objets connectés. Dans notre exemple, le poste de contrôle maîtrise la chaîne de production. Nous pouvons également remarquer que les véhicules contrôlent les feux de signalisation au sein de l'entité A. Le quatrième point développé par les auteurs est la possibilité que les objets connectés exposent des éléments du système qui ne devraient pas interagir avec l'extérieur. C'est le cas de certaines machines qui ne possèdent pas de service d'authentification car elles sont censées fonctionner localement en dehors du réseau Internet. L'intégration de telles machines dans un système IoT représente un important risque de sécurité. En effet, si l'entité B de la figure 2.2 était une personne malveillante ayant réussi à accéder au système déployé par l'entité A, elle pourrait contrôler complètement la chaîne de production. Le cinquième facteur de risque concerne l'accès physique. Il est tout à fait envisageable que des parties du système soient exposées physiquement à l'extérieur ou qu'une entité ait le contrôle total de plusieurs équipements du système IoT. Dans notre exemple, cela est illustré, respectivement, par un drone et par le lien entre l'entité B et la machine de production. Finalement, le dernier point à prendre en compte pour la sécurité des systèmes IoT est que, contrairement aux smartphones, il n'est pas toujours possible de gérer les permissions des différents objets connectés. Ainsi, il n'est pas toujours évident de limiter les accès de certaines parties du système à des objets et de laisser un accès total à d'autres.

Dans la littérature, nous trouvons plusieurs travaux sur les systèmes IoT, dans des domaines d'application totalement différents, comme le trafic automobile [164] ou bien encore le suivi médical de patients [50]. Pour illustrer plus en détail la pertinence de l'utilisation de systèmes IoT, nous considérons le cas d'une maison intelligente qui va chercher à diminuer la consommation d'énergie ou de ressources. Pour cela, il est intéressant d'installer des objets connectés comme des prises de courant connectées couplées à des capteurs de présence. Ainsi, l'application IoT pourra éteindre automatiquement la lumière des pièces vides. Il est également possible d'avertir le propriétaire de ces maisons lorsque des fuites de gaz ou d'eau sont détectées par des capteurs spécifiques.

Le recours aux systèmes IoT s'est démocratisé, avec en 2022 plus de 14 milliards d'objets connectés déployés [74], car ils permettent à la fois de visualiser et de contrôler en temps réel l'état d'un système. Ce nombre est plutôt dans la fourchette basse des estimations données entre 2010 et 2015. En effet, à cette période, certaines entités estimaient possible la présence d'une centaine de milliards d'objets connectés [157] pour 2020. Toutefois, il est fort probable que la

pénurie de puces électroniques de ces dernières années couplée aux problèmes de sécurité des IoT [5] ait freiné leur développement.

Dans le cadre du projet européen, nos travaux ont également été utilisés par un système IIoT avec la présence de partenaires industriels dont les usines fonctionnent sur le concept de l'industrie 4.0. Les systèmes IIoT sont une spécification dans le domaine industriel des systèmes IoT. Nous avons repris dans le tableau 2.1 les différences entre l'IoT et l'IIoT [4]. Nous pouvons observer que, de manière générale, pour l'IoT, la protection des données personnelles est le point le plus important, là où pour l'IIoT ce sera l'absence de pannes dans le système. Cela se traduit pour le premier par une priorisation de la confidentialité et de l'intégrité, tandis que pour le second, c'est la disponibilité et l'intégrité qui seront les propriétés les plus importantes. Pour résumer les autres points du tableau, les objets connectés des systèmes IoT possèdent une faible durée de vie et évoluent dans des environnements classiques, ce qui leur permet d'être facilement mis à jour. De plus, en cas de compromission, ces objets seront désactivés dans la majeure partie des cas. Pour l'IIoT, nous avons plutôt des éléments à grande durée de vie déployés dans des environnements hostiles. De plus, la mise à jour de ces derniers, par exemple en cas de découverte d'une importante faille de sécurité, n'est pas aussi évidente et doit être programmée. Il est également à noter que contrairement à la majorité des cas dans l'IoT, un mauvais comportement du système peut avoir de graves conséquences à la fois sur la production, mais également sur la sécurité des opérateurs humains proches.

Caractéristiques	Internet of Things	Industrial Internet of Things
Objectif	Protection des données personnelles et du matériel	Absence de panne et sûreté
Priorités	Confidentialité, intégrité et disponibilité	Disponibilité, intégrité et confidentialité
Conséquences d'un mal-fonctionnement	Aucune conséquence critique	Interruption du processus, impact sur la production, menaces physiques possibles
Réaction contre une menace	Arrêt possible puis restauration de l'équipement	Programmation d'une maintenance
Gestion des mises à jour	Possible à tout moment	Nécessite d'être réalisée lors d'un arrêt et doit donc être prévue
Cycle de vie	Changement et amélioration de l'équipement fréquent	Équipements à longue durée de vie (plus de 15 ans)
Conditions de déploiement	Environnement standard	Environnement hostile (température, vibration, etc)

Table 2.1 – Tableau récapitulatif des différences entre IoT et IIoT [4]

Dans la majorité des travaux de recherche menés pour améliorer la sécurité des objets connectés, le choix de la terminologie est déterminé par le type de système sur lesquels sont menées les expériences. En effet, les principales différences entre ces deux types de systèmes sont le plus souvent d'ordre contextuel et font appel au même lexique [146]. Ainsi, avec des défis très similaires à relever pour ces deux paradigmes [163], la plupart des solutions de sécurité labélisées pour l'IoT sont tout à fait applicables à l'IIoT et inversement. Dans la suite du manuscrit, nous ferons référence au système IoT dans lequel nous incluons également les systèmes IIoT.

2.2.2 Principales attaques

Comme nous avons pu le voir précédemment, les systèmes et les applications utilisant des objets connectés peuvent présenter de grands risques de sécurité. Les faiblesses utilisées par les attaquants ont été caractérisées par l'Open Web Application Security Project (OWASP) spécialement pour l'IoT [7]. Nous y trouvons les dix vecteurs d'attaques les plus critiques de ces systèmes. Ils sont une conséquence logique de la complexité des systèmes IoT qui utilisent des protocoles et des plateformes hétérogènes. De plus, les ressources limitées des objets connectés en font des cibles de choix. L'augmentation des objets connectés dans les systèmes augmente également la surface d'attaque et la probabilité des attaquants de trouver une porte d'entrée. Il est également fréquent que l'authentification de ces objets soit celle par défaut ou que les mises à jour de sécurité ne soient pas faites. Le tableau 2.2 liste, dans sa première colonne, l'ensemble de ces vecteurs d'attaques qui vont des mots de passe facilement trouvables au manque de protection physique de certains équipements.

Les attaques les plus médiatisées sont, dans la majeure partie des cas, les conséquences d'attaques externes où l'attaquant vise, le plus souvent, à acquérir les mêmes droits qu'un utilisateur autorisé sur le système. Nous retrouvons, dans le cas du malware Mirai [9], la première catégorie du classement OWASP dédié aux IoT, qui estime que l'utilisation de mots de passe de faible complexité ou facilement trouvables est le problème de sécurité le plus critique. Ce malware a infecté jusqu'à près de 600 000 objets connectés en octobre 2016, dont une majorité de caméras, pour lancer des attaques collectives par saturation de services pour attaquer les serveurs OVH et faire tomber les serveurs DNS de l'entreprise Dyn. Une étude de cette attaque analyse l'émergence du botnet Mirai ainsi que son évolution [25]. La conclusion des auteurs est que son apparition est principalement due à l'absence de bonnes pratiques de sécurité dans le domaine IoT. En effet, ce botnet a infecté en priorité des objets connectés en utilisant le mot de passe par défaut du fabricant. Plus récemment, une attaque similaire a eu lieu en 2019 [154] avec une attaque d'applications proposant des flux vidéo en direct regroupant près de 400 000 objets connectés. Ces deux attaques ne sont pas des exceptions, avec plus de 9 millions d'attaques collectives par saturation de services recensées en 2021 [3, 13].

Il arrive également que la maîtrise d'un objet connecté donne à l'attaquant un accès au système IoT. Dans cette configuration, il peut être le vecteur d'entrée de la prise de contrôle de l'ensemble du système. Avec ces considérations, nous pouvons donner un exemple de la seconde catégorie de la classification OWASP avec un réseau non sécurisé. En effet, il est possible d'extraire l'identifiant d'un réseau Z-Wave [139], qui est l'une des technologies de communication sans-fil standard dans la domotique [12]. Une fois l'attaquant authentifié sur le réseau, il a la possibilité de prendre le contrôle de l'ensemble des éléments de celui-ci en utilisant, par exemple, un contrôleur universel.

Les exemples que nous avons évoqués jusqu'à présent font tous partie de la catégorie des attaques externes, toutefois, les attaques internes sont loin d'être négligeables. Contrairement aux attaques externes, l'attaquant a accès à une partie du système et peut, par exemple, chercher à obtenir de nouveaux droits. Dans [121], les auteurs ont proposé une liste des attaques qu'ils envisagent pour chacun des 10 principaux vecteurs d'attaques dans l'IoT. Nous montrons dans le tableau 2.2 une association entre ces 10 vecteurs d'attaques, dans la première colonne, et les différentes attaques qui lui sont associées, dans la seconde colonne.

Nous rencontrons aussi bien des attaques par force brute sur les mots de passe et identifiants, des injections SQL ou bien encore des falsifications de requêtes. Nous trouvons dans les travaux de recherche des applications de ces différentes attaques à la fois dans le cas externe mais également interne. Par exemple, un enregistreur de frappe peut être installé physiquement sur une machine

avec une clé USB ou par le biais d'un fichier contenant un logiciel malveillant.

Dans ce manuscrit, nous avons fait le choix d'axer nos recherches sur des méthodes pouvant détecter des attaques appartenant aux deux catégories. Ainsi, nous pourrons à la fois mettre en avant les comportements suspects détectés au niveau des éléments du système IoT, mais également voir un comportement suspect au niveau de la communication entre notre système et l'extérieur.

Vecteurs d'attaques		Attaques
1	Mots de passe faibles, faciles à deviner ou codés en dur	enregistreur de frappe, homme au milieu, force brute, dictionnaire, bourrage d'identifiants, pulvérisation de mots de passe, hameçonnage
2	Services réseau peu sécurisés	écoute de paquets, analyse du trafic, attaque par mascarade, balayage de ping, scan des ports, déni de service, modification de messages, rejeu de messages
3	Interfaces de l'écosystème peu sécurisées	injection SQL, violation de l'authentification, traversée de répertoires, injection de commandes, problème de logique, contrefaçon de requêtes côté serveur, entités externes XML, script inter-sites, falsification de requêtes inter-sites, attaque de WebSocket
4	Absence de système de mise à jour sécurisé	modification du processus de démarrage, accès direct à la mémoire, contournement de la sécurité, exploit au niveau du processeur, attaque lors de la mise à jour du logiciel, piratage du logiciel, élévation des privilèges
5	Utilisation de composants obsolètes	recherche des faiblesses sur Shodan, exploitation des failles non corrigées, utilisation des codes d'exploit accessibles au public
6	Faible protection de la vie privée	écoute clandestine, accès non autorisé, ingénierie sociale, distorsion de l'information, vol d'identité, sabotage
7	Transfert et stockage des données peu sécurisés	vol d'identité, fraude, attaque de redirection, contournement SSL, vol de propriété intellectuelle, vol d'informations sensibles
8	Absence de gestion des appareils connectés	exploitation des failles non corrigées, attaque lors de la mise à jour du logiciel, attaque par énumération pour la récupération de mot de passe
9	Paramètres par défaut peu sécurisés	prise de contrôle de comptes, exploitation de failles non corrigées, élévation des privilèges, exposition des services administrateur
10	Absence de protection physique	accès physique aux appareils pour les endommager, désactiver, voler ou utiliser le système à des fins malveillantes

TABLE 2.2 – Tableau récapitulatif des différentes attaques associées à chaque catégorie OWASP pour les systèmes IoT [7, 121]

Les différents éléments présentés dans cette section montrent l'importance de travailler sur des solutions de sécurité pour les systèmes IoT.

2.2.3 Sécurisation des systèmes IoT

Le tableau 2.3 synthétise les travaux de recherche sur la sécurisation des systèmes IoT [146]. Huit grandes catégories de travaux cherchant à améliorer la sécurité ont été identifiées. Parmi ces catégories, cinq s'adressent en priorité aux questions de confidentialité et d'intégrité des données, tandis que deux autres se focalisent sur la disponibilité du système. La dernière catégorie comporte des travaux cherchant à détecter les anomalies dans les systèmes et peut donc à la fois mettre en lumière les attaques externes ainsi que les attaques internes. Les cinq premières catégories comportent des travaux concernant le partage et la sécurité des données, le contrôle d'accès, l'authentification, la sécurité réseau ainsi que les modèles et méthodes de sécurité. Les sixième et septième catégories contiennent des travaux sur la résilience et la maintenabilité, là où le thème de la huitième catégorie est la surveillance de sécurité.

Il est possible de faire le rapprochement entre les travaux appartenant aux huit catégories, présentées dans le tableau précédent, avec les attaques mentionnées par [121] qui reprennent le top 10 de la classification OWASP [7]. Dans le tableau 2.4, nous avons grisé la case lorsque les travaux de sécurisation de la catégorie considérée permettaient de traiter au moins l'une des attaques. Nous remarquons que la catégorie couvrant le plus de vecteurs d'attaques répertoriés dans la classification OWASP est la huitième catégorie qui correspond à la surveillance de sécurité. Nous y trouvons l'analyse des données et la création de modèles de comportement permettant de mettre en avant les anomalies rencontrées par les systèmes.

	Catégorie	Sous-catégorie	Références
1	Partage et sécurisation des données	Transport des données	[88], [101], [100], [89], [116]
		Éléments extérieurs	[93], [168], [119], [120], [29]
		Contrôle des flux de données	[17], [39], [142]
		Confidentialité des données	[134], [29], [52], [64], [72]
2	Résilience		[144], [123], [26], [31], [37]
3	Maintenabilité	Maintenabilité intelligente	[36], [102], [99], [101], [135]
4	Contrôle d'accès		[33], [156], [103], [106], [29]
5	Authentification	Distribution des clés	[16], [19], [30], [38], [60]
		Authentification mutuelle	[114], [18], [28], [55], [57]
		Non-répudiation	[24], [67], [87], [103], [106]
		Anonymat et confidentialité	[55], [106], [53], [131]
		Attestation	[130], [140], [79], [96], [160]
6	Sécurité réseau	Latence	[73], [82], [90], [97], [115]
		Disponibilité	[20]
		Sans fil	[85], [86], [49], [95], [148]
7	Modèles et méthodes de sécurité		[44], [51], [110], [111], [122]
8	Surveillance de sécurité		[128], [168], [162], [21], [22]

TABLE 2.3 – Tableau récapitulatif de différentes approches de sécurisation des systèmes IoT [146]

En revanche, les travaux appartenant à cette catégorie rencontrent des difficultés pour dé-

tecter l'écoute passive des données d'un système par un attaquant. Ainsi, bien que les travaux de recherche de cette catégorie apportent beaucoup pour sécuriser les systèmes, ils ne sont pas suffisants. C'est pourquoi il est également nécessaire d'y ajouter des mécanismes de chiffrement afin de s'assurer de la confidentialité des données stockées et échangées sur le réseau.

Les solutions architecturales retenues par le projet européen SecureIoT, évoquées dans le chapitre 1, prennent en compte plusieurs critères de sécurité, de performance et de passage à l'échelle. Les mécanismes de sécurité des architectures de sécurité s'adressent le plus souvent aux attaques externes. Ainsi, pour lutter contre celles-ci, des méthodes d'authentification utilisant des techniques cryptographiques sont intégrées au système. Par exemple, il est possible d'utiliser une authentification reposant sur des certificats [130]. Cette solution permet aux noeuds distribués du réseau de vérifier les propriétés de sécurité localement et de modifier de manière fiable les propriétés des certificats dans les noeuds IoT. La gestion des certificats est faite de manière autonome. Il est à noter que, pour certains systèmes, comme par exemple le bus de données CAN (Controller Area Network) des véhicules connectés, il n'existe pas de champ d'authentification [54] et il est donc possible de prendre facilement le contrôle du système. Cette prise de contrôle a pu être mise en avant avec plusieurs scénarios d'attaque allant d'une connexion physique au module embarqué du véhicule à l'exploitation du service d'appel d'urgence [48]. Pour éviter ce type d'attaque, il existe six solutions majeures de chiffrement qui limitent le problème de sécurité du bus CAN [68]. Pour chacune de ces six solutions, respectivement nommées LiBrA-CAN [70], WooAuth [161], Vecure [158], CaCAN [94], VatiCAN [129] et VulCAN [149], plusieurs critères de sécurité ont été évalués. Parmi ces critères on trouve l'intégrité, la confidentialité ou bien encore la possibilité d'utiliser de l'authentification.

classement \ catégorie	1	2	3	4	5	6	7	8
1 Mots de passe faibles	■			■	■		■	■
2 Services réseau peu sécurisés	■	■			■	■		■
3 Interfaces de l'écosystème peu sécurisées					■			■
4 Absence de système de mise à jour sécurisé			■	■				■
5 Utilisation de composants obsolètes			■					■
6 Faible protection de la vie privée	■			■	■			■
7 Transfert et stockage des données peu sécurisés	■			■				
8 Absence de gestion des appareils connectés			■		■			■
9 Paramètres par défaut peu sécurisés				■	■			■
10 Absence de protection physique							■	■

Table 2.4 – Tableau d'association entre les catégories des travaux de sécurisation et la classification des vecteurs d'attaques pour les systèmes IoT proposée par OWASP

Dans le cadre de notre travail, nous nous sommes focalisés sur la détection des anomalies. La mise en place de solutions d'authentification et de chiffrement a été réalisée par un autre de nos partenaires dans le projet SecureIoT.

2.3 Méthodes de détection d'attaques

L'objectif de cette section est de présenter comment ont été utilisés plusieurs mécanismes permettant la détection des attaques. Pour ce faire, nous discutons, tout d'abord, des travaux proposant des méthodes de détection par signature. Ensuite, nous étudions les méthodes d'apprentissage par renforcement ainsi que les méthodes d'apprentissage supervisées. Puis, nous examinons un panel de méthodes semi et non supervisées utilisées pour détecter des attaques. Dans le cas de la détection par apprentissage, les méthodes permettent de mettre en avant les points de données anormaux pouvant être le résultat d'une attaque. Un point de données contient les informations de l'ensemble des différents attributs d'un système pour un temps donné, il représente donc l'état du système à cet instant. Dans le cas du positionnement d'un objet dans l'espace, le point de données donne les valeurs de la largeur, de la profondeur et de la hauteur, ce qui permet de le placer dans l'espace. Finalement, nous regardons comment est utilisé le *process mining* dans la détection des anomalies. La figure 2.3 synthétise notre classification et place les méthodes de détection dont nous allons parler dans cette section.

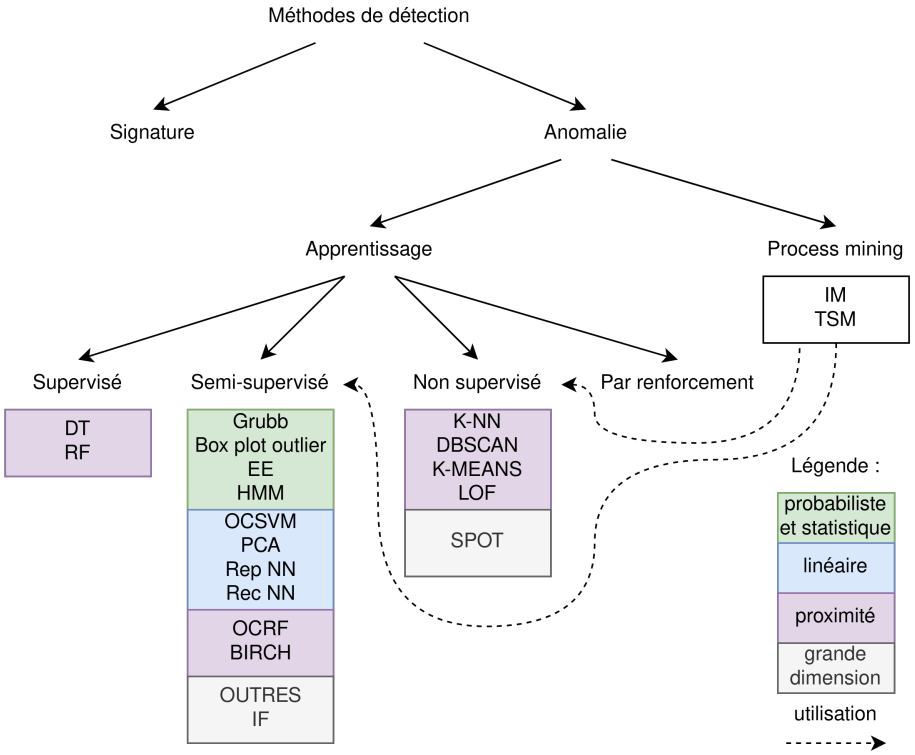


FIGURE 2.3 – Classification des méthodes de détection

2.3.1 Détection par signature

Il existe dans la littérature plusieurs travaux qui reposent sur la détection par signature. Bien que ces types d'approches soient plus précises pour détecter les attaques connues, elles rencontrent des difficultés à en repérer de nouvelles. Dans [80], les auteurs proposent de modéliser des attaques connues sous la forme de graphe. L'objectif est de pouvoir identifier les étapes de l'attaque en cours à l'aide des alertes du système de détection des intrusions. Ainsi, une fois l'étape de l'attaque identifiée, il est possible de prédire son évolution. Concrètement, la reconnaissance des étapes

d'une attaque en cours se fait via les alertes du système de détection des intrusions Snort [138]. Le graphe de l'attaque, représenté par une chaîne de Markov cachée (HMM), permet de prédire la prochaine étape de l'attaque et donc de prendre des contre-mesures adaptées pour limiter ses dommages. Malheureusement, dans ce cas, les modèles d'attaque sont directement créés par des experts et permettent uniquement d'identifier les attaques qu'ils ont modélisées.

La plupart des travaux ne font pas intervenir cette notion d'attaque en plusieurs étapes, comme dans [32] et [84], où les auteurs vont directement chercher à détecter un certain type d'attaque à l'aide de leur système de détection par signature.

Dans le cadre de nos travaux, la plupart des attaques contenues dans les jeux de données présentaient des variations de données importantes et il semblait difficile d'utiliser des systèmes de détection par signature et de généraliser les résultats. En effet, nous aurions été rapidement confrontés à un phénomène de sur-apprentissage. De plus, nous ne souhaitions pas nous limiter à la détection des attaques connues. Notre objectif était de permettre la détection des comportements anormaux d'objets connectés.

2.3.2 Détection des anomalies par apprentissage

Dans cette section, nous présentons les quatre grandes familles qui composent la détection des anomalies par apprentissage, à savoir l'apprentissage supervisé, semi-supervisé, non supervisé et par renforcement comme décrit dans la figure 2.3. La détection des anomalies par apprentissage repose sur l'apprentissage automatique. Elle cherche, à partir des données, à construire des modèles mathématiques permettant de faire la distinction entre les données normales et anormales. Nous discutons, dans un premier temps, des limites des méthodes de détection par renforcement et par apprentissage supervisé. Puis, nous détaillons un ensemble de méthodes qui ont été considérées pour nos travaux et qui relèvent de la détection par apprentissage non supervisé et semi-supervisé.

A. Détection des anomalies par renforcement

La détection d'anomalies par renforcement constitue un domaine de recherche en pleine expansion. Les méthodes par renforcement se rapprochent du mécanisme d'apprentissage humain avec un système de récompenses. Ce dernier doit permettre d'orienter les prises de décisions de l'agent qui classifie les données. La définition du système de récompenses peut, selon les circonstances, se révéler complexe. C'est pourquoi, la mise en place de ces méthodes pour faire de la détection d'anomalies est souvent plus difficile que dans le cas des méthodes d'apprentissage classiques. Dans le cadre du projet SecureIoT, nous avions pour objectif de détecter des anomalies dans des systèmes où la définition soit d'une fonction de coût, à minimiser, ou de gain, à maximiser, n'était pas évidente. L'un des principaux intérêts de l'utilisation de l'apprentissage par renforcement est de laisser la possibilité à l'agent de s'adapter et donc d'être utilisé dans des contextes proches. Il est ainsi possible d'entraîner un robot à manipuler des objets d'une taille donnée, puis d'utiliser l'apprentissage par renforcement pour permettre à la main du robot de manipuler des objets de tailles variables [113]. Malheureusement, les scénarios d'application de la plupart des travaux de recherche, qui utilisent l'apprentissage par renforcement pour faire de la détection des anomalies, sont souvent simples [124]. De plus, ils apportent rarement de plus-value par rapport à des méthodes de détection par apprentissage en détectant, par exemple, les données contenant un bruit gaussien [165]. La détection de telles anomalies se fait facilement avec des méthodes de détection moins complexes à mettre en place, comme avec des algorithmes de partitionnement [104, 105, 145].

Pour ces différentes raisons, nous avons fait le choix d'étudier plus en détail les méthodes de détection par apprentissage ne faisant pas intervenir de notion de renforcement.

B. Détection des anomalies par apprentissage supervisé

De nombreux travaux de recherche ont été réalisés pour permettre de détecter des comportements anormaux dans les réseaux informatiques. Parmi eux, nous trouvons les méthodes de détection des anomalies par apprentissage supervisé. Pour appliquer ces dernières, il est nécessaire que les données soient labélisées. Dans [58], deux modules de détection ont été associés, un pour les anomalies et l'autre pour les cas de mauvais usage ou d'utilisation abusive. La décision finale pour la détection est prise par un arbre de décision en prenant en compte les retours des deux modules cités précédemment. Les arbres de décision sont l'une des méthodes les plus populaires et intuitives utilisées pour la classification. Cette méthode partitionne les données de façon récursive en considérant les valeurs des différents attributs jusqu'à atteindre une condition d'arrêt. La figure 2.4 présente un exemple d'arbre de décision (DT) pouvant être obtenu pour des données contenant trois attributs (a,b et c). L'un des problèmes majeurs de cette méthode de classification est la grande probabilité de sur-apprentissage. En effet, il est courant que le modèle ainsi généré soit trop proche des données d'apprentissage. Ainsi, la moindre variation du jeu de données d'apprentissage peut fortement influencer le modèle ce qui va empêcher sa généralisation à un autre jeu de données.

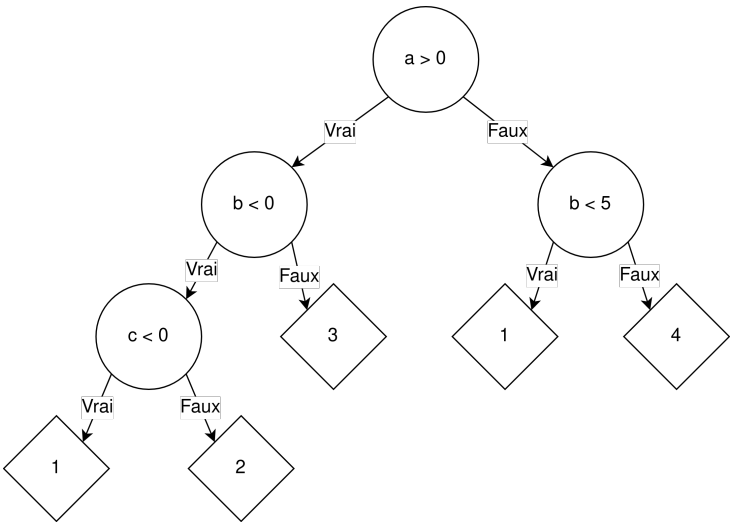


FIGURE 2.4 – Exemple d'un arbre de décision

Afin d'éviter ce phénomène, la méthode de forêt aléatoire [41] (RF) a été proposée. Elle consiste en une utilisation d'un grand nombre d'arbres de décision plutôt que d'un seul. Ces arbres de décision sont construits à l'aide de différents sous-ensembles de données et en considérant des sous-ensembles différents d'attributs. La figure 2.5 illustre, en se rapprochant de l'exemple précédent, la forme que pourrait avoir la forêt aléatoire toujours dans le cas d'un jeu de données contenant trois attributs. La décision finale de classification est prise soit en prenant en compte la majorité des votes de chacun des arbres de décision soit en calculant un score d'anormalité pour chacun des points de données.

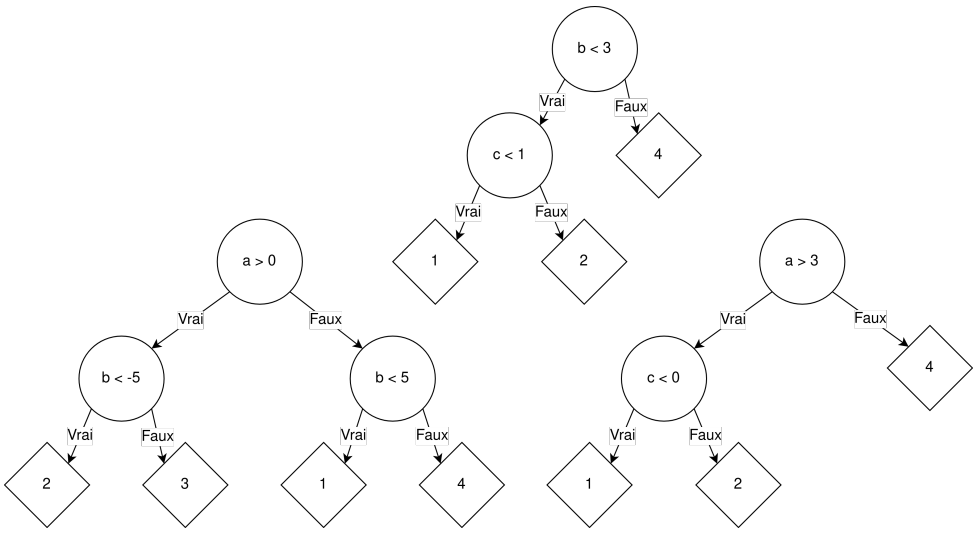


FIGURE 2.5 – Exemple d'une forêt aléatoire

Les méthodes utilisant les arbres de décision, que ce soit la méthode originelle ou la forêt aléatoire, sont des méthodes d'apprentissage supervisées. En effet, les arbres de décision tentent de faire correspondre les nouvelles données à tester avec celles utilisées dans l'apprentissage. De plus, il est nécessaire de connaître à quel groupe appartient chaque point de données lors de l'apprentissage. Cette condition ne permet pas de facilement traiter des données qui n'appartiendraient à aucun des groupes trouvés dans la phase d'apprentissage. C'est pourquoi, nous avons fait le choix de considérer exclusivement des méthodes semi et non supervisées dans nos travaux.

C. Détection des anomalies par apprentissage non supervisé

Le principal avantage de telles méthodes est de pouvoir directement les appliquer sur les données sans avoir besoin de les labéliser. En effet, ces méthodes vont séparer les données en plusieurs groupes. Ceux contenant la majorité des données déterminent les données normales tandis que les autres groupes permettent de détecter les anomalies. Dans cette section, nous nous inspirons de la classification des méthodes de détection proposées dans [15] et utilisons deux des grandes familles de méthodes données par les auteurs. Ces deux familles regroupent les méthodes reposant sur la proximité et celles pour les grandes dimensions. Nous montrons comment ont été utilisées les méthodes de détection appartenant à ces différentes catégories dans la détection d'anomalies.

Méthodes reposant sur la proximité. Ces dernières ont l'avantage d'être simples à paramétrer et ne nécessitent aucune hypothèse sur le type de distribution des données. En revanche, elles demandent un temps de traitement important car elles doivent calculer les distances entre les différents points du jeu de données.

Dans cette famille, nous trouvons un grand nombre de méthodes utilisant *K-Nearest Neighbour* [66] comme base. Le plus souvent, elles utilisent la distance euclidienne ou celle de Mahalanobis pour déterminer quels sont les éléments les plus proches les uns des autres. À cause de cela, ces méthodes ont des difficultés à traiter à la fois des données de grandes dimensions et de grands jeux de données. C'est pourquoi, un certain nombre de travaux cherchent à optimiser cette méthode, par exemple en attribuant les données à des cellules et en vérifiant que les voisinages de ces cellules contiennent suffisamment de points [137]. En utilisant cette méthode, il est

possible de distinguer les zones de données denses, et donc normales, des zones de données peu denses, où se trouvent les données anormales.

Il existe d'autres travaux adaptant la méthode des K plus proches voisins (K-NN) afin de réduire la dimension de l'espace des données. Différentes solutions proposent d'extraire les attributs les plus pertinents et donc d'ignorer les autres [132], [92] et [15]. Une autre amélioration de K-NN utilise une structure d'indexation efficace, ce qui lui permet d'obtenir des temps de traitement linéaires [63].

Le principe de distinguer les zones de données denses et peu denses fait penser à la méthode de partitionnement DBSCAN [62] qui a, elle aussi, été utilisée pour faire de la détection d'anomalies. Cette méthode est certainement la méthode de partitionnement la plus utilisée pour cette famille. Son algorithme sélectionne aléatoirement les points du jeu de données lors de la phase d'apprentissage, puis il regarde combien de voisins sont présents. Dans le cas où le nombre de voisins est plus grand qu'un paramètre prédéfini, le point de données va former un groupe. Lorsqu'un nouveau groupe est créé, l'algorithme va l'agrandir autant que possible en y intégrant ses voisins ainsi que l'ensemble des points atteignables. Les principaux avantages de cette méthode de partitionnement sont sa résistance au bruit ainsi que sa capacité à trouver des groupes de formes non sphériques. En revanche, elle rencontre des difficultés dans le traitement de données lorsque la densité des données n'est pas homogène. De plus, trouver les meilleurs paramètres à utiliser peut présenter un défi.

Concernant son application à de la détection des anomalies, des travaux de recherche proposent de considérer comme anormal l'ensemble des groupes trouvés par cet algorithme qui ne contiendraient pas assez de points [46]. Par exemple, si l'on considère que chaque groupe doit au moins contenir 10% des données alors il faudra considérer les groupes 2 et 5 de la figure 2.6 comme étant composé de points anormaux.

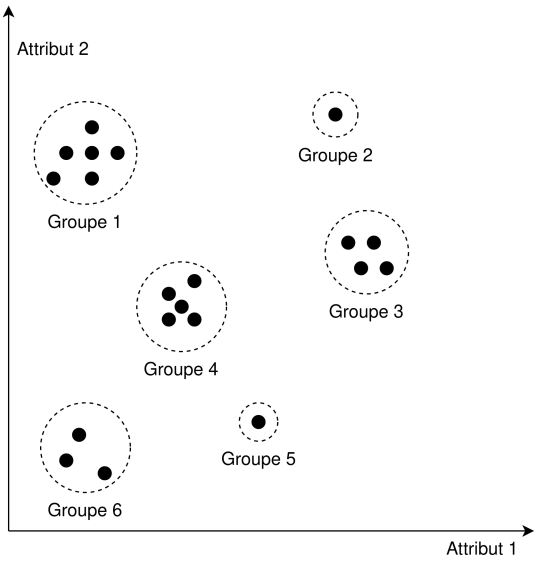


Figure 2.6 – Détection d'anomalies en utilisant DBSCAN

Les méthodes reposant sur la proximité peuvent également contenir des méthodes de partitionnement comme K-MEANS [109] pour détecter des anomalies [23] et [126]. Cet algorithme répartit les points de données en différents groupes partageant des similarités lors de la phase d'apprentissage. Chacun de ces groupes va occuper un certain espace du plan selon la dimension des

points de données. Ainsi, l'algorithme va chercher les paramètres pour définir des hypersphères pour chacun des groupes de données qui englobent l'ensemble des points qui le composent. Lors de la détection, les données en dehors des hypersphères seront considérées comme anormales. Il faut toutefois noter l'inconvénient de l'algorithme K-MEANS qui demande de connaître le nombre de groupes dans le jeu de données. Dans la plupart des cas cette méthode devra être appliquée plusieurs fois pour trouver le nombre de groupes optimal.

L'une des méthodes de proximité de référence dans les travaux de comparaison des performances de détection est le Local Outlier Factor [42] (LOF). Cette méthode étudie la densité aux alentours des différents points de données en prenant en compte la distance entre les points et les densités locales. De cette façon, les points se trouvant dans les zones peu denses seront considérés comme anormaux.

Méthodes pour les grandes dimensions. Ce sont les méthodes qui ont été conçues pour pouvoir traiter les données de grandes dimensions. En effet, dans ces cas précis, il n'est pas rare de se voir confronté aux problèmes induits par les grandes dimensions (*curse of dimensionality*) rendant complexe la recherche d'espace dense de points de données.

Plusieurs solutions ont été proposées dans la littérature pour réduire l'influence de ce phénomène. La méthode SPOT [166] essaie de trouver un ensemble de sous-espaces contenant les points anormaux. Elle cherche à la fois à proposer une solution de détection utilisable en grande dimension mais également de minimiser sa complexité temporelle afin de pouvoir l'utiliser en temps réel. Lors de la phase de détection, l'algorithme va tester si le point de données est suffisamment proche d'un sous-espace anormal pour être labélisé comme tel.

D. Détection des anomalies par apprentissage semi-supervisé

En plus des méthodes non supervisées, nous avons également choisi de prendre en compte des méthodes semi-supervisées dans le but d'identifier les anomalies. La principale distinction entre ces deux catégories intervient pendant l'apprentissage. Dans le cas des méthodes semi-supervisées, l'hypothèse retenue est que l'intégralité du jeu de données ne contient pas d'anomalies. De cette façon, la méthode de détection va construire des modèles pour caractériser les données normales. Ainsi, il suffira, dans la phase de détection, d'évaluer pour chaque point de données à traiter s'il fait partie des données normales.

Méthodes probabilistes et statistiques. Ce sont les premières méthodes de détection d'anomalies que l'on trouve dans la littérature bien que la plupart d'entre elles ne puissent être appliquées qu'à une variable unique. Par exemple, la méthode de Grubb [71], qui ne fonctionne qu'en dimension 1, va standardiser, avec la méthode Z-score, les valeurs prises par l'attribut avant de vérifier que chacune de ces valeurs ne s'éloigne pas trop de la moyenne. Concrètement, après la standardisation des données, celles qui ont une valeur absolue supérieure à un seuil, proposé entre 1 et 5% par les auteurs, seront considérées comme étant anormales.

Nous trouvons également une autre méthode plus visuelle, le box plot outlier [15]. Son principe est de générer la boîte à moustaches composée des cinq éléments suivants : la valeur minimale, le quartile minimal, la médiane, le quartile maximal et la valeur maximale. L'ensemble des données dont la valeur est en dehors d'un intervalle équivalent à quatre fois la différence entre le quartile minimal et le quartile maximal sont considérées comme anormales. Cette méthode a été testée sur des données médicales et l'ensemble des anomalies détectées n'a pas nécessairement été validé par les experts [98]. En effet, l'attribut utilisé par les auteurs présente des variations impliquant un recoupement des données normales et anormales dans l'intervalle.

Certaines méthodes de détection probabilistes et statistiques vont construire des modèles en supposant que le phénomène que l'on étudie peut être résumé comme étant une distribution probabiliste de plusieurs attributs. La méthode *elliptic envelope* (EE) [83] fait l'hypothèse que les données suivent une distribution de Gauss et cherche donc à déterminer les paramètres qui vont permettre de décrire au mieux les points de données de la phase d'apprentissage. À la fin de l'apprentissage, comme le montre la figure 2.7, l'algorithme va définir une ellipse dans laquelle s'inscrivent tous les points considérés comme normaux. Ainsi, lors de la phase de détection, les points se trouvant en dehors de l'ellipse seront traités comme étant anormaux, représentés en rouge sur la figure, tandis que les points normaux, colorés en bleu, seront situés dans l'ellipse.

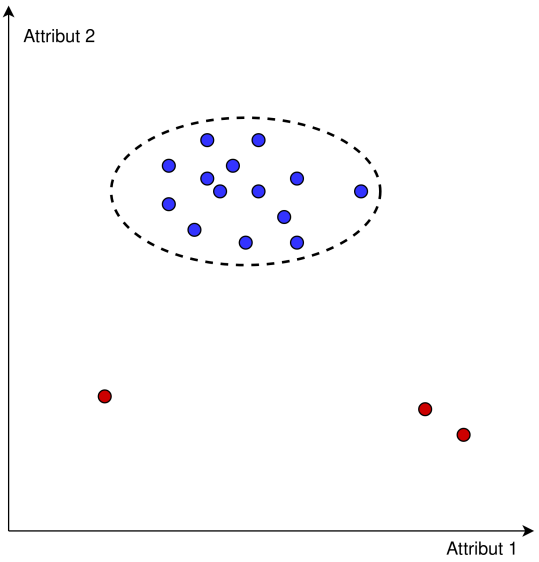


FIGURE 2.7 – Détection des anomalies par l'elliptic envelope (EE)

Les méthodes de détection que nous avons décrites jusqu'à présent permettent de visualiser les points anormaux dans les jeux de données. En revanche, dans les cas où l'anomalie proviendrait d'un enchaînement d'événements dans une séquence temporelle, alors nous pourrions nous trouver dans la configuration illustrée par la figure 2.8 ou il n'est pas possible de repérer l'anomalie en considérant les valeurs des attributs des points de données de manière indépendante. En effet, la valeur de l'attribut au moment de l'attaque, à t_a, est incluse dans l'intervalle des valeurs normales. C'est pourquoi nous avons été amenés à considérer des méthodes qui prennent en compte l'évolution temporelle des données.

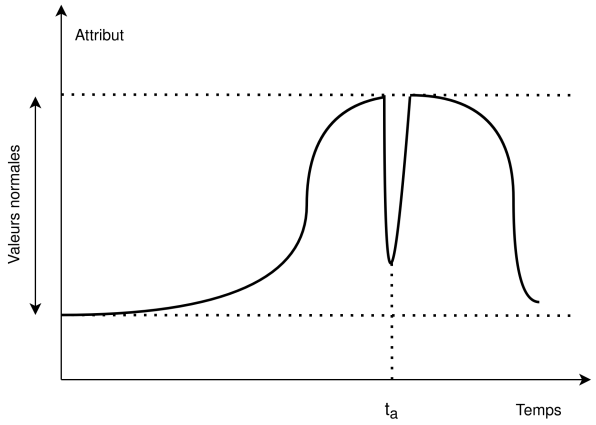


FIGURE 2.8 – Exemple d'anomalie dans une séquence temporelle

Différentes techniques ont été développées pour permettre de détecter des séquences anormales. Par exemple, nous trouvons dans la littérature des méthodes qui vont décrire l'évolution des données sous la forme de graphes. C'est notamment le cas des modèles des chaînes de Markov cachées (Hidden Markov Model [136]). Les modèles de Markov sont des modèles statistiques qui visent à modéliser les possibilités d'évolution d'un système. Dans le cas classique d'usage les états pris par le système sont connus de l'utilisateur. Si cela n'est pas possible alors le recours aux HMM est intéressant car ils vont interpréter les valeurs de différents attributs pour deviner les états du système. Une utilisation de HMM appliqué à de la détection a été réalisée dans [159]. Dans un premier temps les auteurs ont construit les modèles lors de la phase d'apprentissage avant de faire un test de vraisemblance entre le modèle et les données de la phase de détection. Ils ont également testé une autre méthode de détection où ils ont pris en compte les transitions apparaissant dans les données mais ne pouvant être répercutées sur le modèle. L'utilisation de graphe en tant que modèle de comportement permet de prendre en compte l'évolution passée du système et donc de détecter à la fois des points de données anormaux mais également de repérer les séquences d'événements qui s'éloignent du modèle.

Méthodes linéaires. Ces méthodes, dites linéaires, cherchent à intégrer le maximum de données dans un ou des sous-espaces de dimensions inférieures au nombre d'attributs des données. Par exemple, comme le montre la figure 2.9, les données bien que de dimension 3 pourraient être représentées en dimension 2.

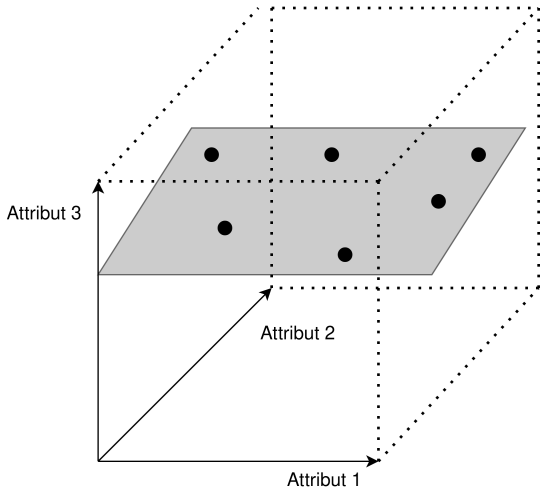


FIGURE 2.9 – Réduction de l'espace de représentation des données

La phase d'apprentissage de ce type de méthodes consiste à trouver ces sous-espaces. Lors de la phase de détection, les points qui ne font pas partie de ces sous-ensembles seront considérés comme des anomalies. La méthode de détection de référence pour cette famille est le *One Class SVM* (OCSVM) [143] qui est l'adaptation semi-supervisée de la méthode *SVM* [40]. En effet, nous allons exclusivement utiliser des données normales lors de l'apprentissage pour trouver les sous-espaces des données normales et pouvoir repérer les anomalies. La figure 2.10 montre de quelle façon cette distinction entre les données normales et anormales va être faite avec les points en bleu qui appartiennent au sous-espace, et sont donc normaux, tandis que les points en rouge se situent en dehors du sous-espace, et sont donc des anomalies.

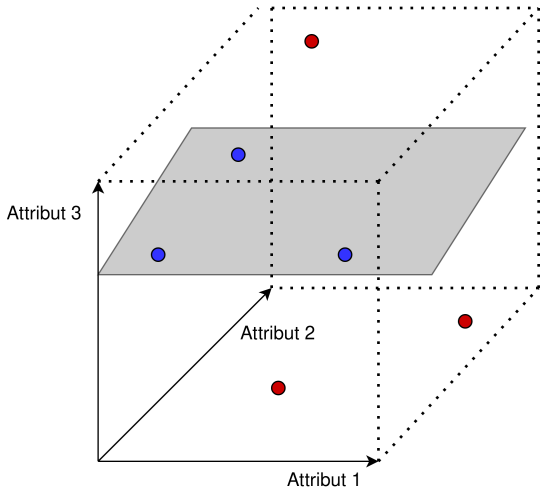


FIGURE 2.10 – Détection des anomalies pour le OCSVM

Les méthodes de détection utilisant l'analyse en composantes principales (Principal component analysis [81]), ou PCA, reposent sur le même principe que OCSVM. Les points de données s'éloignant d'un sous-espace, défini dans la phase d'apprentissage, sont considérés comme des anomalies. Ces méthodes ont été appliquées dans des domaines variés, comme par exemple en statistique [45] ou en astronomie [59].

Dans cette catégorie des méthodes linéaires, nous trouvons également certains types de réseaux de neurones comme les autoencoder et les Replicator Neural Network [56] (Rep NN). Dans les deux cas, la détection des anomalies se fait au moment de la reconstruction dans le décodeur que l'on peut voir dans la figure 2.11. En effet, le principe de ces deux algorithmes est d'arriver à reconstruire les données d'entrée à la sortie du réseau de neurones. Ainsi, lors de la phase de détection, les données normales pourront bien être reconstruites à la sortie du modèle contrairement aux données anormales. La comparaison de la sortie du réseau de neurones avec le point de données initial va ainsi permettre de déterminer si le point de données est normal ou non.

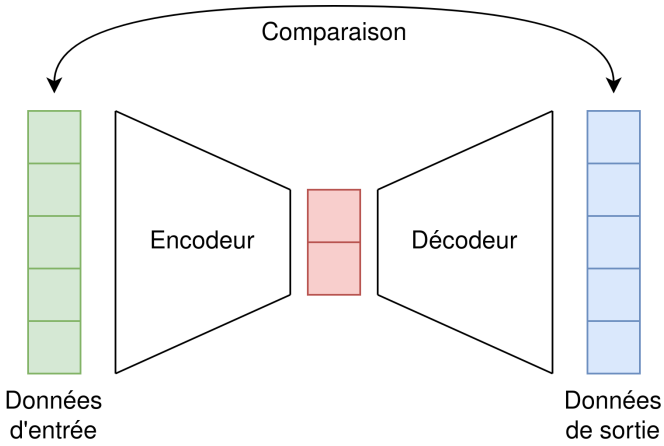


FIGURE 2.11 – Détection des anomalies pour les réseaux de neurones

Nous trouvons également des techniques propres aux réseaux de neurones qui peuvent détecter des séquences anormales. Par exemple, la famille des Reccurent Neural Network (Rec NN) a la possibilité de garder en mémoire les données précédentes. Dans cette famille nous trouvons la méthode Long Short Term Memory [47] (LSTM) qui est composée de cellules pilotées chacune par trois portes, comme illustré par la figure 2.12.

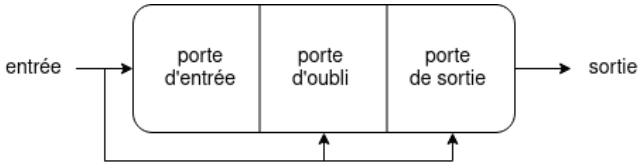


FIGURE 2.12 – Composition d'une cellule de la méthode LSTM [47]

La première est la porte d'entrée qui va décider si la cellule doit être modifiée. La seconde est la porte d'oubli qui va définir dans quelle mesure la cellule doit prendre en compte les données précédentes. Finalement, nous allons également trouver la porte de sortie qui décide des éléments qui vont être transmis à la prochaine cellule. En ce qui concerne l'application à la détection d'anomalies, les auteurs de [112] ont entraîné un modèle sur des données normales. Puis, dans la phase de détection, ils ont calculé l'erreur entre la valeur attendue et la valeur fournie par le point de données pour repérer les anomalies.

Méthodes basées sur la proximité. Nous allons trouver dans cette famille une adaptation de l'algorithme de forêt aléatoire que nous avons évoqué dans la section précédente. Cette adaptation

va permettre de transformer cette méthode supervisée en une méthode semi-supervisée. Ainsi, dans [69] les auteurs ont proposé l'algorithme de la forêt aléatoire à une classe (OCRF). Durant la phase de détection, un ensemble d'arbres de décision va être construit sur le même modèle que pour la forêt aléatoire sauf qu'ici chacune des feuilles des arbres correspondra à un même groupe, le groupe des données normales. De cette façon, il sera possible de détecter les anomalies lors de la phase de détection. Pour cela, il faudra faire parcourir les différents arbres à nos points de données à tester, et ceux ne pouvant pas aller assez loin seront considérés comme des points anormaux.

Nous trouvons aussi d'autres algorithmes comme BIRCH [167] qui est une méthode de partitionnement hiérarchique qui va, dans un premier temps, construire un arbre où chaque feuille correspondra à un point de données. La partie (a) de la figure 2.13 montre la répartition des points de données tandis que la partie (b) fait apparaître les liens entre les différentes feuilles en fonction de la distance séparant ces points. Au-delà d'une distance maximale, les sous-ensembles de points appartiendront à des groupes différents. Dans notre exemple, nous observons la présence de deux groupes, le premier constitué des points a et b et le second des points c, d et e.

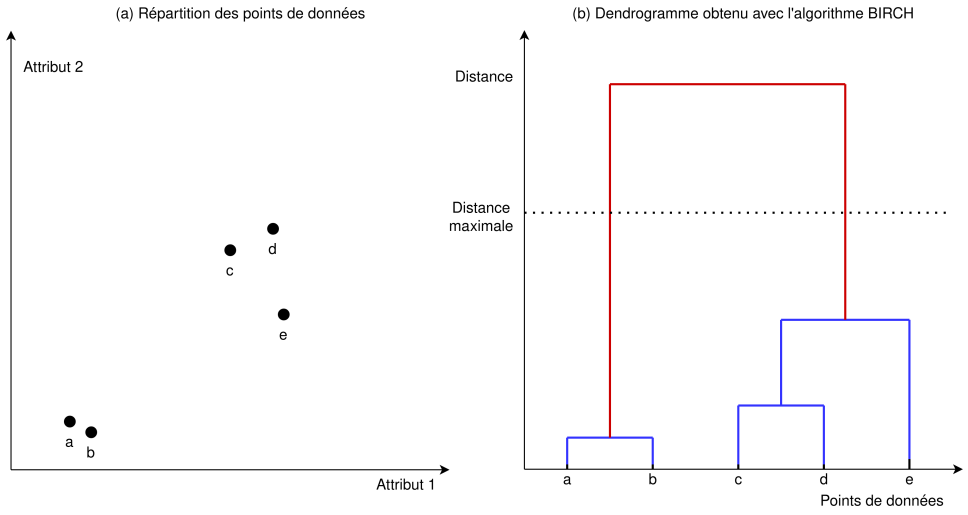


FIGURE 2.13 – Illustration du fonctionnement du partitionnement hiérarchique

Lors d'une dernière étape, il est possible d'appliquer un autre algorithme de partitionnement, le plus souvent K-MEANS, directement sur les groupes trouvés précédemment.

L'application de cette méthode à la détection d'anomalies a été réalisée dans [43]. Les auteurs ont proposé de construire un modèle en utilisant les données normales dans la phase d'apprentissage. Nous sommes donc bien dans un cas d'apprentissage semi-supervisé. Puis, lors de la détection, leur méthode va chercher le groupe le plus proche du point à tester avant de calculer la distance qui sépare ces deux éléments. Si cette distance est plus importante qu'une valeur de seuil alors le point n'appartient pas au groupe. Il n'est donc représenté par aucun des groupes trouvés lors de la phase d'apprentissage sur les données normales. Dans ce cas, la méthode considère que ce point représente une anomalie.

Méthodes pour les grandes dimensions. Plusieurs solutions ont été proposées dans la littérature pour essayer de limiter la dégradation des performances de la détection des anomalies dans les espaces de grandes dimensions. L'algorithme OUTRES [125] utilise une exploration

récursive des sous-espaces d'un point de données. En effet, il est possible, en ne considérant qu'un sous-espace, qu'un point de données anormal passe inaperçu. Ainsi, lors de la prise de décision finale, l'algorithme prend en compte l'ensemble des scores de détection de chacun des sous-espaces pour distinguer un point normal d'un anormal.

La méthode de la forêt d'isolement (Isolation Forest) [107] est la plus couramment utilisée, pour cette famille, dans les travaux qui comparent les performances de plusieurs méthodes de détection. Durant la phase d'apprentissage, elle réduit la dimension du jeu de données en sélectionnant aléatoirement plusieurs sous-ensembles d'attributs. Puis, un arbre binaire est construit pour chacun de ces sous-ensembles. Pour la détection, une vérification est effectuée afin de s'assurer que les données peuvent bien se propager dans les arbres binaires. Pour un point de données, si la progression n'est pas possible dans un nombre suffisant d'arbres alors cela signifie que ce point est anormal. Avec ses bonnes performances de détection et sa faible complexité temporelle, cette méthode est couramment utilisée dans l'industrie.

2.3.3 Détection des anomalies par utilisation du *process mining*

Le *process mining* regroupe un ensemble de méthodes à la jonction entre l'analyse des données et l'analyse métier (Business Analytics). En effet, le principe fondamental est de permettre de modéliser le plus fidèlement possible les systèmes à états dont les événements peuvent être observés sur un fichier de log. La figure 2.14 illustre le mécanisme de transformation des fichiers de logs en modèles en utilisant les méthodes de *process mining*. Ce type de méthodes met l'accent sur la construction de modèles facilement compréhensibles car leur utilisation doit faciliter les échanges entre des équipes de différents domaines (technique, commerciale, etc) sur un système. Le début de l'engouement pour ce sujet remonte à 1998 où Wil van der Aalst propose une utilisation des réseaux de Petri pour modéliser les flux opérationnels [150]. Ces flux représentent une suite de tâches ou d'opérations effectuées par une entité.

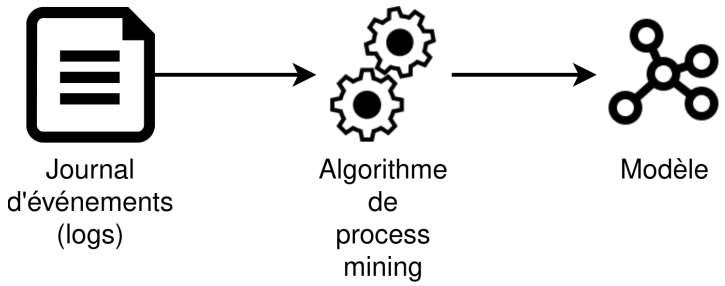


FIGURE 2.14 – Illustration du fonctionnement du *process mining*

L'application du *process mining* à de la détection des anomalies est plus récente, avec les premiers travaux publiés un peu avant les années 2010. Ainsi, dans [35] les auteurs introduisent une métrique servant à évaluer la ressemblance entre les modèles générés dans une phase initiale avec les données à évaluer. Cette métrique permet de détecter les séquences d'événements anormaux, qui pourraient être la traduction d'un mauvais fonctionnement causé par une attaque.

Depuis, les méthodes de *process mining* ont pu montrer leurs atouts dans différents domaines d'application [152], allant de la santé à la production en passant par la maintenance. Ces méthodes, associées aux méthodes de fouille de données (Data mininig), ont initialement été conçues pour modéliser le comportement de systèmes à événements [14]. Nous avons pu voir des utilisations hybrides du *process mining* le combinant notamment avec des méthodes d'apprentissage

machine (machine learning). Dans [61] les auteurs construisent un modèle d'un système industriel utilisant une méthode de *process mining* avant d'optimiser l'allocation de ses ressources à l'aide de méthodes d'apprentissage automatique.

L'un des problèmes rencontré dans le déploiement des solutions de détection qui utilisent le *process mining* est la contrainte sur le format des données d'entrée. En effet, pour créer les modèles, l'algorithme va devoir traiter un journal d'événements. Pour traiter d'autres formats de données, nous allons devoir faire intervenir des méthodes en amont afin d'obtenir ces données sous la forme d'un journal d'événements. C'est ce que nous avons fait dans nos travaux [75–77]. Une utilisation similaire sur des données médicales a été conceptualisée dans [147]. Dans cet article, les auteurs considèrent que les anomalies détectées correspondent à des maladies.

2.4 Application de deux algorithmes de *process mining*

Dans nos travaux, nous avons initialement fait le choix d'utiliser les méthodes de *process mining* qui permettent de créer des modèles simples et facilement compréhensibles du comportement des systèmes. Dans cette section, nous illustrons le fonctionnement de deux algorithmes couramment utilisés et reposant sur des fonctionnements différents. Le premier d'entre eux, l'*inductive miner* (IM), fait intervenir le graphe des suivants directs (*directly-follows graph*) duquel il déduit le modèle de comportement. Le second algorithme est le *transition system miner* (TSM) qui repose sur la notion de région pour construire son modèle. Nous faisons apparaître cette distinction dans les deux sous-sections dédiées à ces algorithmes.

2.4.1 Algorithme de l'inductive miner

L'utilisation de l'*inductive miner* est la première approche que nous avons considérée. En effet, son principal avantage est sa capacité à créer des modèles, sous la forme de graphes, ne comportant pas d'impasse ou de cycle bloquant. Cependant, avant de décrire ses différentes étapes, il est nécessaire d'introduire un certain nombre de notions telles que le graphe des suivants directs (*directly-follows graph*), l'activité silencieuse et l'arbre des processus (*process tree*). Nous illustrerons, dans cette sous-section, le fonctionnement de cet algorithme en l'utilisant sur un journal d'événements J. Ce journal d'événements contient l'agrégation de 350 jeux de données. Nous allons trouver, par exemple, cinquante fois le même jeu de données $< a, b, c, a, b, e, f >$, ce qui signifie que nous allons voir 50 fois cette suite d'événements dans nos jeux de données. On remarque dans cet exemple que ce système peut évoluer différemment. En effet, dans 100 des jeux de données nous allons rencontrer la suite d'événements $< a, b, f, e >$. Dans les 200 jeux de données restant, dont la première moitié est de la forme $< d, e, f >$ et l'autre décrit la séquence $< d, f, e >$, nous observons l'apparition de l'événement d et la disparition de a.

$$J = [< a, b, c, a, b, e, f >^{50}, < a, b, f, e >^{100}, < d, e, f >^{100}, < d, f, e >^{100}] \tag{2.1}$$

Le graphe des suivants directs (*directly-follows graph*) est construit à partir des données en sortie du bloc de pré-traitement. Il correspond à un graphe, où chaque état est directement relié à un des états mis en avant par le pré-traitement des données. De plus, celui-ci ne contiendra que des transitions visibles dans les données. Ainsi, si l'on peut voir une transition entre deux états du graphe cela signifie que cette évolution est directement visible dans les données qui ont servi à le générer. En partant du graphe des suivants directs (*directly-follows graph*), on va déduire un arbre de processus (*process tree*). Cela permet de représenter de façon compacte un réseau de Petri où chaque feuille est un événement, ou un état dans notre cas, et les nœuds

correspondent aux opérateurs qui décrivent les interactions entre les différents états. Il existe quatre types d'opérateurs :

— le choix exclusif $\times$

— la composition séquentielle $\rightarrow$

— la boucle $\circlearrowleft$

— la composition parallèle $\hat{}$

Lors de la transformation de l'arbre de processus (*process tree*), il est à noter qu'un type particulier de transition peut apparaître dans le réseau de Petri final. Il s'agit d'une activité silencieuse (*silent activity*) [2], c'est-à-dire un événement non visible dans les données mais qui est nécessaire à la construction d'un réseau de Petri où chaque état est accessible. Ainsi, ces activités silencieuses permettent de relier complètement tous les états du réseau de Petri, cela inclut également l'état initial et final qui sont ajoutés aux autres états visibles. Les différentes étapes de la construction du réseau de Petri par l'inductive miner sont décrites ci-dessous en se basant sur le journal d'événements décrit par l'équation 2.1.

Construction du graphe des suivants directs (*directly-follows graph*). La figure 2.15 correspond au graphe des suivants directs (*directly-follows graph*), construit à partir du journal d'événements où le poids des arêtes est la traduction du nombre de fois où la transition a pu être vue dans les données d'entrée. Dans notre graphe, on peut voir en vert les états d'entrée et en rouge les états de sorties.

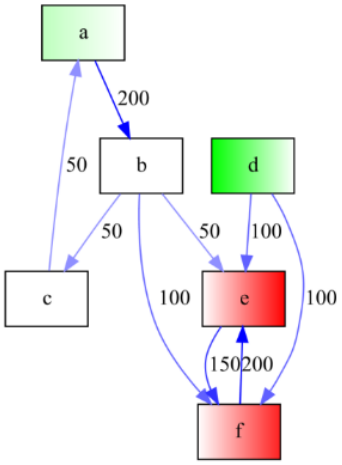


FIGURE 2.15 – Graphe des suivants directs obtenu à partir des données

Déduction d'un arbre de processus (*process tree*). L'algorithme cherche quel est l'opérateur le plus adéquat (choix exclusif, séquentiel, boucle, parallélisation) pour découper le graphe des suivants directs (*directly-follows graph*). Les différents opérateurs permettant cette découpe sont donnés par la figure 2.16. L'ensemble des événements possibles est d'abord coupé en deux sous-ensembles, que l'on notera set_1 et set_2. Nous aurons donc deux parties du journal d'événements, que l'on notera J_1 et J_2, qui ne pourront contenir respectivement que des événements des sous-ensembles set_1 et set_2. Récursivement, une nouvelle découpe est effectuée jusqu'à ce que chaque sous-ensemble set_i ne contiennent plus qu'un seul événement. Lors de chacune de

ces étapes, le graphe des suivants directs (*directly-follows graph*) est découpé et l'opérateur sélectionné est celui qui permet d'agréger le nombre le plus important de noeuds/événements.

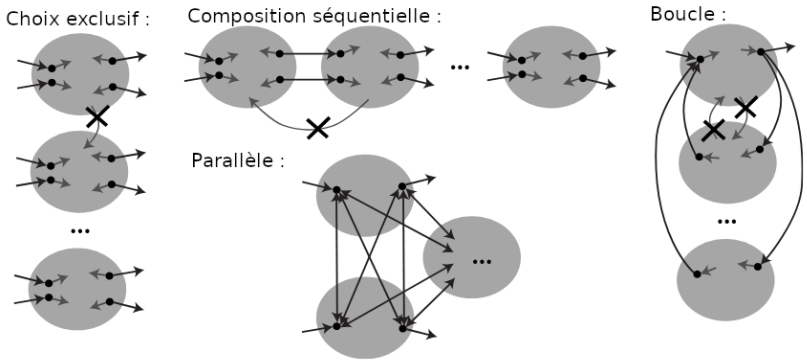


FIGURE 2.16 – Opérateurs servant à la découpe du graphe des suivants directs

Dans notre exemple, la première découpe, illustrée par la figure 2.17, correspond à un opérateur séquentiel. Ainsi, les différents événements du sous-ensemble initial $set : \{a, b, c, d, e, f\}$ sont séparés pour donner $set_1 : \{a, b, c, d\}$ et $set_2 : \{e, f\}$. De même, le journal d'événements initial est lui aussi découpé, pour donner deux nouveaux journaux d'événements J_1 et J_2 dont le contenu est décrit par les équations 2.2 et 2.3.

$$J_1 = [< a, b, c, a, b >^{50}, < a, b >^{100}, < d >^{200}] \tag{2.2}$$

$$J_2 = [< e, f >^{150}, < f, e >^{200}] \tag{2.3}$$

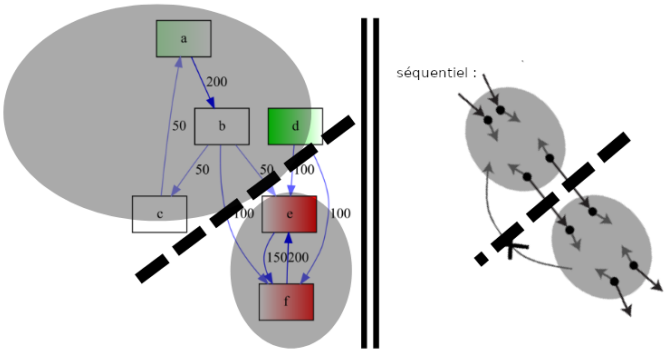


FIGURE 2.17 – Première étape de découpe du graphe des suivants directs avec l'opérateur séquentiel

Nous allons maintenant chercher l'opérateur qui permet au mieux de séparer les éléments de J_1. Il s'avère qu'ici, l'algorithme choisit l'opérateur de choix exclusif pour la séparation des données, comme montré par la figure 2.18. Ainsi, les événements du $set_1 : \{a, b, c, d\}$ sont une nouvelle fois séparés en deux sous-ensembles $set_3 : \{a, b, c\}$ et $set_4 : \{d\}$. Les descriptions des journaux d'événements correspondant à ces deux sous-ensembles J_3 et J_4 sont données par les équations 2.4 et 2.5. On remarque que set_4 ne contient qu'un seul événement possible, nous avons

donc trouvé la première feuille de notre arbre de processus (*process tree*) et il n'est plus nécessaire de sous-diviser J_4.

$$J_3 = [< a, b, c, a, b >^{50}, < a, b >^{100}] \tag{2.4}$$
$$J_4 = [< d >^{200}] \tag{2.5}$$

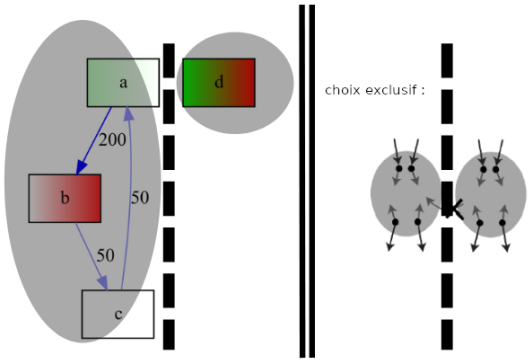


FIGURE 2.18 – Seconde étape de découpe du graphe des suivants directs avec l'opérateur du choix exclusif

En revanche set_3 contient encore plusieurs éléments, donc dans ce cas nous allons continuer à sous-diviser le journal d'événements J_3. Dans le cas de J_3, l'algorithme propose d'utiliser l'opérateur boucle, comme décrit par la figure 2.19. Ainsi, set_3 est découpé en deux sous-ensembles $set_5 : \{a, b\}$ et $set_6 : \{c\}$ qui se traduit, là encore, par une séparation de J_3 en deux sous-ensembles dont les contenus sont donnés par les équations 2.6 et 2.7.

$$J_5 = [< a, b >^{200}] \tag{2.6}$$
$$J_6 = [< c >^{50}] \tag{2.7}$$

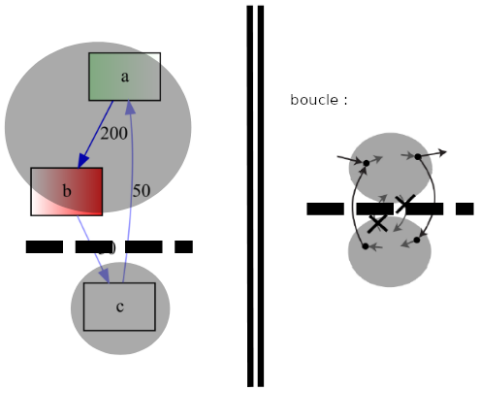


FIGURE 2.19 – Troisième étape de découpe du graphe des suivants directs avec l'opérateur boucle

Nous allons à présent considérer le journal d'événements J_5, qui peut être séparé en utilisant l'opérateur séquentiel, comme cela est montré dans la figure 2.20. Ainsi, set_5 est découpé en

$set_7 : \{a\}$ et $set_8\{b\}$, et donc par conséquent, J_5 est découpé en L_7 et L_8 dont les descriptions sont données par les équations 2.8 et 2.9.

$$J_7 = [< a >^{200}] \tag{2.8}$$
$$J_8 = [< b >^{200}] \tag{2.9}$$

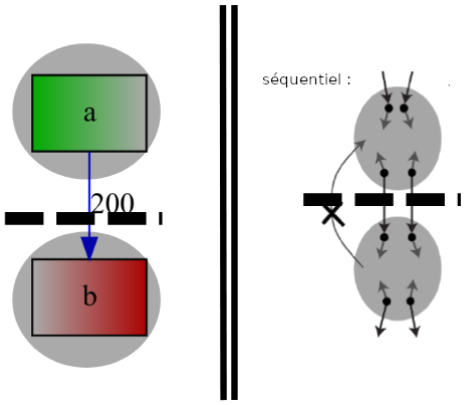


FIGURE 2.20 – Quatrième étape de découpe du graphe des suivants directs avec l'opérateur séquentiel

Maintenant que nous sommes allés au bout du processus avec J_1, revenons au sous-ensemble J_2. Ce dernier peut être découpé par un opérateur parallèle, comme montré par la figure 2.21. Ainsi le sous-ensemble d'événements $set_2 : \{e, f\}$ peut être séparé en deux sous-ensembles $set_9 : \{e\}$ et $set_{10} : \{f\}$ dont les journaux d'événements correspondant J_9 et J_{10} sont décrits par les équations 2.10 et 2.11.

$$J_9 = [< e >^{350}] \tag{2.10}$$
$$J_{10} = [< f >^{350}] \tag{2.11}$$

Ainsi, après avoir appliqué l'ensemble de ces découpes, nous obtenons l'abre de processus (*process tree*) complet retourné par l'algorithme et dont l'expression littérale est donnée par l'équation 2.12.

$$\rightarrow (\times(\circlearrowleft (\rightarrow a, b), c), d), \curlywedge(e, f)) \tag{2.12}$$

Génération du réseau de Petri. Le réseau de Petri final est déductible de l'arbre de processus (*process tree*). En effet, l'utilisation de la figure 2.22 permet de convertir l'expression obtenue en un réseau de Petri en montrant comment transformer chaque lien entre deux éléments E_1 et E_2 de l'arbre de processus (*process tree*). C'est lors de cette étape que l'on peut voir apparaître les activités silencieuses (*silent activities*) qui sont introduites par l'opérateur parallèle.

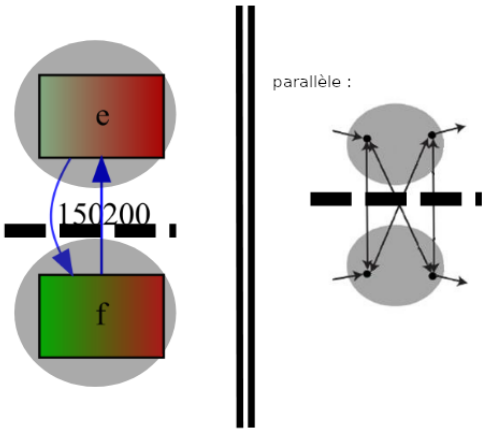


FIGURE 2.21 – Cinquième étape de découpe du graphe des suivants directs avec l'opérateur parallèle

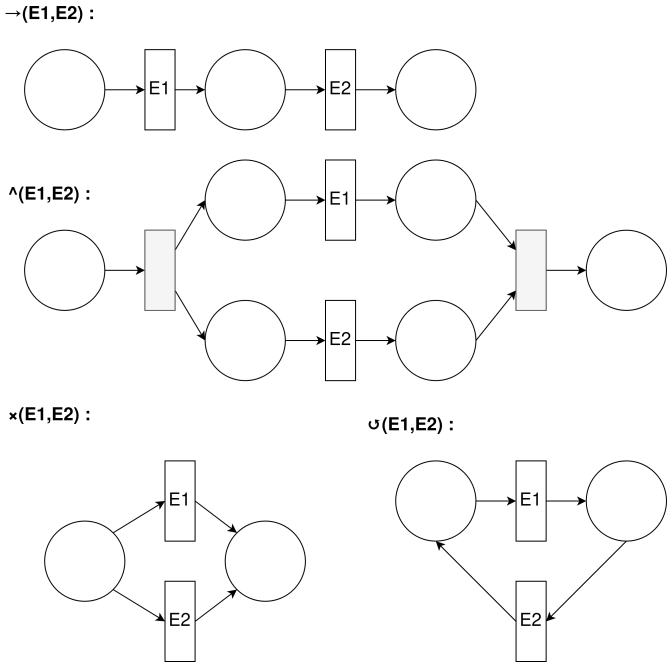


FIGURE 2.22 – Conversion de l'arbre de processus (*process tree*) en réseau de Petri

Finalement, du journal d'événements initial, nous obtenons le réseau de Petri de la figure 2.23 L'algorithme de l'inductive miner permet également de filtrer les arêtes peu présentes dans le journal d'événements. Ainsi, il est possible de prendre en compte uniquement les transitions qui représentent une proportion du jeu de données supérieure à un seuil prédéfini. Par exemple, si l'on considère le journal d'événements de notre exemple, qui contient 1000 transitions, avec un seuil de 5%, alors les liens entre b et c ainsi qu'entre c et a n'apparaîtront pas sur le modèle. En effet, ces transitions représentent 5% du journal d'événements et seront donc ignorées si le seuil est placé à 5%. Un tel procédé permet à la fois d'ignorer les données aberrantes mais surtout de réduire la complexité des réseaux de Petri et donc des modèles comportementaux générés.

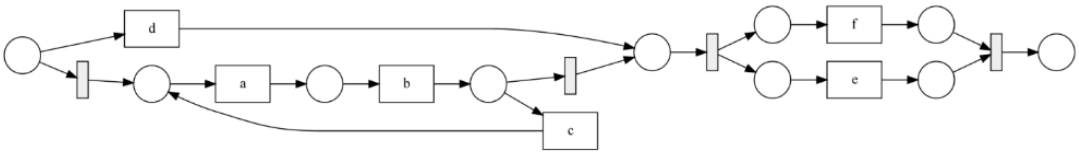


FIGURE 2.23 – Réseau de Petri obtenu par l'inductive miner

2.4.2 Algorithme du transition system miner

Le *transition system miner* est la seconde approche de *process mining* considérée [151]. Cette approche, bien qu'appartenant elle aussi à la famille du *process mining* s'éloigne du fonctionnement de l'*inductive miner* en faisant intervenir la notion de région pour générer les modèles. L'algorithme va construire un système de transitions (*transition system*) dans ses étapes intermédiaires. Celui-ci est un graphe (S, E, T) où S représente les états, E les événements présents dans le journal d'événement et T qui sont les transitions entre les états avec leur événement associé. L'algorithme comporte trois étapes dont la première d'entre elles consiste à créer ce système de transitions. On considère que les états du système ne sont pas explicitement donnés par les données en entrée et qu'il faut donc les paramétrer. Pour cela, l'algorithme permet de prendre en compte les valeurs passées et futures des différents attributs, ainsi que d'éventuelles connaissances supplémentaires afin de définir les états du système. La figure 2.24 résume les quatre paramètres à définir pour faire fonctionner l'algorithme et qui vont servir à extraire les états du système étudié. De cette façon, le *transition system miner* va être amené à considérer, dans le jeu de données, soit exclusivement les données passées ou futures pour définir un état du système, soit un ensemble de celles-ci. De plus, l'algorithme permet également de prendre en compte des connaissances complémentaires ne faisant pas directement partie du jeu de données lors de son paramétrage. Ainsi, il est possible de renseigner directement l'état d'un système pour une partie du jeu de données. Lors de nos tests avec cet algorithme, nous avons utilisé exclusivement les jeux de données sans faire d'hypothèse quant à l'état du système.

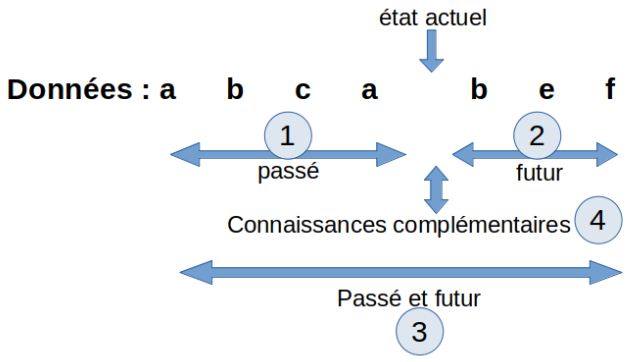


FIGURE 2.24 – Paramètres permettant de caractériser un état

Dans notre cas, nous ne considérerons pas d'éléments futurs, car notre but est de détecter au plus tôt les anomalies. Dans ce contexte, attendre d'avoir accès aux données des points suivants avant d'effectuer notre analyse n'est pas souhaitable. Ainsi, nous n'utiliserons que les données passées pour l'analyse. Pour notre exemple, nous avons pris en compte l'événement

courant ainsi que le précédent pour construire un système de transitions. Ainsi, en reprenant le journal d'événements J, utilisé pour illustrer le fonctionnement de l'*inductive miner*, l'algorithme construit un système de transitions où chaque état qui le compose comprend au maximum 2 événements.

$$J = [< a, b, c, a, b, e, f >^{50}, < a, b, f, e >^{100}, < d, e, f >^{100}, < d, f, e >^{100}] \tag{2.13}$$

Pour le construire, l'algorithme va utiliser un premier jeu de données Afin d'illustrer la construction du système de transition, nous avons choisi de considérer le jeu de données $< d, e, f >$. Initialement, l'algorithme n'a aucune information sur le système et va donc représenter l'état de celui-ci par []. Puis, il va rencontrer le premier événement d ce qui va amener le système dans l'état $[d]$. Après cela, il va traiter l'événement e, en conservant en mémoire la précédente transition d, ce qui amène le système dans l'état $[e, d]$. Finalement, en rencontrant f, l'algorithme va créer autre état pour le système. Si nous n'avions pas limité la définition des états aux 2 dernières transitions alors l'état décrivant le système serait $[f, e, d]$, mais avec cette limite l'algorithme trouve l'état $[f, e]$. La figure 2.25 est obtenue après avoir utilisé l'ensemble des jeux de données de J.

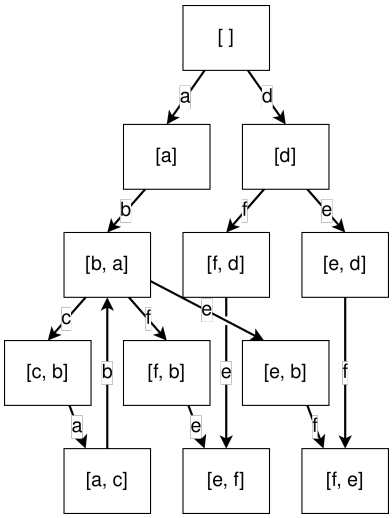


FIGURE 2.25 – Système de transitions obtenu en considérant les deux précédents points de données pour en déduire l'état

Lors de la seconde étape de l'algorithme *transition system miner*, plusieurs techniques sont utilisées afin de simplifier le système de transitions qui sera généré. On note par exemple la suppression des boucles ainsi que le regroupement des états ayant les mêmes entrées et sorties. Enfin, au cours de la troisième étape, un réseau de Petri va être généré à partir du système de transitions. Cette transformation est menée en se basant sur le concept de régions. Elles correspondent à un groupe d'états remplissant plusieurs conditions que nous décrivons par la suite. Chacune des régions minimales que nous trouverons sera une place du réseau de Petri. Ainsi pour un système de transitions TS = (S, E, T) et S' un ensemble d'états comprenant certains éléments de S, si S' est une région alors l'une des affirmations suivantes est vraie pour chaque événement e de E :

• Il y a S_1 et S_2 dans S, de telle sorte que $S_1 \xrightarrow{e} S_2$ aille dans S', c'est-à-dire que S_1 n'est pas dans S' mais que S_2 l'est, comme montré par la figure 2.26. Dans notre exemple, pour nous

retrouver dans cette situation, l'état [] peut jouer le rôle de S_1 tandis que les états $[a]$ et $[a, c]$ seraient dans S_2. S' serait l'ensemble des états en dehors de [].

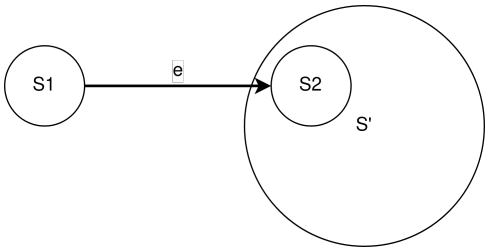


FIGURE 2.26 – Définition d'une région S_1 vers S_2 où S_2 est dans S'

• Il y a S_1 et S_2 dans S, de telle sorte que $S_1 \xrightarrow{e} S_2$ sorte de S', c'est-à-dire que S_1 est dans S' mais que S_2 ne l'est pas, comme montré par la figure 2.27. Nous pouvons nous trouver dans cette situation lorsque les états [] et $[c, b]$ appartiennent à S_1 tandis que les états $[a]$ et $[a, c]$ seraient dans S_2. S' serait l'ensemble des états en dehors de $[a]$ et $[a, c]$.

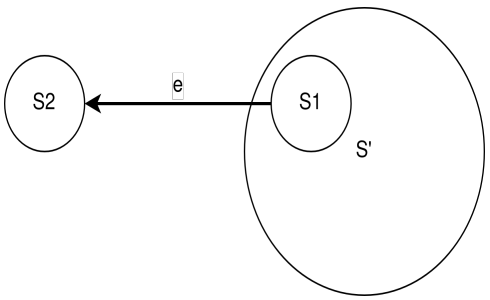


FIGURE 2.27 – Définition d'une région S_1 vers S_2 où S_1 est dans S'

• Il y a S_1 et S_2 dans S, de telle sorte que $S_1 \xrightarrow{e} S_2$ ne traverse pas S', c'est-à-dire que soit S_1 S_2 sont dans S' soit ils ne le sont pas, comme montré par la figure 2.28. Pour illustrer ce cas de figure, nous considérons l'état $[b, a]$ comme étant S_1 tandis que les états $[a]$ et $[a, c]$ joueront le rôle de S_2. Dans cette configuration, S' comprendre l'ensemble des états du système de transitions.

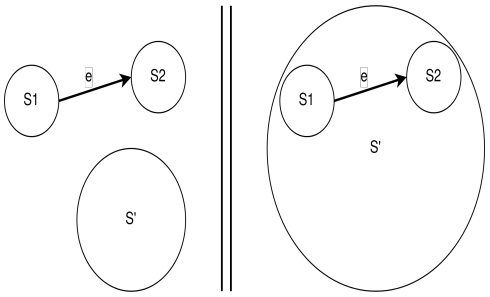


FIGURE 2.28 – Définition d'une région S_1 vers S_2 où S_1 et S_2 sont ou non dans S'

Le réseau de Petri que nous obtenons pour notre exemple, en utilisant l'algorithme *transition system miner*, est donné par la figure 2.29.

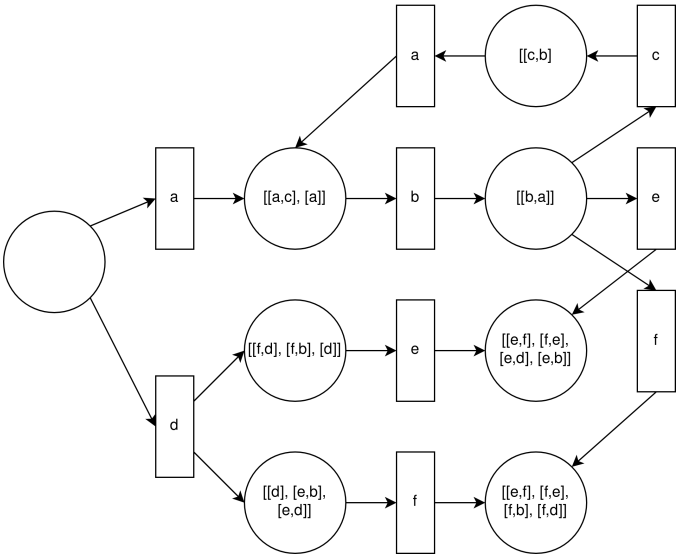


FIGURE 2.29 – Réseau de Petri obtenu par le *transition mining system*

2.5 Synthèse

Dans ce chapitre, nous avons commencé par définir les systèmes IoT et présenter leurs caractéristiques, qui en font des environnements particulièrement vulnérables. Nous avons ensuite analysé les principales attaques auxquelles ils sont confrontés, notamment au regard de la classification établie par l'OWASP, et avons associé à celles-ci un ensemble d'approches contribuant à leur sécurisation. Bien que ces approches intègrent des architectures de sécurité avec des mécanismes d'authentification et de chiffrement, les méthodes de détection restent essentielles pour identifier et caractériser des comportements anormaux, qui peuvent révéler des attaques potentielles, incluant des attaques internes aux systèmes IoT. Nous avons ensuite décrit trois grandes catégories de méthodes de détection, et plus particulièrement les algorithmes non supervisés et semi-supervisés. Le principal problème que l'on rencontre avec ces algorithmes est leur difficulté à détecter des séquences anormales de données. Cela est principalement dû à leur fonctionnement avec un traitement point de données par point de données, qui ne permet pas de prendre en compte l'évolution du système IoT. Les rares méthodes qui le permettent, comme HMM ou LSTM, sont couramment utilisées à cette fin, mais génèrent des modèles qui s'avèrent être souvent complexes. Il est nécessaire à la fois de considérer l'aspect historique des données, ainsi que la complexité des modèles produits. C'est pourquoi, nous avons terminé le chapitre en présentant des méthodes de *process mining*, qui analysent l'enchaînement des points de données et construisent des modèles suffisamment compréhensibles pour être exploitables par un opérateur du système. Un verrou important, cependant, est que ces méthodes peuvent rapidement produire une explosion des états, lorsqu'on les applique à des systèmes IoT hétérogènes.

Chapitre 3

Approche de détection utilisant le *process mining* pour l'IoT

Sommaire

3.1 Introduction 37
3.2 Architecture 38
3.2.1 Principaux composants et interactions 38
3.2.2 Collecte des données 38
3.2.3 Modélisation des données d'entrée 39
3.2.4 Description des modèles de comportement utilisés par le *process mining* 40
3.3 Phase d'apprentissage 41
3.3.1 Normalisation des données numériques 41
3.3.2 Partitionnement de données 43
3.3.3 Découpe temporelle 45
3.3.4 Construction des modèles de comportement 46
3.4 Phase de détection 47
3.4.1 Normalisation des données numériques 47
3.4.2 Regroupement par la distance euclidienne 48
3.4.3 Découpe temporelle 49
3.4.4 Méthode d'alignement d'un modèle avec les données 49
3.4.5 Mécanisme de détection d'anomalies 50
3.5 Synthèse 52

3.1 Introduction

L'utilisation de systèmes composés en partie ou exclusivement d'objets connectés, c'est-à-dire entrant dans le cadre de ce qu'on appelle communément l'Internet des objets (IoT), a fortement progressé dans de nombreux domaines d'application. Les principaux problèmes de ces systèmes, qui peuvent être complexes, résident dans l'hétérogénéité de leurs protocoles et dans la limitation de leurs ressources en matière de mémoire, de calcul ou bien encore de batterie. De plus, les objets connectés peuvent comporter des faiblesses de conception permettant à un attaquant de prendre le contrôle du système IoT. Pour pallier le problème, nous avons considéré, dans le chapitre précédent, différentes méthodes de détection des anomalies et notamment les méthodes

de la famille du *process mining* qui présentent des caractéristiques intéressantes. En effet, il permet de possiblement détecter des attaques peu ou pas connues en cherchant les déviations par rapport au comportement normal. De plus, le *process mining* ne prend pas seulement en compte un unique point de données, mais bien l'ensemble de l'historique. Cela permet de ne pas exclusivement évaluer si un point de données est anormal mais également de voir si une séquence d'évènements dévie du comportement normal. Son utilisation nécessite d'inclure une partie de pré-traitement des données afin de convertir celles-ci en un format exploitable et compréhensible par les algorithmes de *process mining*.

Dans les sections suivantes, nous détaillons dans un premier temps l'architecture de la solution de détection proposée. Puis, nous décrivons les éléments intervenant lors de la phase d'apprentissage avant de faire de même avec la phase de détection. Finalement, dans la synthèse, nous faisons un bilan du travail présenté dans ce chapitre.

3.2 Architecture

Dans cette section, nous décrivons plus en détails l'architecture utilisant le *process mining* dans le but de détecter les anomalies présentes dans un jeu de données provenant d'un système IoT. Ainsi, dans un premier temps, nous discutons de l'architecture globale dans laquelle s'inscrit notre approche de détection. Puis, nous expliquons la façon dont les données ont été collectées et sous quel format ces dernières ont été générées. Finalement, nous donnons un exemple volontairement simple d'un réseau de Petri, qui est le support utilisé pour la représentation des modèles de comportement pour notre solution de détection d'anomalies.

3.2.1 Principaux composants et interactions

Le système de détection d'anomalies que nous avons développé, et qui repose sur le *process mining*, est illustré dans la figure 3.1. On peut y voir les trois principales parties représentées par trois blocs rectangulaires : le bloc de *pré-traitement des données* qui intervient initialement, puis deux blocs *process mining*, le premier pour la *construction des modèles* de comportement et le second qui fait une *comparaison de similarités* entre les données à tester et les modèles de comportement afin de détecter de possibles anomalies.

Pendant la phase d'apprentissage (flèches bleues dans la figure 3.1), les données brutes sont transformées par le bloc de *pré-traitement* pour veiller à ce que les données soient fournies dans un format interprétable par le bloc de *construction de modèles*. En général, les algorithmes de *process mining* utilisent en entrée des journaux d'événements. Cependant, la phase de pré-traitement permet d'utiliser des formats de données d'entrée hétérogènes.

L'objectif au cours de la phase de détection (flèches rouges de la figure synthétisant l'architecture) est de détecter des déviations à partir des données brutes. D'ailleurs, les transformations subies par les données dans le cadre de la détection d'anomalie partagent un certain nombre de points communs avec la phase de création de modèles. En effet, les données brutes sont également pré-traitées avant de comparer leur similarité avec les modèles enregistrés en amont. Une description des données d'entrée est fournie dans la section 3.2.3.

3.2.2 Collecte des données

Les ressources limitées des objets connectés faisant partie de systèmes IoT complexes rendent difficile voire impossible le traitement des données directement dans l'objet. Il est donc nécessaire de faire remonter un ensemble varié d'informations à un système externe dédié à l'analyse de

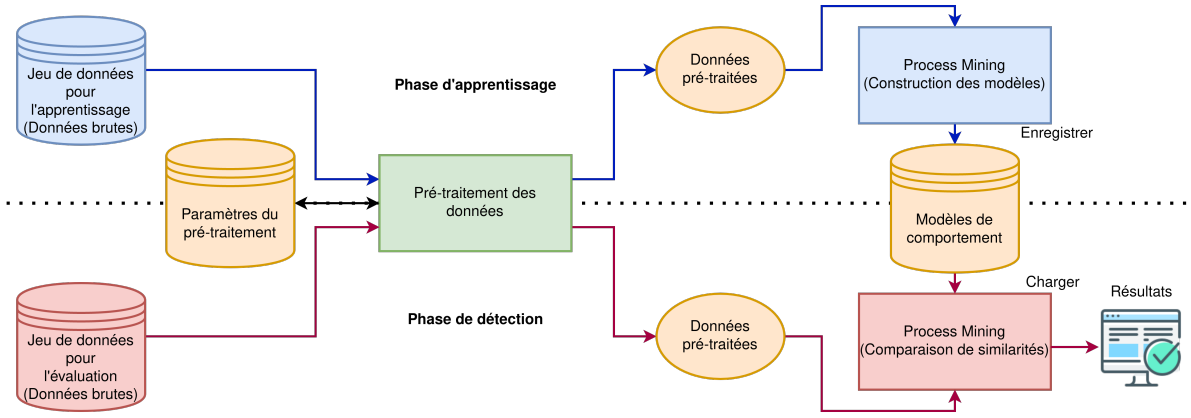


FIGURE 3.1 – Vue d'ensemble de l'architecture du système de détection d'anomalies

ces données et dont le but est de détecter au plus tôt de possibles attaques apparaissant comme des anomalies. Cette remontée d'informations se fait à l'aide de capteurs soit numériques, soit physiques, installés sur l'objet surveillé. Ces données sont regroupées en plusieurs ensembles représentant chacun un type de données. Par exemple, dans le cas de données provenant d'un véhicule connecté, l'un de ces ensembles peut contenir toutes les données relatives à la conduite telles que la vitesse du véhicule ou bien encore le régime moteur. Un second ensemble peut contenir les données relatives aux interactions réseaux de l'objet comme le nombre de bits échangés ainsi que des informations sur les systèmes interagissant avec celui que nous surveillons. Dans le but de détecter au plus tôt les anomalies, les données collectées sont envoyées pour être mises à disposition, avec un minimum de délais, au système de détection d'anomalies.

3.2.3 Modélisation des données d'entrée

Les données brutes en entrée du système de détection sont constituées de plusieurs enregistrements où le ième enregistrement est appelé Er_i. La formule 3.1 présente les données dans le cas où le jeu de données est constitué de n enregistrements.

$$Données = \{Er_1, \ldots, Er_n\} \tag{3.1}$$

Chacun des enregistrements Er_i du jeu de données contient m éléments E_{ij}, comme décrit dans la formule 3.2, tandis que chaque élément est composé d'un attribut associé à une valeur. Une description est donnée par la formule 3.3. Dans le but d'obtenir des résultats exploitables, il est nécessaire que l'ensemble des différents attributs de chacun des enregistrements soient identiques. En effet, que ce soit lors de la phase de construction des modèles ou de la phase de détection, les données doivent conserver une certaine forme de cohérence, comme le montre la formule 3.4.

$$\forall i \in [\![1, n]\!], Er_i = \{E_{i1}, \ldots, E_{im}\} \tag{3.2}$$

$$\forall i \in [\![1, n]\!], \forall j \in [\![1, m]\!], E_{ij} = \{attribut_{ij} : valeur_{ij}\} \tag{3.3}$$

$$Soit\ j\ \in [\![1, m]\!], \forall i, k \in [\![1, n]\!], attribut_{ij} = attribut_{kj} \tag{3.4}$$

Afin d'illustrer au mieux cette décomposition, nous allons considérer l'exemple d'un jeu de données fourni par un véhicule connecté contenant cinq enregistrements eux-mêmes composés de trois éléments. Les attributs de ces éléments étant l'angle pris par le volant, la vitesse du véhicule et, enfin, la vitesse enclenchée au niveau de la boîte de vitesses. Avec cet exemple, l'équation 3.1 rend compte du nombre d'enregistrements présents dans le jeu de données comme cela est montré dans la formule 3.5, tandis que la formule 3.6 montre le nombre d'éléments dans chacun des enregistrements de notre exemple.

$$Données = \{Er_1, Er_2, Er_3, Er_4, Er_5\} \tag{3.5}$$

$$\forall i \in [\![1, 5]\!], Er_i = \{E_{i1}, E_{i2}, E_{i3}\} \tag{3.6}$$

Chacun de ces enregistrements est composé des trois éléments mentionnés précédemment. Ainsi, la formule 3.3 permet de lister, pour un enregistrement donné, le nom de ces trois éléments, que l'on nomme attribut, ainsi que leurs valeurs respectives. On peut retrouver cette description dans l'équation 3.7 où l'on considère le quatrième enregistrement.

$$E_{41} = \{angle\_volant : 90°\} \tag{3.7}$$

$$E_{42} = \{vitesse : 25km.h^{-1}\} \tag{3.8}$$

$$E_{43} = \{vitesse\_enclenchée : 2\} \tag{3.9}$$

La formule 3.4 est ici pour montrer que nous allons maintenir une certaine cohérence entre deux enregistrements. Ainsi, si l'on reprend notre exemple, l'attribut du deuxième élément de chacun des enregistrements correspondra à la vitesse du véhicule, comme cela est montré par l'équation 3.10. En revanche, bien que les noms des attributs soient identiques, leurs valeurs peuvent être différentes.

$$attribut_{12} = attribut_{22} = attribut_{32} = attribut_{42} = attribut_{52} = vitesse \tag{3.10}$$

Dans la sous-section suivante, nous montrons sous quelle forme sont enregistrés les modèles de comportement que nous générons pour traduire le comportement normal du système IoT étudié.

3.2.4 Description des modèles de comportement utilisés par le *process mining*

Les méthodes de *process mining* reposent sur une analyse de conformité des données par rapport à des modèles représentés par des réseaux de Petri. Ces modèles permettent de représenter le système IoT en se basant sur un ensemble d'événements discrets. Ces réseaux de Petri, dont un exemple est donné par la figure 3.2, sont des graphes composés à la fois d'un sous-ensemble de places, représentées par des cercles, et de transitions, illustrées par des rectangles. Ces transitions correspondent aux événements indiquant un changement d'état du système. Si l'on reprend notre exemple du véhicule connecté, ce réseau de Petri peut, par exemple, décrire dans la place $P1$ un véhicule qui avance en ligne droite avec une certaine vitesse. Depuis cet état, le véhicule a la possibilité de ralentir son allure traversant ainsi la transition $T1$ et arrivant dans la place $P2$. Après avoir ralenti suffisamment, le véhicule peut alors tourner à droite et donc, traverser la transition $T2$ pour arriver dans la place $P3$ qui symbolise le véhicule se déplaçant à faible vitesse et avec un angle du volant permettant de le faire tourner à droite. Finalement, le véhicule revient à sa vitesse de croisière en ligne droite que décrit la place $P1$ après être passé par la transition $T3$.

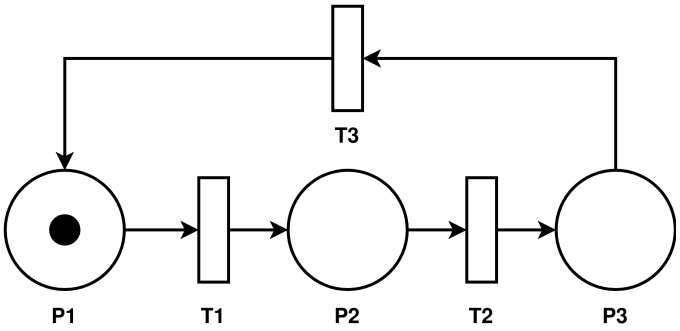


FIGURE 3.2 – Exemple d'un réseau de Petri

Les modèles obtenus décrivent les possibles évolutions du système IoT observé et permettent d'avoir immédiatement une idée du fonctionnement de celui-ci. L'un des inconvénients majeurs de ces algorithmes et l'utilisation d'un format très normalisé des données d'entrée ce qui empêche leur généralisation à des systèmes faisant remonter des données sous une autre forme. Nous voyons dans la section suivante comment pallier ce problème.

3.3 Phase d'apprentissage

Dans cette section, nous décrivons les différentes étapes de la phase d'apprentissage. Le but de cette phase est de construire des modèles du comportement des systèmes IoT étudiés. Cela permet, par la suite, de détecter d'éventuelles déviations qui peuvent traduire le résultat d'une attaque du système. La construction des modèles repose principalement sur l'utilisation d'algorithmes de *process mining*. Il est toutefois nécessaire de transformer les données d'entrée car ces dernières ne sont pas directement utilisables par ces algorithmes. En effet, il existe un certain nombre de problèmes à traiter avant de pouvoir appliquer un algorithme de *process mining* à nos données. Tout d'abord, un partitionnement doit leur être appliqué pour faire ressortir ce que nous considérons comme les états possibles que peut prendre le système IoT. Ensuite, une fois les données partitionnées, il est nécessaire de les transformer en un format exploitable par l'algorithme. Dans la suite de cette section, nous traitons de la normalisation des données numériques, qui nous permet d'obtenir un meilleur partitionnement, ainsi que de la découpe temporelle qui vise à découper les données pour que les modèles générés restent suffisamment simples et généralisables.

3.3.1 Normalisation des données numériques

Le bloc de *pré-traitement des données* est composé de trois sous-blocs, comme montré sur la figure 3.3, le bloc de *normalisation* est le premier d'entre eux. Ce bloc permet une remise à l'échelle des attributs ayant des valeurs numériques, en excluant les attributs catégoriques et booléens. Ainsi, les attributs correspondant aux données numériques sont stockées dans une liste notée L_{num}, tandis que les autres données qui ne nécessitent pas de traitement préliminaire sont stockées dans la liste notée $L_{\overline{num}}$. Le but est d'homogénéiser l'impact des différents attributs avant de procéder au partitionnement des données. En effet, lors du partitionnement, la plupart des algorithmes vont calculer la distance entre les différents points pour en déduire des sous-groupes. Ainsi, traiter des attributs ayant des ordres de grandeur différents ne doit pas impacter le partitionnement. L'utilisation d'un attribut dont les valeurs se situent entre zéro et cent aura

donc le même impact qu'un autre attribut qui aurait des valeurs comprises entre un et dix millions. Après une normalisation, ce sont les variations des valeurs des différents attributs qui sont déterminantes pour classifier les points de données.

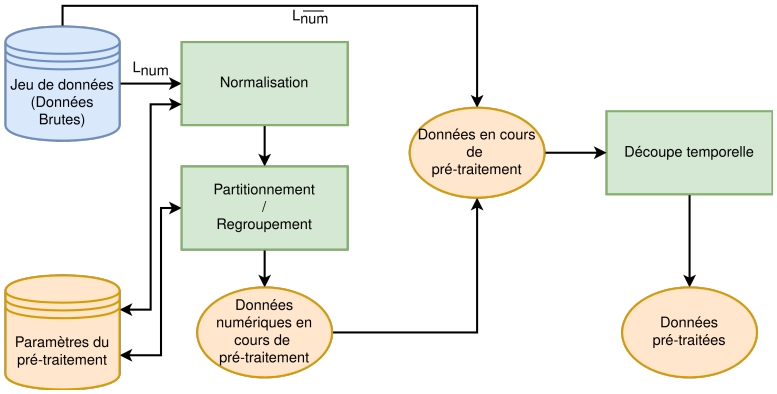


FIGURE 3.3 – Vue détaillée du sous-bloc de pré-traitement des données

En reprenant la description du jeu de données, définie à partir de la formule 3.1, nous introduisons la formule 3.11 qui indique sur quelle partie des données est appliquée la normalisation.

$$\begin{gathered} \forall i \in [\![1,n]\!], \forall j \in [\![1,m]\!], \\ Entrée_{normalisation_i} = \{E_{1j}, \ldots, E_{nj}\} \backslash A \\ où\ A = \{E_{ij} \,|\, attribut_{ij} \in L_{\overline{num}}\} \end{gathered} \tag{3.11}$$

Nous avons choisi d'utiliser les deux méthodes de normalisation existantes les plus couramment utilisées. Ainsi, le choix de la méthode à utiliser sera l'un des paramètres de notre solution de détection. La première de ces méthodes, appelée min-max, est décrite par l'équation 3.12, elle va projeter la valeur des différents attributs numériques sur l'intervalle [0,1]. De cette façon, après la normalisation, c'est l'évolution des valeurs des différents attributs qui sera déterminante pour classifier les points de données. Concrètement, pour trouver la valeur normalisée d'un attribut d'un enregistrement précis nous retirons la valeur minimale, X_{min}, qu'a pris cet attribut dans le jeu de données, puis nous divisons cette valeur par l'intervalle des valeurs possibles de l'attribut, $X_{max} - X_{min}$.

$$X_{min-max} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{3.12}$$

La seconde méthode est la cote standard, introduite par l'équation 3.13, elle transforme les données numériques pour que les valeurs relatives à chaque attribut puissent renvoyer, après traitement, la distance de ces valeurs par rapport à la moyenne. Cette normalisation, par rapport à la normalisation min-max, donne plus de poids aux valeurs extrêmes des attributs tout en réglant le problème d'échelle entre les valeurs des différents attributs. On trouve la nouvelle valeur normalisée d'un attribut pour un enregistrement particulier en lui retirant la valeur moyenne, notée μ, de l'attribut sur l'ensemble du jeu de données, puis en divisant cette valeur par l'écart-type σ.

$$X_{cote\_standard} = \frac{X - \mu}{\sigma} \tag{3.13}$$

D'autres méthodes de normalisation, comme celle utilisant la tangente hyperbolique, donnée par l'équation 3.14, présentent certains avantages. Pour cette méthode, les données numériques sont transformées sur le même modèle que pour la normalisation de la cote standard, c'est-à-dire en retirant la moyenne prise par l'attribut, μ, avant de diviser par la valeur de l'écart-type σ. Puis, cette valeur est multipliée par un coefficient avant de passer par une fonction de tangente hyperbolique qui va limiter à un intervalle strict les valeurs des différents attributs. Les premiers résultats obtenus en utilisant cette méthode de normalisation étaient très proches de ceux obtenus avec la cote standard. Nous avons donc décidé d'utiliser exclusivement les méthodes min-max et cote standard qui nous évitent d'ajouter un paramètre supplémentaire dans notre système.

$$X_{tangente\_hyperbolique} = 0.5 \times tanh(0.001 \times \frac{X - \mu}{\sigma}) \tag{3.14}$$

Les paramètres de normalisation utilisés dans la phase d'apprentissage sont réutilisés lors de la phase de détection afin d'assurer l'homogénéité de traitement pendant les deux phases. Nous nous assurons ainsi que la transformation des données de la phase d'apprentissage et de détection soient exactement les mêmes pour que la comparaison des valeurs obtenues ait un sens.

3.3.2 Partitionnement de données

Dans le sous-bloc de *partitionnement*, les données normalisées sont regroupées dans des groupes qui définiront, en partie, les différents états possibles que peut prendre le système IoT. Les techniques de partitionnement sont souvent utilisées pour l'exploration de données. Dans notre cas, ces techniques nous permettent de réduire le nombre d'états servant à caractériser un système utilisant des objets connectés. Ainsi, chaque multiplet contenant les données numériques sera associé à un identifiant de groupe, noté Gr_i comme indiqué dans la formule 3.15. Cette formule montre sous quelle forme se présentent les données issues du ième enregistrement Er_i à la fin du partitionnement. Ce résultat est noté $Sortie_{partitionnement_i}$. L'équation 3.16 résume les informations sauvegardées et stockées pour le $Groupe_{Gr_i}$ qui est le groupe associé au ième point du jeu de données. Nous trouvons l'identifiant Gr_i et le barycentre $Barycentre_{Gr_i}$ du groupe ainsi que la distance maximale $Dmax_{Gr_i}$ entre les données normalisées et le barycentre du ième groupe. Ces éléments seront réutilisés lors de la phase de détection. Une fois la normalisation appliquée sur les données appartenant à L_{num}, il reste à intégrer les données de $L_{\overline{num}}$ pour prendre en compte tous les éléments des enregistrements. Ainsi, en sortie de partitionnement, chaque enregistrement comportera un identifiant de groupe Gr_i, dans le cas où il y aurait au moins un élément dans L_{num}, ainsi que l'ensemble des éléments dont les attributs sont dans $L_{\overline{num}}$.

$$\begin{gathered} \forall i \in [\![1, n]\!], \forall j \in [\![1, m]\!], \\ Sortie_{\text{partitionnement}_\text{i}} = \{E_{i1}, \ldots, E_{im}, Gr_i\} \backslash C \\ \text{où } C = \{E_{ij} \,|\, attribut_{ij} \in L_{num}\} \end{gathered} \tag{3.15}$$

$$Groupe_{Gr_i} = \{Gr_i, Barycentre_{Gr_i}, Dmax_{Gr_i}\} \tag{3.16}$$

Le partitionnement des données est un élément important de notre approche. Son but est de regrouper les données similaires dans un jeu de données. L'utilisation de tels algorithmes permet d'aider à identifier des groupements de points présentant des données proches et donc, dans notre cas, de mettre en avant des caractéristiques propres aux différents états possibles du

système IoT. Il existe plusieurs types d'algorithmes de partitionnement qui utilisent des stratégies différentes pour agréger les données. Dans la suite de cette sous-section, nous décrivons le principe de fonctionnement des trois principales familles de méthodes de partitionnement. Ainsi, nous décrivons le fonctionnement des algorithmes de partitionnement classiques, puis nous détaillons celui des algorithmes de partitionnement hiérarchiques, enfin nous abordons les méthodes de partitionnement utilisant la densité des données. Notre architecture permet l'utilisation d'un algorithme de partitionnement de chacune de ces trois familles d'algorithmes de partitionnement.

Les algorithmes de partitionnement classiques. Leur but est de minimiser un critère, le plus souvent la distance entre le point de données et les groupes présents. Ces algorithmes attribuent itérativement chacun des points de données à un groupe, ce qui définit ces groupes va évoluer en fonction des points de données qui seront distribués. Chaque groupe doit contenir au moins un point de données et chaque point de données doit forcément appartenir à un seul groupe. Dans notre solution de détection, nous proposons d'utiliser K-MEANS [109] qui est l'un des algorithmes de partitionnement classiques les plus simples et certainement le plus utilisé. Dans un premier temps, l'algorithme prend K points de données en tant que centre des groupes. Il n'y a pas de règle particulière pour sélectionner ces points, cela peut être fait de façon aléatoire ou directement choisi par l'utilisateur. Après cela, chaque point du jeu de données est associé au groupe pour lequel la distance les séparant est la plus faible. Après cette association, les centres des groupes sont recalculés en prenant la moyenne des points présents dans le groupe et la phase d'association est répétée. L'algorithme s'arrête lorsqu'il n'y a aucun changement entre deux répétitions au niveau de l'attribution des points aux groupes.

Les algorithmes de partitionnement hiérarchique. Ils peuvent être représentés comme des dendrogrammes, c'est-à-dire sous la forme d'un diagramme organisant les données en arborescence en fonction de leurs similarités. Dans le cas de ces algorithmes, chaque feuille de l'arborescence est un point de données. Un exemple de dendrogrammes est donné par la figure 2.13 (b) du chapitre 2. Le fonctionnement de tels algorithmes peut se faire de deux manières : la première, dite de la division, est d'attribuer l'ensemble des points à un unique groupe et, à chaque étape, de séparer en deux le groupe présentant les données les plus éloignées. La seconde manière, dite agglomérante, considère, au contraire, chacun des points comme un groupe à part. Lors de chaque étape, l'algorithme va associer les deux groupes présentant le plus de similarité. Pour notre architecture, nous proposons d'utiliser l'algorithme BIRCH [167], qui a été conçu pour pouvoir traiter un nombre important de données. Il est nécessaire que l'utilisateur définisse les paramètres qu'il souhaite utiliser, comme le nombre maximum d'enfants que peut avoir un nœud qui n'est pas une feuille (*branching factor*) ou bien encore la valeur maximale du diamètre du groupe (*threshold*). Une fois l'arbre construit, il est possible de le reconstruire afin de limiter sa taille en augmentant le seuil au besoin. Une fois cette étape effectuée, un autre algorithme de partitionnement, comme K-MEANS, est utilisé sur les feuilles de l'arbre pour déterminer les groupes. Finalement, la dernière étape, optionnelle, est d'appliquer de nouveau l'algorithme K-MEANS sur l'ensemble des données en prenant comme groupes initiaux les K groupes mis en avant lors de l'étape précédente. L'un des principaux avantages de l'algorithme BIRCH est qu'il ne lit le jeu de données qu'une seule fois et qu'il insère dynamiquement les nouveaux points directement dans l'arbre. Ainsi, pour commencer à être utilisé, il ne nécessite pas d'avoir accès à toutes les données, ce qui peut présenter un avantage dans des configurations où l'on souhaite commencer la phase d'apprentissage au plus tôt. Ses inconvénients sont très similaires à ceux présentés par les algorithmes de partitionnement qui cherchent à minimiser un critère, dans le

sens où il ne fonctionne que pour des données complètement numériques et leurs performances peuvent être fortement réduites dans les cas où les groupes ne sont pas sphériques dans l'espace vectoriel où évoluent les données.

Les algorithmes de partitionnement utilisant la densité. Ils vont mettre en avant les zones où l'on trouve les plus fortes concentrations de points. L'idée derrière l'utilisation de ces méthodes est, lors de la phase de pré-traitement, de choisir ces zones de fortes concentration comme étant des groupes qui symbolisent les états possibles que peut prendre le système IoT. Nous proposons d'utiliser dans notre architecture l'algorithme DBSCAN, qui est l'algorithme utilisant la notion de densité le plus couramment utilisé. Dans un premier temps, il va prendre un point de données, qui n'a pas déjà été traité, aléatoirement dans le jeu de données. Il va ensuite regarder combien de voisins proches ce point possède. Dans le cas où il y a suffisamment de voisins, c'est-à-dire que ce nombre est supérieur au paramètre fourni à l'algorithme, alors le point de données va former un groupe, sinon il sera considéré comme un point aberrant. Lorsqu'un groupe est créé, l'algorithme va tenter de l'agrandir en lui fournissant l'ensemble de ses voisins et points atteignables. DBSCAN va répéter ces étapes jusqu'à ce que chaque point ait été traité. Les principaux avantages de cette méthode sont sa résistante au bruit et sa possibilité de trouver des groupes de formes non sphériques. En revanche, l'un de ses inconvénients est la difficulté de choisir les paramètres les plus pertinents. En effet, il va être nécessaire de notifier à l'algorithme le nombre minimal de points de données que doit comporter un groupe avant de pouvoir considérer celui-ci. De plus, le choix de la distance maximale au sein d'un groupe peut fortement changer les résultats de DBSCAN, et cette valeur va grandement dépendre des jeux de données à notre disposition.
Une fois le partitionnement effectué sur les données numériques, il nous reste à agréger les résultats obtenus avec les attributs booléens et catégoriels de $L_{\overline{num}}$ dans le but d'obtenir une description des états clés du système. Ainsi, les multiplets $Sortie_{partitionnement_i}$, obtenus pour chaque enregistrement Er_i et décrits par la formule 3.15, représentent les différents états pris par le système. Pour alléger la notation, nous notons $état_i$ l'identifiant d'état du ième point de données. Le nombre d'états différents, que l'on note p, est supérieur à 1 et ne peut pas excéder le nombre d'enregistrements dans le jeu de données.

Dans le cas où deux multiplets sont strictement identiques, nous considérons qu'ils décrivent le même état du système IoT, et ainsi partagent le même identifiant d'état. Par conséquent, il est possible de réduire chaque enregistrement Er_{i} du jeu de données à l'identifiant d'état et à l'horodatage, comme le montre l'équation 3.17. En effet, ces deux éléments résument l'intégralité des informations contenues dans un enregistrement.

$$Er_i = \{horodatage : t_i, état : état_i\} \tag{3.17}$$

3.3.3 Découpe temporelle

Dans le sous-bloc *découpe temporelle*, chaque enregistrement du jeu de données est découpé en k sous-parties que l'on note P_l et qui correspondent à des sous-ensembles du jeu de données transformés par les étapes de normalisation et de partitionnement. Pour chacune de ces sous-parties, l'horodatage de l'ensemble des points de données est inclus dans un intervalle temporel I. Cette découpe vise à réduire la complexité des modèles de comportement en évitant que ceux-ci ne représentent trop d'informations. Ainsi, on évite des modèles trop spécifiques et qui peuvent rencontrer des problèmes lors de la phase de détection. Par exemple, dans le cas d'un jeu de données d'un véhicule connecté, apprendre un modèle spécifique à un trajet particulier ne nous

permet pas de détecter efficacement des anomalies lors d'un nouveau trajet. Dans ce cas, il est plus intéressant de découper les données pour générer des modèles plus généraux. Ainsi, on peut obtenir des modèles résumant comment le véhicule va tourner à droite ou bien encore un modèle pour l'accélération en ligne droite. Ces modèles peuvent ainsi être utilisés pour d'autres trajets. Cette étape est décrite par la formule 3.18.

$$\forall a, b \in [\![1, n-1]\!] \; avec \; a \geq b \; et \; l \in [\![1, k]\!],$$
$$P_l = \begin{cases} Er_a & \\ \vdots & \text{avec } t_b\text{-}t_a \leq I \\ & \text{mais } t_{b+1}\text{-}t_a > I \\ Er_b & \end{cases} \tag{3.18}$$

3.3.4 Construction des modèles de comportement

Les algorithmes de *process mining* génèrent des modèles de comportement, sous la forme d'un réseau de Petri, à partir des différents sous-ensembles de données définis par l'équation 3.18. Ces modèles sont ensuite utilisés dans la phase de détection dans le but d'identifier des déviations par rapport au comportement normal attendu. Dans l'éventualité où aucun des modèles n'est suffisamment proche des données que l'on souhaite évaluer, on considère que le système IoT a un comportement anormal. Un tel écart peut souvent s'expliquer par l'attaque du système. Ces algorithmes de *process mining* permettent de générer des modèles de comportement à partir des données pré-traitées. Dans ce contexte, nous avons considéré deux algorithmes que sont l'algorithme de l'*inductive miner*, et l'algorithme du *transition system miner*. Ces deux algorithmes vont construire des réseaux de Petri qui représentent les évolutions possibles du système IoT étudié.

L'*inductive miner*

Cette méthode est la première approche que nous avons considérée. Un exemple de son fonctionnement est donné dans la sous-section 2.4.1 où nous avons introduit un certain nombre de notions telles que le graphe des suivants directs (*directly-follows graph*), l'activité silencieuse et l'arbre des processus (*process tree*). L'algorithme *inductive miner* est constitué de trois étapes permettant, à partir d'un ou de plusieurs jeux de données, de construire leurs modèles de comportement. Ainsi, dans un premier temps, l'algorithme construit un graphe des suivants directs à partir des données en sortie du bloc de *pré-traitement*. Il correspond à un graphe, où nous pouvons voir l'évolution du système IoT grâce à l'enchaînement des différents états mis en avant par le pré-traitement des données. Il est à noter que le graphe des suivants directs contient exclusivement des transitions visibles dans les données. Ainsi, si une transition entre deux états est visible sur ce graphe, alors on peut directement la retrouver dans les données nous ayant servi à le créer. Lors de la seconde étape, nous déduisons un arbre de processus (*process tree*) à partir du graphe des suivants directs (*directly-follows graph*). C'est un outil qui nous permet de représenter de façon compacte un réseau de Petri. Ainsi, nous pouvons générer le réseau de Petri correspondant à partir de l'arbre de processus.

Le *transition system miner*

Cette méthode est la seconde approche de *process mining* considérée [151]. L'algorithme comporte trois étapes. La première d'entre elles consiste à créer un système de transitions. Un système de transitions (*transition system*) est un graphe tripartite composé d'un ensemble d'états, d'événements et de transitions. L'algorithme fait l'hypothèse que nous n'avons pas les connaissances suffisantes pour définir les états du système grâce aux données en entrée, il faut donc les trouver à l'aide d'un paramétrage initial de l'algorithme. Pour faire cela, il peut prendre en compte les valeurs passées et futures des différents attributs, ainsi que d'éventuelles connaissances supplémentaires. Ces éléments lui permettent de déduire les états du système.

Dans le cas de notre exemple présenté dans la sous-section 2.4.2, nous avons exclusivement utilisé les éléments passés. En effet, comme nous souhaitons détecter au plus tôt les anomalies, attendre d'avoir accès aux données des points suivants avant d'effectuer notre analyse ne semblait pas souhaitable. Une fois le paramétrage de l'algorithme *transition system miner* et la création du premier système de transitions effectués, l'algorithme utilise plusieurs techniques afin de simplifier ce système. Il va par exemple supprimer des boucles ainsi que regrouper des états ayant les mêmes entrées et sorties. Enfin, un réseau de Petri est généré à partir de la dernière version du système de transitions. Cette transformation est menée en se basant sur le concept de régions, qui a été abordé dans la sous-section2.4.2, et permet de trouver les places du réseau de Petri.

Par la suite, nous avons favorisé l'algorithme de l'*inductive miner* qui nous permet de construire des modèles pendant la phase d'apprentissage en tenant compte de toutes les variations contenues dans le jeu de données initial. De plus, il reste le plus simple d'utilisation et n'ajoute pas de paramètre supplémentaire.

3.4 Phase de détection

Dans cette section, nous allons décrire la seconde phase qui correspond à la détection des anomalies. Nous distinguons les indices et transformations des données entre la phase d'apprentissage et de détection avec l'ajout de $'$. Par exemple, le nombre d'enregistrements de la phase de détection sera noté n'. Le principe de cette phase est d'analyser des données d'un système que l'on souhaite surveiller en les comparant aux modèles qui ont été générés dans la phase d'apprentissage. Dans les sous-sections suivantes, nous allons d'abord décrire comment sont adaptés la normalisation et l'équivalent du partitionnement des données pour la phase de détection. Ensuite, nous revenons sur la formalisation de la découpe temporelle des données transformées. Finalement, nous introduisons la métrique utilisée pour quantifier la déviation des données traitées par rapport aux modèles correspondant à des comportements normaux et nous expliquons le fonctionnement du mécanisme de détection d'anomalies.

3.4.1 Normalisation des données numériques

Dans le but de conserver une cohérence entre les données d'entrée et les modèles qui ont pu être générés en amont, il est nécessaire d'utiliser les mêmes paramètres de normalisation que ceux qui ont été utilisés pour la construction des modèles. Ainsi, nous utilisons X_{max} et X_{min} qui correspondent à la valeur maximale et la valeur minimale d'un attribut lors de la phase d'apprentissage. De la même façon, la moyenne et l'écart-type de la phase d'apprentissage sont notés μ et σ. Les équations 3.12 et 3.13 qui formalisent les méthodes de normalisation appliquées aux données pendant la phase d'apprentissage correspondent respectivement aux équations 3.19

et 3.20.

$$X'_{min-max} = \frac{X' - X_{min}}{X_{max} - X_{min}} \tag{3.19}$$

$$X'_{cote\_standard} = \frac{X' - \mu}{\sigma} \tag{3.20}$$

Utiliser les mêmes paramètres présente deux avantages. Tout d'abord les données en entrée peuvent être normalisées immédiatement. Puis, dans un second temps, la conservation des paramètres aidera à retrouver fidèlement les états du système IoT qui ont été mis en avant lors de la phase d'apprentissage. Une fois les données normalisées, le sous-bloc de *partitionnement / regroupement* permet d'associer, quand cela est possible, la partie des points de données qui appartiennent à L_{num} aux états trouvés lors du partitionnement de la phase d'apprentissage. Nous avons intégré ces deux méthodes de normalisation dans notre solution. L'utilisateur a donc le choix d'utiliser l'une ou l'autre.

3.4.2 Regroupement par la distance euclidienne

Dans cette sous-section nous décrivons la façon dont nous associons les groupes Gr_i de données numériques, trouvés dans la phase d'apprentissage, et apparaissant dans la formule 3.15, aux données que nous souhaitons évaluer. Une fois ce rapprochement effectué pour les données numériques, nous prenons en compte les données catégorielles et booléennes afin de trouver l'état du système. Pour faire ce rapprochement, nous utilisons la distance euclidienne entre le point du jeu de données d'évaluation avec l'ensemble des barycentres des groupes trouvés dans la phase d'apprentissage. Nous avons sélectionné cette distance car elle est la distance par défaut utilisée par les trois algorithmes de partitionnement intégrés à notre solution (K-MEANS, BIRCH et DBSCAN) qui nous ont servi pour construire les modèles de comportement dans la phase d'apprentissage. Nous considérons uniquement le groupe le plus proche du point à associer, c'est-à-dire le groupe candidat Gr_i possédant la plus faible distance par rapport au point, que l'on note $point_{i'}$, dont les données ont été normalisées. Concrètement, nous attribuons ce point de données au groupe candidat $Groupe_{Gr_i}$ à la condition que la distance entre ce point et le barycentre du groupe candidat, donnée par l'équation 3.21, ne soit pas supérieure à un coefficient proportionnel à la distance du point le plus éloigné du centre appartenant au groupe, noté $Dmax_{Gr_i}$.

$$\forall i \in [\![1, n]\!],\ soit\ i' \in [\![1, n']\!]\ tel\ que$$

$$Distance(point_{i'}, groupe\ candidat) = \min_{\forall i \in [\![1,n]\!]} (Distance(point_{i'}, Barycentre_{Gr_i})) \tag{3.21}$$

Ainsi, comme décrit par l'équation 3.22, le point de données $\mathrm{Er_{i'}}$ n'appartiendra au groupe que si la distance entre ces deux entités est inférieure à D_{max} multiplié par un facteur $1 + \epsilon$, où ϵ est une valeur paramétrable.

$$Soient\ i' \in [\![1, n']\!]\ et\ i \in [\![1, n]\!]$$

$$Si\ Distance(point_{i'}, groupe\ candidat) \begin{cases} \leq (1+\epsilon) Dmax_{Gr_i}\ alors\ Gr_i = Gr_{i'} \\ > (1+\epsilon) Dmax_{Gr_i}\ alors\ Gr_i \neq Gr_{i'} \end{cases} \tag{3.22}$$

Dans le cas où cette distance est supérieure à cette valeur, nous estimons que le point de données appartient à un nouveau groupe. Dès que les données normalisées et partitionnées de notre jeu de données sont attribuées à un groupe, nous allons de nouveau faire intervenir les données de $L_{\overline{num}}$ pour trouver à quels états appartient chacun des points. Après cela, les données en cours de pré-traitement passeront à l'étape suivante dans le sous-bloc de *découpe temporelle*.

3.4.3 Découpe temporelle

Lors de la phase de détection, dans le sous-bloc *découpe temporelle*, chaque enregistrement du jeu de données est découpé en k' sous-parties que l'on note $P'_{l'}$ et qui correspondent à des sous-ensembles du jeu de données initial. De la même façon que pour la phase d'apprentissage, la différence d'horodatage entre le premier enregistrement et le dernier de chacune de ces sous-parties est identique. Cette découpe vise à réduire la complexité des modèles de comportement en évitant que ceux-ci ne représentent trop d'informations. Ainsi, on évite de surcharger le mécanisme de détection d'anomalies et d'éventuellement retarder la détection de possibles attaques. L'intervalle entre la phase d'apprentissage et de détection d'anomalies peut être différent, nous noterons donc I' cet intervalle dans le cas de la détection. La formule 3.23 décrit le calcul de cet intervalle.

$$\forall a', b' \in [\![1, n'-1]\!] \; avec \; a' \geq b' \; et \; l' \in [\![1, k']\!],$$
$$P'_{l'} = \begin{cases} Er'_{a'} & \\ \vdots & \text{avec } t_{b'}\text{-}t_{a'} \leq I' \\ & \text{mais } t_{b'+1}\text{-}t_{a'} > I' \\ Er'_{b'} & \end{cases} \tag{3.23}$$

3.4.4 Méthode d'alignement d'un modèle avec les données

Il est important de quantifier la déviation des données d'évaluation par rapport aux modèles de comportement que nous obtenons dans la phase d'apprentissage. Ainsi, il est possible de détecter et d'évaluer d'éventuelles déviations dans le jeu de données. Cela nous permet également, en utilisant des données labélisées, de pouvoir évaluer par la suite les performances de notre méthode de détection d'anomalies. Pour illustrer le fonctionnement de notre mécanisme, considérons un modèle de comportement et un ensemble de données déjà pré-traitées. Dans un premier temps, nous avons besoin d'aligner ces deux éléments à l'aide d'une méthode dédiée. Concrètement, nous prenons chaque élément du jeu de données à tester et nous regardons si l'évolution entre deux éléments est possible sur le modèle. Si la transition présente dans les données ne peut pas être effectuée sur le modèle, le coût, initialement de 0, est augmenté de 1. Par contre, si le modèle et le journal d'événement, représenté par les données pré-traitées, sont synchronisés, alors le coût n'est pas augmenté. Le coût d'alignement est obtenu en additionnant l'ensemble des coûts intermédiaires. Le but de cette méthode est de trouver le coût d'alignement optimal, c'est-à-dire l'alignement de coût minimal. Pour illustrer cela, nous considérons un modèle de comportement

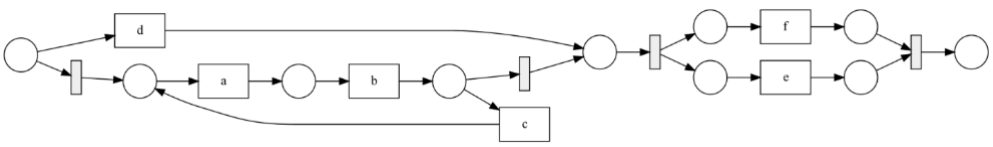


FIGURE 3.4 – Exemple de réseau de Petri

$M_{example}$, décrit par la figure 3.4, que nous utilisons pour tester les données pré-traitées de la formule 3.24, notées $Données_{exemple}$.

$$Données'_{\text{exemple}} = \begin{cases} \{\{horodatage : t_1\}, \{état : a\}\} \\ \{\{horodatage : t_2\}, \{état : b\}\} \\ \{\{horodatage : t_3\}, \{état : f\}\} \\ \{\{horodatage : t_4\}, \{état : e\}\} \end{cases} \tag{3.24}$$

Dans cet exemple, la table 3.1 donne la configuration où le coût d'alignement est maximal, c'est-à-dire que l'évolution dans les données et le modèle ne se font pas de façon synchronisée. Afin d'éviter ce problème, nous utilisons une méthode d'alignement qui va chercher à minimiser ce coût. Concrètement, dans cet exemple, la ligne du modèle va être décalée d'un cran sur la gauche, et le coût d'alignement va être calculé de nouveau. Ces étapes vont être répétées tant que l'ensemble des configurations n'auront pas été testées. La configuration pour laquelle la valeur du coût d'alignement est la plus faible sera retenue. Cette configuration optimale est celle présentée dans la table 3.2. Dans les deux tables, le symbole + indique les moments où le modèle et les données pré-traitées sont désynchronisés, et dans ce cas le coût augmente de 1. Le symbole τ indique des états "cachés/ fantômes" qui nécessitent d'être franchis afin de pouvoir atteindre les différents états possibles du système. Dans notre exemple, le coût optimal est de 1.

Données	a	b	f	e		+	+			+	
Modèle	+	+	+	+	τ	a	b	τ	τ	e	τ

TABLE 3.1 – Coût d'alignement maximal

Données		a	b			f	e	
Modèle	τ	a	b	τ	τ	+	e	τ

TABLE 3.2 – Coût d'alignement optimal

Une fois les deux éléments, que sont les données d'entrée et le modèle, alignés, nous utilisons une métrique pour évaluer leurs similarités. Cette métrique, que l'on note *fitness*, évalue la ressemblance entre les données et le modèle testé et permet donc de considérer que nous avons affaire à une anomalie si aucun modèle ne correspond suffisamment aux données d'entrée. Nous décrivons dans la sous-section suivante la façon dont est calculée cette métrique. Nous montrons également comment nous avons pris en compte les différents modèles générés dans la phase d'apprentissage.

3.4.5 Mécanisme de détection d'anomalies

Ainsi, une fois l'alignement effectué, nous quantifions la similarité entre les données d'entrée à tester et les modèles de comportement à notre disposition. La fitness, qui est la métrique utilisée, nous indique si un modèle, défini par un réseau de Petri noté P, permet d'interpréter correctement les données à évaluer, fournie sous la forme d'un journal d'événements noté J. Plus cette métrique est proche de 1, plus il y a de similarité entre le modèle et les données. La formule 3.25, indique comment la valeur de cette métrique est calculée.

$$Fitness_P(L) = 1 - \frac{f_{coût}(P, J)}{Coût\_maximum} \tag{3.25}$$

La fonction $f_{coût}$(P,L) correspond au coût optimal d'alignement entre J et P. Le dénominateur de la formule, $Coût\_maximum$, représente la valeur maximum d'alignements possibles, c'est-à-dire lorsque aucun élément du modèle P et du journal d'événement J n'est synchronisé. Dans le reste de cette sous-section, nous formalisons le mécanisme de détection qui se base sur l'utilisation de la fitness pour distinguer les déviations par rapport aux modèles de comportement. Nous allons donc considérer k' sous-ensembles de données $P'_{l'}$, provenant d'un jeu de données à tester, et k modèles de comportement M_l, construits pendant la phase d'apprentissage sous la forme de réseaux de Petri. Ainsi, quel que soit l dans l'intervalle $[\![1, k]\!]$, M_l comporte un sous-ensemble de places, notées PL_l, et de transitions, notées TR_l, qui contiennent respectivement o_l et p_l éléments.

$$PL_l = \{pl_1, \ldots, pl_{o_l}\} \tag{3.26}$$
$$TR_l = \{tr_1, \ldots, tr_{p_l}\} \tag{3.27}$$

Notre approche consiste à trouver le modèle qui correspond le mieux à chacun des sous-ensembles de données $P'_{l'}$, c'est-à-dire le modèle donnant la fitness la plus élevée, dont le calcul est donné par la formule 3.28. Dans le cas où aucun modèle ne se rapproche assez des données à tester, c'est-à-dire que la valeur de la *fitness* ne dépasse pas un seuil fixé, alors nous considérons que le comportement du système observé est anormal.

$$\forall l' \in [\![1, k']\!], Fitness_{l'} = \mathbf{max}_{l \in [\![1,k]\!]} Fitness_{M_l}(P'_{l'}) \tag{3.28}$$

Il nous reste à distinguer les éléments anormaux du jeu de données. En effet, jusqu'à présent, nous avons associé chaque sous-ensemble de données au modèle qui les décrit le mieux. De plus, nous avons accès à la valeur de la fitness qui nous permet d'avoir une idée de la proportion de similarité entre ces deux entités. Les résultats retournés sont décrits par l'équation 3.29.

$$\forall l' \in [\![1, k']\!], \exists l \in [\![1, k]\!], RES_{l'} = (P_{l'}, M_l, Fitness_{l'}) \tag{3.29}$$

Une fois ces résultats $RES_{l'}$ disponibles, nous définissons un ensemble q comportant différents seuils de fitness SF_r. Ces seuils permettent de distinguer deux sous-ensembles de résultats $NORMAL_r$ et $ANORMAL_r$, définis par les équations 3.30 et 3.31. Dans ce premier sous-ensemble, nous retrouvons l'ensemble des résultats $RES_{l'}$ dont la fitness est supérieure à la valeur seuil SF_r. Nous trouvons donc dans ce sous-ensemble les éléments de notre jeu de données à tester qui sont suffisamment proches de l'un des modèles pour être considérés comme normaux. Inversement, dans le second sous-ensemble $ANORMAL_r$, nous retrouvons les éléments du jeu de données pour lesquels le calcul de fitness avec le meilleur modèle nous donne une valeur inférieure à notre valeur de seuil. La présence d'un seul élément dans le sous-ensemble $ANORMAL_r$ est suffisante pour mettre en lumière un dysfonctionnement au niveau du comportement du système IoT surveillé. En effet, si la fitness du meilleur modèle est en dessous du seuil considéré, alors les données ne correspondent à aucun modèle de comportement normal et nous avons affaire à une anomalie.

$$\forall l' \in [\![1, k']\!], \forall r \in [\![0, q]\!] \; avec \; q \in \mathbb{N}^*,$$
$$SF_r = \frac{r}{q}$$
$$NORMAL_r = \{RES_{l'} \mid Fitness_{l'} > SF_r\}, \tag{3.30}$$
$$ANORMAL_r = \{RES_{l'} \mid Fitness_{l'} \leq SF_r\} \tag{3.31}$$

Ce mécanisme de détection est compatible avec un ensemble varié de protocoles et plates-formes, qui composent les systèmes IoT, et donc est exploitable dans des scénarios d'attaques impliquant de multiples protocoles et plates-formes.

3.5 Synthèse

Dans ce chapitre, nous proposons une approche de détection pour les systèmes IoT hétérogènes, qui repose sur le couplage de méthodes de pré-traitement à des techniques de *process mining*. Ceci permet de tirer profit du *process mining*, tout en évitant une explosion des états. Nous avons présenté l'architecture de cette solution, ses différents blocs et leurs interactions, et avons détaillé deux phases importantes : la phase d'apprentissage et la phase de détection. La première phase est composée de quatre étapes permettant la génération des modèles de comportement à l'aide d'un algorithme de *process mining*. Après une normalisation des données, nous utilisons une méthode de partitionnement qui sert à extraire les principaux états pris par les systèmes IoT. Un découpage des données intervient ensuite pour que les modèles ne soient pas trop spécifiques, et un algorithme de *process mining* génère enfin les modèles de comportement sous la forme de réseaux de Petri. La seconde phase s'appuie également sur quatre étapes qui permettent de détecter les anomalies dans un jeu de données à partir des modèles construits. Lors de la normalisation, nous réutilisons les paramètres issus de la phase d'apprentissage, tandis que la méthode de partitionnement est remplacée par une méthode de regroupement par distance euclidienne. La détection s'appuie sur une mesure de la déviation des données au regard des modèles disponibles. Dans le chapitre suivant, nous illustrons les résultats obtenus avec plusieurs jeux de données provenant de différents domaines d'applications IoT. Nous utilisons l'algorithme de l'*inductive miner* pour quantifier les performances de la solution et les comparer à celles d'autres méthodes de détection couramment utilisées pour la détection d'anomalies.

Chapitre 4

Évaluation de l'approche de détection utilisant le *process mining*

Sommaire

4.1 Introduction 53
4.2 Description des jeux de données 54
4.2.1 Description qualitative des jeux de données 54
4.2.2 Véhicules connectés 55
4.2.3 Industrie 4.0 56
4.2.4 Robots d'assistance 58
4.3 Performance du *process mining* pour la détection d'attaque 59
4.3.1 Impact des paramètres de configuration sur la détection 59
4.3.2 Évaluation de la phase de détection 62
4.4 Comparaison à d'autres méthodes de détection 70
4.4.1 Description des méthodes 71
4.4.2 Comparaison des performances de détection 73
4.4.3 Conséquences de la présence de perturbations sur la détection 78
4.5 Synthèse 80

4.1 Introduction

Dans ce chapitre nous évaluons la solution proposée précédemment. Ainsi, dans un premier temps, nous décrivons, tout d'abord, les jeux de données à notre disposition.

Nous revenons ensuite sur la labélisation des données, qui nous a permis d'évaluer les performances de notre méthode, et sur les métriques que nous avons retenues pour caractériser les performances de détection. Après cela, nous étudions l'influence des paramètres de notre méthode, en particulier au niveau de l'algorithme de partitionnement ainsi que du temps de découpe. Cette étude nous permet de déterminer les valeurs à utiliser pour optimiser la détection. Afin de valider ce choix et de montrer que nous évitons le sur-apprentissage, nous utilisons un nouveau jeu de données qui n'est intervenu ni dans l'apprentissage ni au moment de choisir les paramètres de la détection. Finalement, nous comparons les résultats de détection de notre méthode à ceux

obtenus par d'autres approches couramment utilisées dans la détection d'anomalies. Nous analysons également les conséquences sur la détection de la présence de perturbations, comme le bruitage des données remontées par les capteurs ou bien encore la disparition de celles-ci.

4.2 Description des jeux de données

Nos expériences reposent sur différents jeux de données applicatives qui nous ont été fournis dans le cadre du projet européen SecureIoT. Ils proviennent de trois domaines d'application différents. Nous retrouvons donc des expérimentations sur des données issues de véhicules connectés, de l'industrie 4.0 et du domaine médical provenant de robot d'assistance pour les enfants autistes. Pour chacun de ces trois cas d'usage, nous avons accès à plusieurs jeux de données. Par exemple, dans le cas des véhicules connectés, nous avons des données de conduite dans différentes villes en Europe. Dans un premier temps, nous faisons une description qualitative générale sur les jeux de données. Puis, nous décrivons les jeux de données ainsi que les scénarios d'attaques à notre disposition.

4.2.1 Description qualitative des jeux de données

Il est nécessaire de pouvoir discerner les éléments des jeux de données devant être normalisés et partitionnés. Ainsi, nous devons trier les différents éléments à notre disposition selon deux critères principaux que sont le type des données et leur variabilité. En effet, les paramètres ayant des valeurs non bornées peuvent être la source d'un certain nombre de difficultés lors de la normalisation. Ainsi, la normalisation sur des données non bornées dépend fortement des valeurs rencontrées dans le jeu de données et peut grandement différer lors du traitement de nouvelles données. De plus, nous devons faire la distinction entre les paramètres dont les valeurs sont discrètes et celles qui sont continues. En effet, certains algorithmes de partitionnement pourraient poser certaines difficultés pour trouver des groupes pertinents en utilisant exclusivement des données continues comme par exemple l'algorithme DBSCAN qui repose sur la densité. Finalement, certaines des données, textuelles ou catégorielles ne peuvent pas être directement utilisées par la plupart des algorithmes de partitionnement et nécessitent un traitement particulier. Dans notre cas, la phase de partitionnement est réalisée en deux étapes. D'abord, elle cherche les groupes représentant au mieux les données aux valeurs numériques. Puis, elle intègre les données catégorielles et booléennes. Ces deux étapes nous permettent de définir les états possibles de notre système.

Comme nous l'avons mentionné précédemment, la variabilité des valeurs prises par chaque paramètre du jeu de données est également un point important. Il serait effectivement possible de réduire la consommation de certains capteurs en réduisant la fréquence de collecte des données pour les paramètres ne changeant que peu fréquemment. En ce qui concerne la variabilité, il est intéressant de savoir si les valeurs d'un paramètre ne changent jamais et conservent une valeur fixe (VF), si elles changent à la suite d'un événement particulier (CE) ou, enfin, si elles changent continuellement (CC).

Ainsi, à partir de ces différentes considérations, nous avons séparé les différents paramètres des jeux de données en cinq groupes qui sont listés ci-dessous :

- Données numériques
  - Valeurs discrètes (VD)
    - Bornées (B)
    - Non bornées
  - Valeurs numériques continues (VN)
    - Bornées (B)
    - Non bornées
- Données booléennes (VB)
- Données textuelles (VT)
- Données catégorielles (VC)
- Agrégation ou regroupement de paramètres (RP)

4.2.2 Véhicules connectés

Les derniers jeux de données que nous avons traités ont été fournis par le partenaire IDIADA. Ils proviennent d'un outil de simulation développé par cette entreprise et qui permet de générer des données de conduite.

Données. Dans un premier temps, nous avons pu avoir à notre disposition les données du bus CAN (Controller Area Network) du véhicule ainsi que les données retournées par le module V2X (Vehicle-to-Everything), qui nous donne accès aux informations mécaniques de la voiture. La plupart de ces informations sont représentées par des valeurs discrètes (VD) ou numériques (VN) qui évoluent continuellement (CC). C'est le cas notamment du régime moteur, de l'angle du volant, de la vitesse et de l'utilisation des pédales de freinage et d'accélération. Toutefois, nous observons des éléments tels que la vitesse enclenchée et l'état moteur qui sont illustrés par des valeurs catégorielles (VC) évoluant à la rencontre d'événements (CE). Nous remarquons que certains paramètres comme, par exemple, la longueur du véhicule reste identique (VF) tout au long des jeux de données. Une description plus complète de ces deux sources de données est disponible dans les tableaux 4.1 et 4.2. Chacun de ces jeux de données contient entre 2000 et 5000 mille points de données de conduite et entre 500 et 1500 données du bus CAN, avec notamment la charge en pourcentage de celui-ci. La différence de taille entre les différents jeux de données s'explique par le choix à la fois de la ville et du trajet retenu lors de la simulation.

	VD	VN	VB	VC	VT	RP	VF	CC	CE
Vitesse enclenchée				X					X
État moteur				X					X
Rotation moteur	X							X	
Angle du volant		X(B)						X	
Vitesse		X						X	
Frein		X(B)						X	
Accélérateur		X(B)						X	
Coordonnées GPS						X		X	
Longueur du véhicule		X					X		
Largeur du véhicule		X					X		
Profil utilisateur						X	X		

TABLE 4.1 – Tableau explicatif des données relatives aux informations de conduite du véhicule

	VD	VN	VB	VC	VT	RP	VF	CC	CE
Charge du bus CAN	X(B)							X	
Séquence invalide	X							X	
Message inattendu	X							X	

TABLE 4.2 – Tableau explicatif des données relatives au système CAN du véhicule

Scénario d'attaque. Dans le cas des véhicules connectés, nous avons cherché à détecter une conduite qui apparaîtrait comme incohérente. Par exemple, il ne semble pas normal qu'un véhicule possède une vitesse d'environ 100 kilomètres par heure avec un volant complètement tourné à droite. Ces anomalies ont été directement introduites par notre partenaire dans les jeux de données où des valeurs d'attributs ont été modifiées. Ainsi, l'objectif principal est de détecter le comportement anormal que peut avoir un véhicule pour faciliter la prise de contre-mesures comme donner l'ordre au véhicule de se garer pour éviter qu'il ne devienne un danger pour les autres automobilistes.

4.2.3 Industrie 4.0

Les jeux de données de l'industrie 4.0 proviennent d'une simulation d'un processus d'injection plastique industriel. Le système permet la production de pièces en plastique par injection.

Données. Les jeux de données à notre disposition contiennent environ 18 000 mille points, ce qui correspond, environ, à un jour de simulation. Nous avons ici accès à des données telles que la température à plusieurs points-clés de l'appareil et la pression des différents pistons intervenant dans le processus. Les températures et les pressions observées sont décrites par des valeurs numériques (VN) et changent continuellement (CC). Le fonctionnement du chauffage ainsi que l'utilisation des différentes valves sont caractérisés par des valeurs booléennes (VB) et sont amenés à changer à la suite d'un événement (CE). Une description de ce jeu de données est faite dans le tableau 4.3. La création d'une pièce à l'aide du système à injection se déroule de la façon suivante : le module de remplissage, qui doit contenir suffisamment de matériaux à fondre, va chauffer son contenu pour ensuite l'injecter dans une cavité qui fera office de moule. Une fois le moule rempli, la pièce est refroidie sous pression afin d'accélérer sa solidification. Finalement, la pièce est extraite du moule par le piston.

	VD	VN	VB	VC	VT	RP	VF	CC	CE
Température de la trémie		X						X	
Température du baril		X						X	
Température de la machine		X						X	
Température du moule		X						X	
Chauffage			X						X
Pression dans le baril		X						X	
Pression dans le moule		X						X	
Valve de remplissage			X						X
Valve d'entrée du moule			X						X
Valve de sortie du moule			X						X

TABLE 4.3 – Tableau explicatif des données relatives au système d'injection plastique (Industrie 4.0)

Scénario d'attaque. Dans le cas de l'industrie 4.0, nous avons cherché à détecter les anomalies dans les jeux de données d'une machine à injection plastique. L'idée est ici de repérer ces déviations au plus tôt pour empêcher que celles-ci ne se transmettent aux autres machines. Ainsi, nous souhaitons éviter à ces machines de subir des dégâts ce qui pourrait mettre en danger les agents travaillant à proximité. En effet, augmenter significativement la pression dans le baril ou dans le moule peut entraîner une destruction du système industriel. De la même façon, provoquer une chute de la température que ce soit dans la trémie, le baril, la machine ou bien encore le moule au mauvais moment du cycle peut être également dangereux. Dans cette configuration la machine va être amenée à travailler avec du plastique solide là où il devrait être liquide, ce qui peut provoquer sa casse et poser de problèmes de sécurité aux agents proches de la machine.

4.2.4 Robots d'assistance

Les données fournies par le partenaire LuxAI viennent d'un test de démonstration effectué par leur personnel sur un robot d'assistance. Ce robot est conçu pour stimuler la mémoire et permettre aux personnes âgées de faire un minimum d'activité physique. De plus, ce type de robot, en captant plus facilement l'attention des enfants autistes, permet également de leur apprendre les bases de l'interaction sociale.

Données. Nous avons à notre disposition cinq jeux de données, de plus de 12 000 points chacun sur une durée d'environ 10 minutes, qui nous donnent les positions et les angles de plusieurs joints du robot comme le montre le tableau 4.5. Ainsi, on trouve les angles du Tangage et du Lacet de la tête, du Roulis et du Tangage des épaules ainsi que ceux du Roulis et du Tangage des coudes. Ces attributs présentent des valeurs numériques bornées (VN) et changent continuellement (CC).

(angle en degré)	VD	VN	VB	VC	VT	RP	VF	CC	CE
Tangage de la tête		X(B)						X	
Lacet de la tête		X(B)						X	
Roulis de l'épaule g&d		X(B)						X	
Tangage de l'épaule g&d		X(B)						X	
Roulis du coude g&d		X(B)						X	
Tangage du coude g&d		X(B)						X	

TABLE 4.4 – Tableau explicatif des données relatives aux moteurs du robot d'assistance

Nous avons également accès à trois autres jeux de données, constitués d'environ 600 cents enregistrements pour une durée proche de 6 minutes, qui donnent un récapitulatif des différents points d'accès wifi visibles par le robot comme le montre le tableau 4.4. Nous pouvons donc avoir accès au routeur sur lequel est connecté le robot ainsi qu'un certain nombre de caractéristiques sur les routeurs visibles telles que leurs noms, qui sont représentés par une valeur textuelle (VT) qui est amenée à changer continuellement (CC). Nous avons également accès à la force du signal wifi en pourcentage, qui est une valeur discrète bornéee (VD) changeant continuellement (CC). Nous pouvons également voir la valeur de la fréquence utilisée par le routeur, qui est exprimée par une valeur discrète (VD) en Hertz restant fixe pendant la durée du jeu de données. Finalement, nous pouvons connaître la position géographique des routeurs (*localisation*) à l'aide de l'attribut regroupant la latitude et la longitude (RP) et qui, là encore, reste identique tout au long de nos jeux de données. Ces derniers jeux de données ont été générés à partir d'un scénario d'attaque s'effectuant en plusieurs étapes.

	VD	VN	VB	VC	VT	RP	VF	CC	CE
Routeur connecté					X				X
Routeur visible (nom)					X			X	
Routeur visible (force en %)	X(B)							X	
Routeur visible (fréquence)	X						X		
Localisation						X	X		

TABLE 4.5 – Tableau explicatif des données relatives aux routeurs wifi visibles par le robot d'assistance

Scénario d'attaque. Ce type de robot pourrait être un vecteur de vulnérabilité de la plateforme médicale auquel il est connecté. Dans le scénario d'attaque, que nous avons mis conjointement en place avec l'industriel, nous avons retenu une attaque de l'homme au milieu. Ainsi, l'attaquant, en utilisant un routeur wifi de substitution et en surchargeant celui existant, va forcer le robot à se connecter au routeur sous son contrôle. Une fois la connexion établie, l'attaquant va pouvoir à la fois transmettre des consignes au robot et recevoir les différentes informations envoyées par celui-ci, comme le flux vidéo.

4.3 Performance du *process mining* pour la détection d'attaque

Dans cette section, nous étudions l'impact du paramétrage sur les performances de détection de la méthode proposée lors de la phase d'apprentissage. Puis, nous en montrons la pertinence sur la détection des anomalies en appliquant les paramètres donnant les meilleurs résultats sur un nouveau jeu de données du même système. Nous présentons l'un des modèles que nous avons obtenus pour chacun des domaines d'application décrits précédemment. De plus, nous montrons également comment nous avons évalué la pertinence de notre méthode dans la détection des anomalies contenues dans les jeux de données testés.

4.3.1 Impact des paramètres de configuration sur la détection

Nous détaillons, dans un premier temps, l'influence de la découpe temporelle des données sur les performances de détection. Puis, nous étudions de quelle façon le paramétrage des algorithmes de partitionnement permet d'obtenir les meilleurs résultats. Lors de nos expériences, la méthode de normalisation min-max a toujours obtenu des performances supérieures à la normalisation utilisant la cote standard. Ainsi, les résultats obtenus dans cette section, ont tous été obtenus avec la normalisation min-max.

Découpe temporelle. Dans une première série d'expériences, nous voulons quantifier l'influence de la découpe temporelle sur les performances de détection de notre solution. Cette découpe intervient durant le pré-traitement des données. Ainsi, l'ensemble des données est découpé en sous-ensembles. Leurs tailles sont définies par un unique intervalle de temps maximal paramétrable, ce qui permet de réduire la complexité des modèles de comportement générés et de localiser plus précisément les parties des données comportant des anomalies. En effet, notre

	véhicule connectés	machine à injection	robot d'assistance
BIRCH	0,824	0,5	0,818
K-MEANS	0,979	0	0,848
DBSCAN	0,694	1	0,5

TABLE 4.6 – Meilleur résultat expérimental pour chacun des jeux de données

métrique de détection est calculée pour chaque sous-ensemble de données et donc réduire leur taille permet d'être plus précis pour situer les anomalies dans le jeu de données. La figure 4.1 montre les courbes ROC (Receiver Operating Characteristic), obtenues pour les trois jeux de données décrits précédemment, lorsque nous avons fait varier le paramètre de découpe temporelle. Les sous-figures (a), (b) et (c) correspondent respectivement aux jeux de données des véhicules connectés, de la machine à injection et des robots d'assistance. Les courbes ROC permettent d'évaluer la pertinence d'une méthode de détection devant faire la distinction entre deux ensembles d'éléments (les données normales et anormales dans notre cas). Elle montre l'évolution de taux de vrais positifs en fonction du taux de faux positifs. Elles sont obtenues en faisant varier la valeur de seuil de détection entre 0% et 100%. Le calcul de l'aire sous la courbe permet de quantifier les performances de détection pour une valeur donnée du paramètre de découpe temporelle. Les courbes qui possèdent les aires les plus grandes présentent de meilleures performances de détection. Nous obtenons les différents points des courbes ROC en faisant varier le seuil de *fitness* (SF) entre 0 et 1 par pas de 0,05. Pour chaque jeu de données, nous pouvons déduire le temps de découpe optimal en faisant varier le paramètre et en cherchant la courbe ROC ayant l'aire la plus importante. Nous avons fait varier le paramètre de découpe sur un intervalle. Cet intervalle a été choisi de telle sorte à contenir les deux cas extrêmes possibles pour nos modèles. Le premier, qui correspond à la valeur basse de l'intervalle, permet de générer des modèles comportant un unique point de données. La limite haute de l'intervalle correspond au second cas extrême où un unique modèle est généré pour l'ensemble du jeu de données. Ainsi, avec les données des véhicules connectés, nous obtenons, dans la meilleure configuration, une aire de 0,923 qui correspond à une découpe temporelle de 10 secondes. Toutefois, nous observons une dégradation des performances lorsque l'on augmente ou diminue le paramètre de découpe. En effet, dans les cas où l'intervalle de temps choisi pour la découpe est trop petit, les modèles de comportement générés ne permettent pas de prendre suffisamment en compte l'historique et se rapprochent donc des performances de l'algorithme de partitionnement seul. Si la taille des sous-ensembles de données est trop importante, nous nous retrouvons face à un problème de sur-apprentissage qui entraîne une dégradation des performances. De plus, le temps de traitement est également accrue à cause de la complexité des modèles.

Ainsi nous observons une dégradation des performances lorsque le paramètre de découpe dépasse les valeurs représentées sur la la légende de la figure 4.1. Durant la phase d'apprentissage, le temps de traitement décroît quand le paramètre de découpe augmente car moins de modèles sont générés. Pour la phase de détection, le temps de traitement est plus important que celui de la phase d'apprentissage. Il est maximal lorsque les données sont représentées par un seul modèle ou que chaque point de données possède son propre modèle et plus faible pour des valeurs intermédiaires.

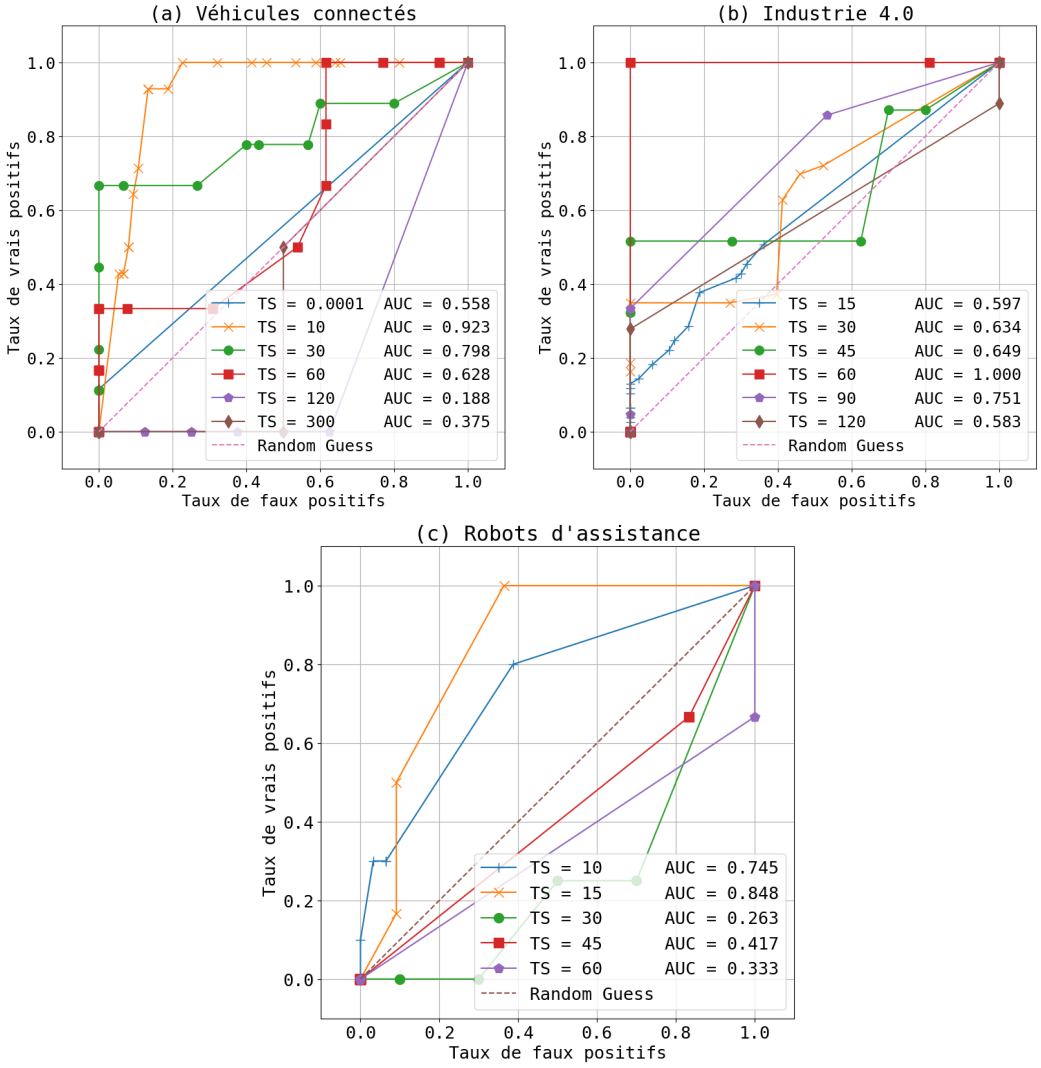


FIGURE 4.1 – Étude de l'influence du temps de découpe sur les performances de détection à l'aide des courbes ROC pour les véhicules connectés (a), l'industrie 4.0 (b) et les robots d'assistance (c)

Phase de pré-traitement. Nous utilisons trois techniques de partitionnement et deux de normalisation pour lesquelles nous avons fait varier les paramètres pour obtenir les meilleures performances atteignables pour les jeux de données étudiés. Pour la normalisation, nous avons utilisé les techniques min-max et cote standard, tandis que pour le partitionnement, nous avons appliqué les algorithmes K-MEANS, BIRCH et DBSCAN. Afin de maintenir la cohérence entre la phase d'apprentissage et de détection, nous avons conservé les mêmes méthodes de normalisation et de partitionnement entre ces deux phases lors de notre évaluation. Le fait de modifier les paramètres des algorithmes de partitionnement n'a pas d'impact significatif sur le temps de traitement de la phase d'apprentissage qui reste, dans tous les cas, proche de 30 secondes. En revanche, cette modification influence, là encore, le temps de traitement de la phase de détection. Le tableau 4.6 présente les meilleurs résultats obtenus pour chacun des algorithmes de partitionnement pour les trois cas d'usage. Il indique la valeur de l'aire sous la courbe des courbes ROC obtenues. Nous remarquons que les meilleurs résultats pour les jeux de données des véhicules et des robots sont obtenus en utilisant la méthode de partitionnement K-MEANS. Pour la machine à injection, l'utilisation de l'algorithme de partitionnement reposant sur la densité (DBSCAN) apparaît le plus adapté car c'est celui qui obtient les meilleurs résultats de détection.

Pour les tests que nous avons effectués avec K-MEANS, nous avons fait varier la paramètre K, qui correspond au nombre de groupes que l'on souhaite obtenir en sortie de l'algorithme et qui donc influence directement le nombre d'états du modèle du système créé par le *process mining*. Pour DBSCAN, nous avons fait varier ϵ, qui correspond au rayon maximum que l'algorithme va considérer dans sa recherche de voisins du point de données. Finalement, pour BIRCH, nous avons fait varier le seuil, qui correspond au rayon maximum des groupes. Le fait d'augmenter K pour K-MEANS, de diminuer ϵ pour DBSCAN ou de diminuer le seuil pour BIRCH permet d'améliorer la précision des modèles de comportement et donc d'améliorer la détection d'anomalies. En revanche, on peut observer une valeur-limite, pour chacun des jeux de données, au-delà de laquelle les performances de détection se dégradent. En effet, l'utilisation d'un trop grand nombre d'états pour le système va générer des modèles de comportement trop spécifiques qui considéreront le moindre bruit comme une anomalie. Ainsi, avec $K = 2500$, le *process mining* crée un modèle pour chacun des points du jeu de données des véhicules connectés. Les résultats obtenus sont peu satisfaisants comme nous pouvons le voir sur la sous-figure (a) de la figure 4.2, qui nous montre une aire sous la courbe ROC proche de 0,5, ce qui correspond aux performances de détection du choix aléatoire. C'est pour éviter cette explosion du nombre d'états que nous utilisons les algorithmes de partitionnement avant le *process mining*. Les sous-figures (a), (b) et (c) présentent l'évolution de l'aire sous la courbe ROC pour différentes valeurs de paramètres de l'algorithme de partitionnement qui ont donné respectivement les meilleurs résultats de détection pour chacun des cas. Ainsi, pour les véhicules connectés, nous considérons dans un premier temps l'utilisation de l'algorithme de partitionnement K-MEANS avec $K = 1000$. En ce qui concerne l'industrie 4.0, qui correspond au système d'injection plastique, nous allons appliquer l'algorithme DBSCAN avec une valeur pour le paramètre ϵ de 0,4. Finalement, nous utilisons K-MEANS avec $K = 10$ pour les robots d'assistance. Comme nous l'avons mentionné dans le cas des véhicules connectés, nous allons devoir valider la pertinence de ces paramétrages dans une phase d'évaluation.

4.3.2 Évaluation de la phase de détection

Dans cette section, nous validons la pertinence du choix des paramètres sur la détection des anomalies sur un nouveau jeu de données du même système. Pour ce faire, nous appliquons les paramètres ayant donné les meilleurs résultats, trouvés précédemment à partir des deux jeux de données initiaux, ayant donné les meilleurs résultats lors de la phase d'apprentissage.

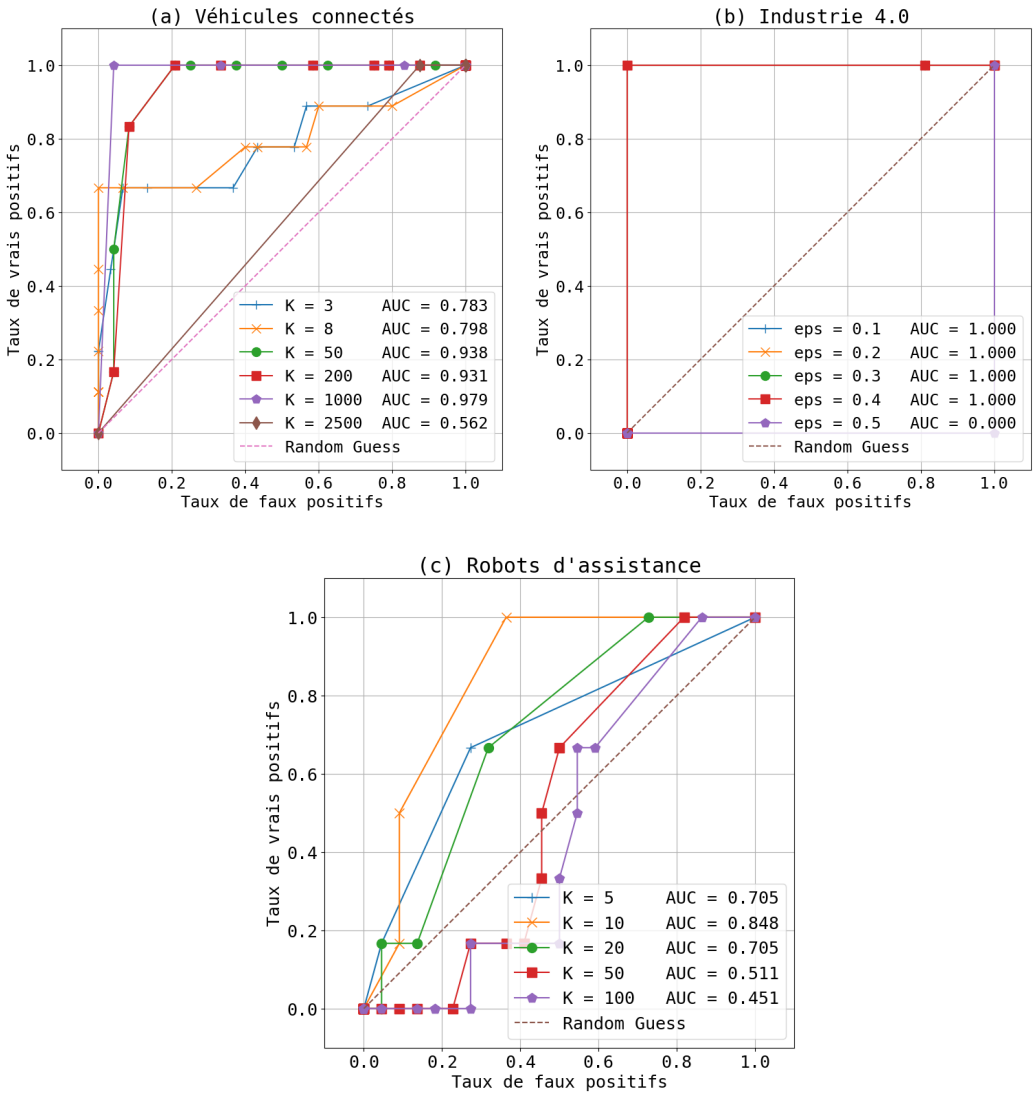


FIGURE 4.2 – Étude de l'influence des paramètres de partitionnement sur les performances de détection à l'aide des courbes ROC pour les véhicules connectés (a), l'industrie 4.0 (b) et les robots d'assistance (c)

Pour les véhicules connectés, chacun des jeux de données provient d'une simulation de conduite dans des villes différentes. Dans le cas de l'industrie 4.0 et de la machine à injection plastique, nous avons découpé le jeu de données initial d'un jour en plusieurs sous-ensembles d'une heure, ce qui correspond à environ soixante cycles de moulage. En ce qui concerne le robot d'assistance, les données ont été collectées lors différentes sessions. De cette façon, nous nous sommes assurés que les jeux de données utilisés lors du choix du paramétrage et celui de la validation étaient différents. Nous présentons un exemple de modèle de comportement pour chacun des domaines d'application et montrons la façon dont nous avons évalué la pertinence de notre méthode en détectant les anomalies contenues dans les jeux de données testés.

	Véhicule connecté	Machine à injection	Robot d'assistance
Fréquence des données	2 Hz	2 Hz	20 Hz
Construction des modèles (nb d'éléments)	3200	9000	12000
Évaluation (nb d'éléments)	2400	7000	10000
Ratio d'anomalies	20%	30%	10%

TABLE 4.7 – Description des caractéristiques des jeux de données utilisés dans l'évaluation des performances

Le tableau 4.7 nous donne les caractéristiques moyennes des jeux de données utilisées pour chacun des cas d'usage. Nous y trouvons la fréquence de génération des points de données, le nombre d'éléments contenus dans les jeux de données pour les phases d'apprentissage et de détection, ainsi que le ratio d'anomalies inclues dans le jeu de données d'évaluation.

Véhicule connecté. Nous construisons des modèles de comportement normal pour les véhicules à partir des données fournies par notre partenaire et cherchons à détecter des déviations par rapport à ceux-ci. Nous avons également appliqué les paramètres ayant donné les meilleurs résultats lors de la phase d'apprentissage, c'est-à-dire la normalisation min-max accompagnée de la méthode de partitionnement K-MEANS (K=1000) et un temps de découpage des données de 10 secondes. Afin de tester la pertinence du choix de ces paramètres, nous avons utilisé un nouveau jeu de données pour la détection. À partir de cette sélection, nous obtenons de l'ensemble du jeu de données d'apprentissage un certain nombre de modèles qui sont conservés dans notre architecture, dont celui donné par la figure 4.3. Ce modèle correspond au démarrage du véhicule. On retrouve donc l'insertion de la clé, puis le démarrage du moteur ainsi que le début de l'accélération afin d'atteindre une vitesse lente.

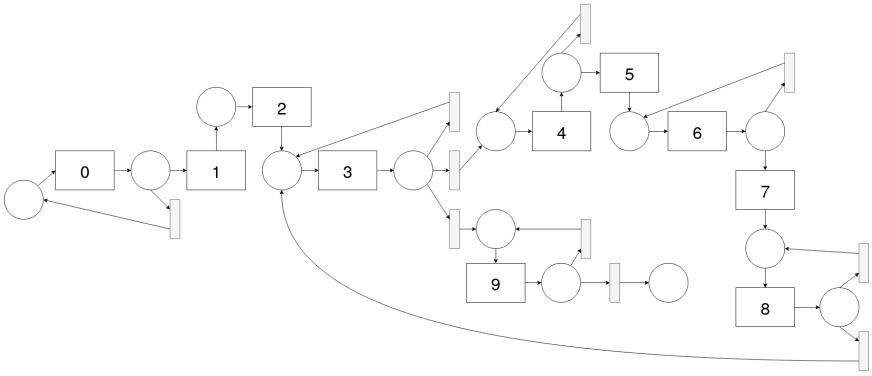
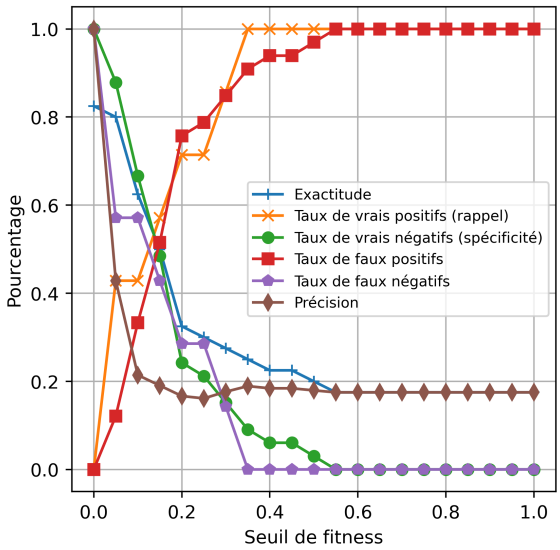


FIGURE 4.3 – Modèle de comportement au démarrage du véhicule

Dans notre cas, les données du nouveau jeu de données sont labélisées, il est donc possible de tester la pertinence de notre système de détection d'anomalies. On retrouve, sur la partie gauche de la figure 4.4, un graphe présentant les courbes des différentes métriques de détection ainsi que la courbe ROC correspondante sur la partie droite. La figure représentant les métriques montre une détérioration rapide de la précision et de l'exactitude qui chute à moins de 35% pour un seuil de 0,2. En ce qui concerne la courbe ROC, elle affiche une aire sous la courbe de 0,58, ce qui est très proche en termes de performances du choix aléatoire. La valeur de seuil optimale pour obtenir les meilleurs résultats de détection est 0,05, ce qui permet de détecter un peu plus de 40% des anomalies avec un peu plus de 10% de faux positifs.

(a) Valeurs des métriques



(b) Courbe ROC

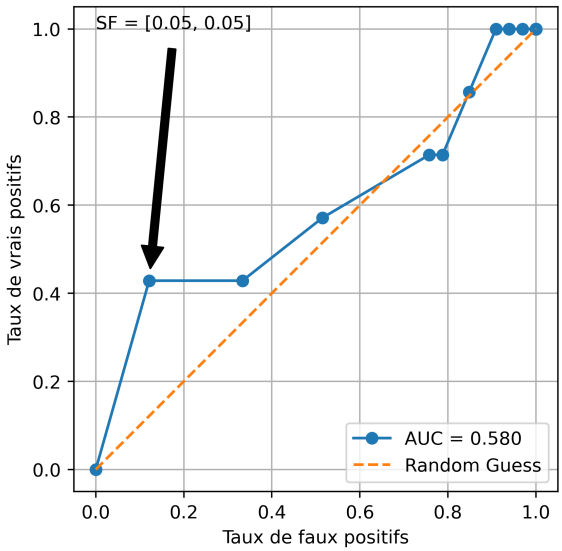


FIGURE 4.4 – Détection des anomalies sur le véhicule connecté dans un cas de sur-apprentissage

On remarque que dans le cas présent, nous avons affaire à un sur-apprentissage. En effet, nous avons pu obtenir des résultats très intéressants avec des courbes ROC possédant une aire sous la courbe de 0,923, comme nous avons pu le voir dans la figure 4.1, ce qui se rapproche d'une détection parfaite. En revanche, nous ne retrouvons pas ces résultats lorsque nous appliquons les mêmes paramètres à un autre jeu de données. Nous avons donc fait le choix d'utiliser d'autres

paramètres qui ont donné de moins bons résultats en phase d'apprentissage mais qui évitent ce phénomène de sur-apprentissage. Ainsi, nous avons appliqué une fois de plus la phase de détection, en utilisant cette fois-ci un temps de découpage de 30 secondes et un plus faible nombre de groupes avec K-MEANS (K=8), sur le même jeu de données que pour les figures précédentes. Nous avons considéré que les éléments du jeu de données possédant des valeurs numériques continues (VN) pouvaient représenter 8 états possibles, d'où K=8. Le premier d'entre eux traduit l'immobilité du véhicule, les deux suivants correspondent à une conduite lente en ligne droite où seul la rotation moteur, faible ou élevée, permet de les distinguer. On se trouve dans la même configuration pour le cas d'une conduite à vitesse intermédiaire, tandis que pour une vitesse rapide on ne considère que le cas d'une vitesse de rotation moteur importante. Il nous reste à prendre un compte un état qui décrit le fait que le véhicule tourne à droite et un autre pour le côté gauche pour obtenir nos 8 états différents. Nous nous sommes également demandé si l'utilisation d'un temps de découpe de 10 secondes était le plus pertinent. En effet, l'utilisation du *process mining* va nous permettre de construire des réseaux de Petri qui vont décrire le comportement du système étudié. L'utilisation d'un temps de découpe trop faible va grandement limiter la pertinence de ces modèles. Dans un cas extrême, ces derniers ne permettront plus de prendre en compte l'historique des données car il ne resterait plus qu'un état par modèle. C'est pourquoi nous avons décidé d'utiliser un temps de découpe de 30 secondes plutôt que de 10 secondes.

(a) Valeurs des métriques

(b) Courbe ROC

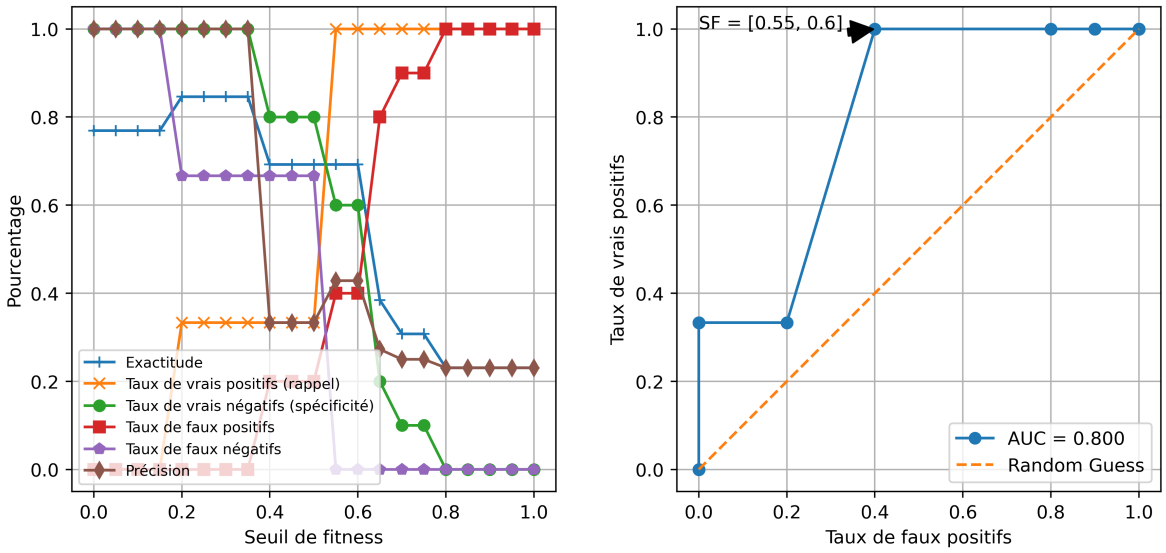


FIGURE 4.5 – Détection d'anomalies sur le véhicule connecté

La sous-figure (a) présente dans 4.5 donne l'évolution d'un ensemble de métriques d'évaluation, tandis que (b) représente la courbe ROC associée. Une fois de plus, la courbe ROC nous permet de choisir la valeur du seuil optimale à utiliser. Nos résultats ont été obtenus en utilisant une découpe du jeu de données de 30 secondes et en ayant recours à la méthode de normalisation min-max ainsi qu'à l'algorithme de partitionnement K-MEANS (K=8). Nous obtenons une aire sous la courbe ROC de 0,8 ce qui nous permet de valider ce choix de paramètres pour les véhicules connectés.

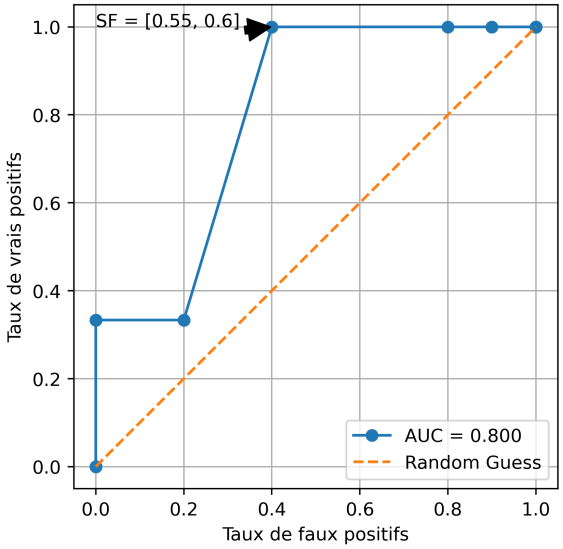


FIGURE 4.6 – Courbe ROC obtenue pour le véhicule connecté

Industrie 4.0. Nous avons également conduit des expériences sur les jeux de données de l'industrie 4.0. Ces données décrites par le tableau 4.3 sont tirées d'une simulation d'un processus d'injection plastique et nous permettent d'accéder aux informations relatives à la température et à la pression des pistons dans divers endroits du système industriel. La figure 4.7 rend compte de l'intégralité d'un cycle du processus d'injection qui va du début de l'injection du plastique jusqu'à la libération de la pièce à l'aide du piston de sortie.

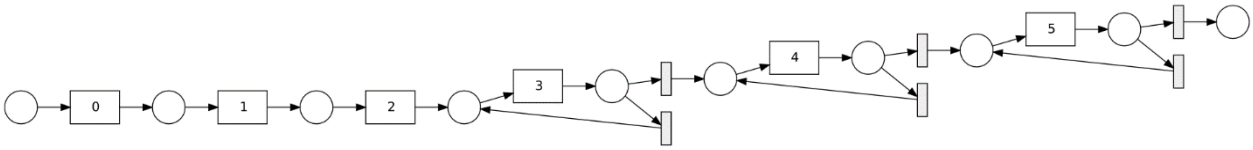
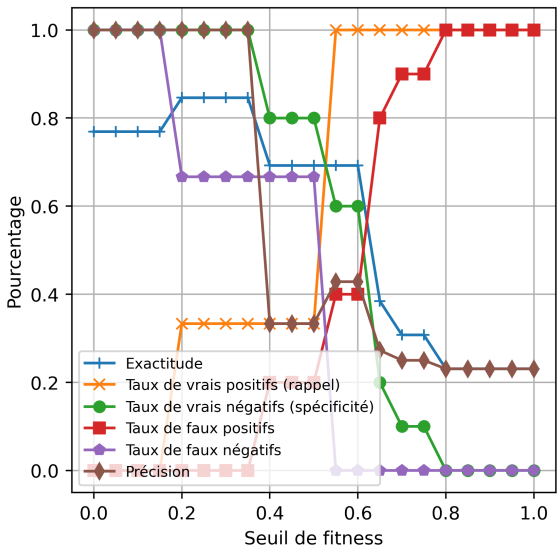


FIGURE 4.7 – Modèle d'un cycle du processus d'injection

Ici, environ 30% des données ont été remplacées par des anomalies dans les jeux de données, ce qui nous a permis de tester notre solution de détection. Nous avons découpé les jeux de données en sous-ensembles de soixante secondes après avoir normalisé les données avec la méthode min-max et les avoir partitionnés avec DBSCAN. Là encore, nous avons utilisé les paramètres de la phase d'apprentissage qui donnent les meilleurs résultats de détection. La partie gauche de la figure 4.8 représente différentes métriques utilisées dans la détection d'anomalies telles que les vrais et faux positifs / négatifs ainsi que l'exactitude et la précision. Sur cette courbe, on peut remarquer que pour une valeur de seuil de 0,85, la solution de détection permet de reconnaître l'intégralité des anomalies sans déclencher de faux positifs. En particulier, pour cette valeur de seuil, l'exactitude et la précision sont égales à 1.

(a) Valeurs des métriques



(b) Courbe ROC

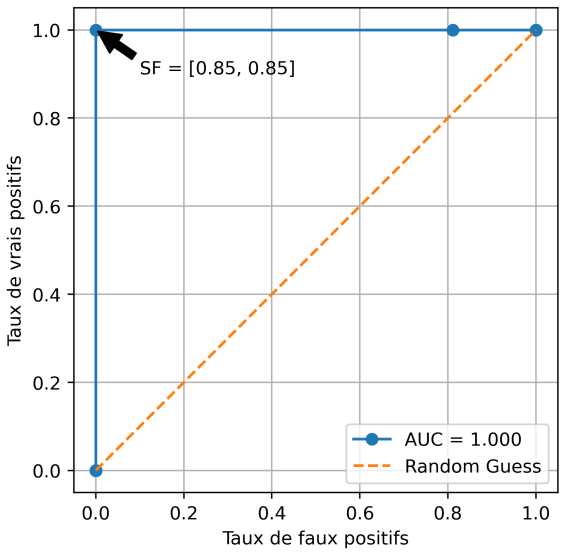


FIGURE 4.8 – Détection d'anomalies sur le système d'injection (Industrie 4.0)

La partie droite de la figure 4.8 représente la courbe ROC obtenue en représentant l'évolution des métriques précédentes en prenant en abscisse le taux de faux négatifs et en ordonnée le taux de vrais positifs. Cette courbe met bien en évidence la présence d'un unique seuil de fitness pour lequel la détection est optimale, c'est-à-dire pour lequel notre méthode de détection obtient un taux de faux positifs de 0% et un taux de vrais positifs de 100%.

Nous obtenons ici un cas assez particulier avec l'utilisation de la normalisation min-max et l'algorithme de partitionnement DBSCAN, avec une valeur de rayon maximal d'appartenance de 0,4. En effet, notre méthode de détection est capable de trouver l'ensemble des anomalies injectées sans avoir de faux positifs. Cela est explicable par le fait que les données proviennent d'un système simulé qui, dans un premier temps, ne contenait aucun bruit. Ainsi, chaque cycle du processus était exactement identique au précédent et donc finalement la seule différence entre le jeu de données pour l'apprentissage et celui de la détection était les anomalies injectées. Ces anomalies étaient donc facilement repérables.

Robots d'assistance. Les jeux de données que LuxAI a mis à notre disposition contenaient des informations relatives aux moteurs du robot. Le contenu des différents jeux de données a été présenté dans la section précédente et synthétisés dans le tableau 4.4. La figure 4.9 montre l'un des modèles générés par notre méthode dans sa phase d'apprentissage. Il est constitué de 22 places et de 32 transitions.

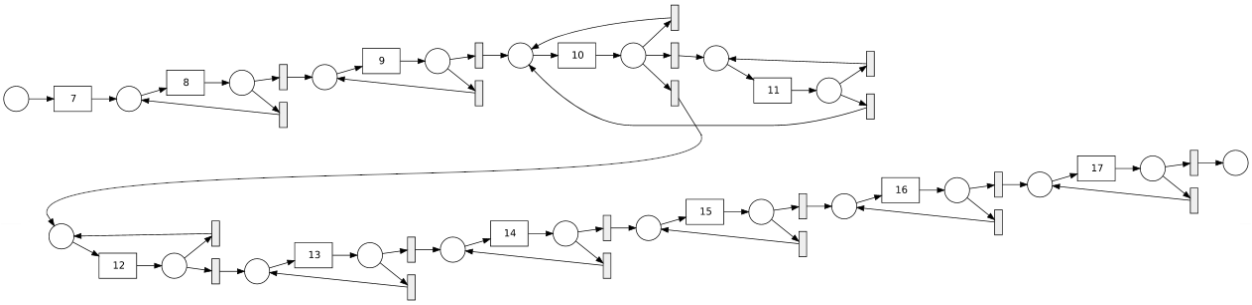


FIGURE 4.9 – Modèle d'un enchaînement de geste du robot d'assistance

Bien que ce modèle ne soit pas forcément évident à interpréter, la connaissance des valeurs des différents attributs au centre des groupes, trouvés par la phase de pré-traitement des données, permet de le déchiffrer. En effet, lorsqu'un groupe est composé de points de données ayant, par exemple, une valeur de 180 degrés pour le roulis de l'épaule gauche alors on considère que le maintien du bras gauche levé est l'un des états possibles du système. La figure 4.9 montre une décomposition d'un mouvement correspondant au lever progressif du bras gauche du robot. En plus des différentes données, un fichier indiquant la localisation des anomalies injectées nous a été fourni. Nous avons ainsi pu appliquer notre méthode de détection sur un nouveau jeu de données labélisées afin de vérifier la conservation des performances de détection. L'un de nos buts était également de s'assurer que nous avions évité le phénomène de sur-apprentissage lors du choix des paramètres de pré-traitement et de partitionnement.

(a) Valeurs des métriques

(b) Courbe ROC

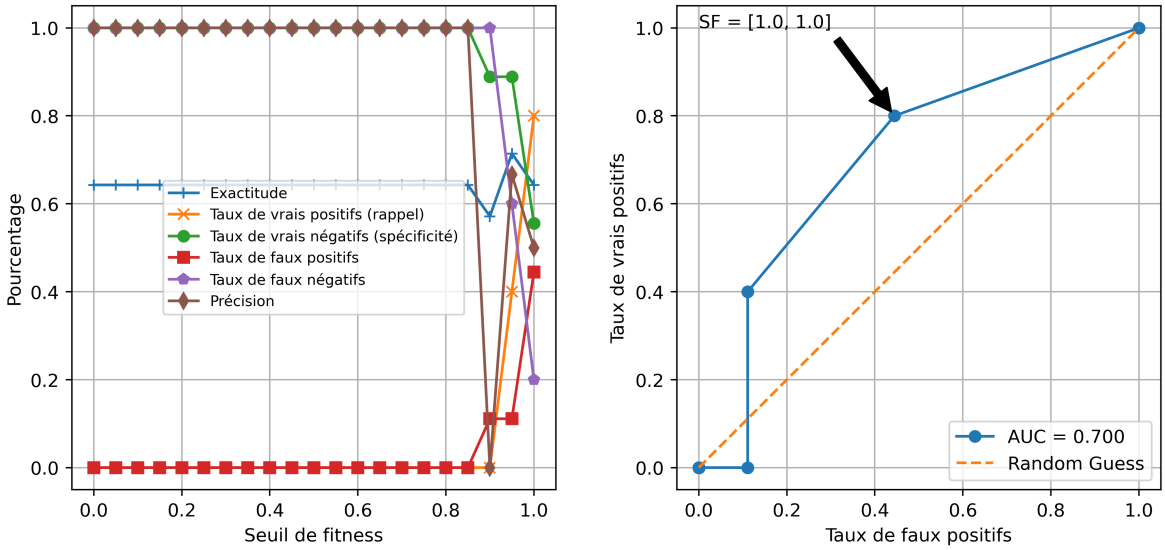


FIGURE 4.10 – Détection d'anomalies sur le robot d'assistance

Les résultats, que nous présentons ici, ont été obtenus en utilisant la normalisation min-max couplé à l'algorithme de partitionnement k-MEANS, avec K=10, avec un intervalle pour la découpe temporelle de 15 secondes. Ces paramètres sont ceux ayant obtenu les meilleurs résultats de détection lors de la phase d'apprentissage. La sous-figure (a) de la figure 4.10 représente l'évolution d'un ensemble de métriques telles que les taux de vrais et faux positifs / négatifs ou

bien encore l'exactitude et la précision. Ces différentes courbes n'évoluent pas avant l'utilisation d'un seuil de fitness de 0,85. Une fois ce seuil franchi, nous pouvons observer une baisse du taux de faux et de vrais négatifs tandis que ceux des vrais et faux positifs augmentent. L'exactitude a ici tendance à augmenter et la précision à diminuer, pour passer respectivement d'un peu plus de 0,6 à 0,8 pour l'exactitude et de 1 à 0,5 pour la précision. Cela signifie que nous détectons moins d'anomalies mais, surtout, que nous limitons fortement le nombre de faux positifs. La sous-figure (b) représente la courbe ROC obtenue pour le robot d'assistance. Cette dernière est constituée de quatre parties. Dans la première, notre système détecte exclusivement des points normaux comme étant des anomalies. Puis, à l'inverse lorsque nous allons augmenter un peu plus le seuil il va commencer à repérer des anomalies. Dans les deux dernières parties de la courbe, on va avoir simultanément une augmentation du taux de vrais et de faux positifs avec un ratio entre les taux de vrais positifs et de faux positifs initialement plus important.

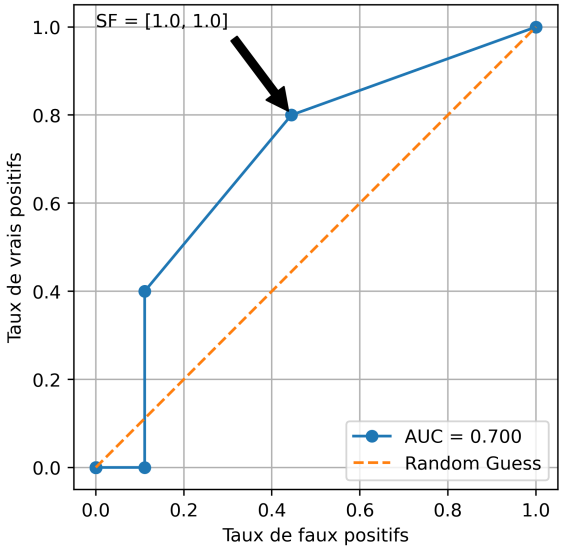


FIGURE 4.11 – Courbe ROC obtenue pour le robot d'assistance

À partir de cette courbe ROC, nous déduisons la valeur du seuil optimal à utiliser. En effet, la valeur du seuil pour laquelle nous nous rapprochons le plus possible des coordonnées (0 ;1) correspond au seuil pour lequel l'évaluation se rapproche le plus d'une détection parfaite des anomalies. Le point le plus proche d'une détection parfaite est indiqué par une flèche sur la figure à côté de laquelle nous pouvons lire l'intervalle des valeurs à utiliser comme seuil pour la *fitness* pour obtenir ce résultat.

4.4 Comparaison à d'autres méthodes de détection

Plusieurs méthodes ont été proposées dans le cadre de la détection d'anomalies. Dans [15], les auteurs ont classé ces méthodes en quatre catégories principales : les méthodes probabilistes et statistiques, les méthodes linéaires, les méthodes reposant sur la proximité et celles sur les grandes dimensions. Nous avons choisi un des algorithmes parmi les plus utilisés, pour chacune de ces catégories. Ils sont, respectivement, l'algorithme de l'*elliptic envelope*, l'algorithme du *one class SVM*, l'algorithme du *local outlier factor* et l'algorithme de l'*isolation forest*. En plus de ces algorithmes de détection d'anomalies, nous appliquons l'algorithme du *one class random*

forest, qui est une proposition récente de la littérature, et appartient aux méthodes reposant sur la proximité. Chacune de ces méthodes a été présentées dans le chapitre 2. Nous décrivons ci-dessous comment ces méthodes peuvent être intégrées à notre système de détection.

Dans la suite de cette section nous détaillons ces méthodes en nous appuyant sur les notations utilisées dans le chapitre 3 pour notre méthode de détection. Après cela, nous comparons les performances de détection obtenues par ces méthodes avec le *process mining*. Finalement, nous étudions quelles peuvent être les conséquences de perturbations sur ces résultats de détection. Nous considérons ici deux types de perturbations, le bruit et la perte de données, que nous illustrons avec l'ajout d'un bruit gaussien pour le premier et la suppression aléatoire d'un pourcentage des données à tester pour le second.

4.4.1 Description des méthodes

Dans cette section, nous formalisons les cinq méthodes de détection qui nous servent à évaluer la pertinence de notre méthode reposant sur le *process mining*.

Elliptic Envelope [83]. Les méthodes appartenant à la catégorie probabiliste et statistique supposent que les données suivent une distribution dont les paramètres doivent être déterminés pendant la phase d'apprentissage. En particulier, l'*elliptic envelope* fait l'hypothèse que les données suivent une distribution de Gausse. Le but de la phase d'apprentissage est de définir une ellipse contenant les données que l'on considère comme étant normales. Ainsi, lors de la phase de détection, les points se trouvant en dehors de l'ellipse seront considérés comme étant des données anormales et donc de possibles attaques. La taille de l'ellipse est calculée en utilisant la méthode des déterminants de covariance FAST-minimum. Celle-ci va utiliser la distance Mahalanobis D_M comme une fonction score S qui va permettre de déterminer le caractère anormal d'un enregistrement Er_i :

$$S(Er_i) = D_M(Er_i) = \sqrt{(val_i - \mu)^T \times S^{-1} \times (val_i - \mu)} \tag{4.1}$$

Dans cette équation, le terme μ est égal à $(\mu_1, ..., \mu_m)$ où, pour tout j dans l'intervalle $[\![1, m]\!]$, μ_j est la moyenne de $\{val_{1j}, ..., val_{nj}\}$. Pour tout i dans l'intervalle $[\![1, n]\!]$, val_i est égale à $(val_{i1}, ..., val_{im})^T$ et S est la matrice de covariance de $(val_i)_{i \in [\![1;n]\!]}$.

One Class SVM [143]. Les méthodes linéaires ont pour but d'amalgamer le maximum de points de données dans un sous-espace de dimension plus faible que celui initial, c'est-à-dire celui formé par l'ensemble des éléments du jeu de données. Les points qui ne sont pas inclus dans cet espace seront considérés comme étant des anomalies pendant la phase de détection car trop éloignés des points de comportements normaux.

One class SVM est l'algorithme appartenant à cette catégorie que nous avons considéré. Sa caractéristique principale est d'ajouter des marges au sous-espace afin de limiter le sur-apprentissage. Ainsi, pour classer un enregistrement Er_i, cette méthode repose sur la fonction de score $S(Er_i)$:

$$S(Er_i) = \sum_{j=1}^{m} \alpha_j . K(Er_i, val_{ij}) - \rho \tag{4.2}$$

Dans cette équation, les termes $(\alpha_j)_{j \in [\![1,m]\!]}$ sont les multiplicateurs de Lagrange. K est la fonction noyau qui permet de faire une projection de l'enregistrement sur le sous-espace définissant

l'ensemble de données normales et ρ est la marge. Les multiplicateurs de Lagrange sont obtenus lors de l'optimisation des éléments suivants :

$$\forall i \in [\![1,n]\!], \forall j,k \in [\![1,m]\!] : \min_{\alpha} \frac{1}{2}\alpha_j.\alpha_k.K(val_{ij}, val_{ik}) \tag{4.3}$$

tel que $0 \leq \alpha_j \leq \frac{1}{\nu m}$ et $\sum_{j=1}^{m} \alpha_j = 1$ où ν est le ratio d'enregistrements que l'on considère a priori anormaux.

Local Outlier Factor [42]. Les méthodes utilisant les proximités détectent comme anormaux les points de données possédant peu de voisins proches. L'algorithme que nous avons retenu est le *local outlier factor* qui prend en compte à la fois la distance entre les points de données et les densités locales. Ainsi, pour un enregistrement Er_i, cette densité locale correspond à la moyenne du ratio entre la densité d'accessibilité locale (lrd) de l'enregistrement Er_i et de celle de ses K plus proches voisins.

$$S(Er_i) = lof(Er_i) = \frac{1}{k}. \sum_{Er_j \in knn(Er_i)} \frac{lrd(Er_j)}{lrd(Er_i)} \tag{4.4}$$

Dans cette équation, k est un paramètre et $knn(Er_i)$ est l'ensemble des enregistrements des k plus proches voisins de Er_i. La densité d'accessibilité locale (lrd) se calcule comme montré dans l'équation 4.5.

$$lrd(Er_i) = \frac{k}{\sum\limits_{Er_j \in knn(Er_i)} reach\text{-}dist(R_i, R_j)} \tag{4.5}$$

Dans cette équation, le terme $reach\text{-}dist(Er_i, Er_j)$ est égal à $max\{k - dist(Er_j), d(Er_i, Er_j)\}$ où $k - dist(Er_j)$ correspond à la distance euclidienne des k voisins les plus proches par rapport à l'enregistrement Er_j. La distance entre Er_i et Er_j est donnée par l'équation 4.6.

$$d(Er_i, Er_j) = \sqrt{\sum_{k=1}^{m} |E_{ik} - E_{jk}|^2} \tag{4.6}$$

Isolation Forest [107]. Le fait d'identifier les données anormales lorsqu'elles possèdent un nombre important de caractéristiques peut présenter quelques difficultés. En effet, certaines de ses caractéristiques peuvent n'apporter aucune valeur lors de la détection, voir pourraient la rendre plus complexe. L'algorithme *Isolation Forest* est l'une des méthodes les plus efficaces pour éviter les problèmes induits par les grandes dimensions (curse of dimensionality). L'algorithme découpe les données de façon aléatoire en des sous-ensembles de dimension inférieure à la dimension initiale des données avant de construire des arbres binaires pour chacun de ces sous-ensembles. Chaque arbre est obtenu en découpant de façon récursive les données qu'il contient. Cette séparation est faite en sélectionnant un attribut aléatoire att_{ij} ainsi que la valeur de séparation val_{ij}. Cette partie de découpe récursive prend fin lorsque soit une hauteur d'arbre maximum est atteinte, soit il ne reste plus qu'un seul enregistrement dans le sous-ensemble à traiter, soit lorsque l'ensemble des enregistrements restant possèdent exactement les mêmes valeurs pour chacun des attributs. Pour permettre de distinguer un enregistrement anormal Er_i, l'algorithme va calculer la profondeur moyenne associée à cet enregistrement.

$$S(Er_i) = 2^{-\frac{E(h(Er_i))}{c(n)}} \tag{4.7}$$

On suppose que $c(n) = 2H(n-1) - 2(n-1)/|F|$, où $H(i)$ est le $i^{ème}$ harmonique : $H(i) = \sum_{k=1}^{i} \frac{1}{k}$.

One Class Random Forest [69]. L'algorithme *One Class Random Forest* est une extension de l'algorithme *random forest*. Il permet d'appliquer la méthode random forest même dans le cas où aucune anomalie ne serait présente dans le jeu de données d'apprentissage. Ainsi, cette extension considère les points suffisamment proches de ceux qui ont servi lors de la construction des modèles comme étant des points décrivant un état normal du système. Pour être plus précis, pour chaque noeud o de chacun des arbres qui sont construits par cet algorithme, les données anormales sont placées de façon à ce que leur cardinal soit proportionnel à celui des données normales. Notons F_o le sous-ensemble tiré du jeu de données *Données* dans le noeud o. On a donc :

$$|F'_o| = \gamma |F_o| \tag{4.8}$$

L'idée derrière l'algorithme du one class random forest est, dans un premier temps, de conserver une proportion identique de données normales et anormales dans chaque noeud. Ainsi, γ doit être le plus proche possible de 1. Afin d'évaluer l'hétérogénéité des données, l'indice Gini est utilisé. Cet indice est exprimé de la même façon que dans le cas d'un problème de classification faisant intervenir deux classes (les données normales et les anormales) :

$$i_G(o) = 2\frac{|F_o|}{|F_o| + |F'_o|}\frac{|F'_o|}{|F_o| + |F'_o|} \tag{4.9}$$

où $|F_o|$ (resp. $|F'_o|$) correspond au nombre d'enregistrements ayant le label 0 (respectivement 1) dans le noeud o. Afin de décider de l'anormalité d'un enregistrement $Er_i \in F$, on utilise une fonction score adaptée de celle utilisée par la méthode *Random Forest* que l'on décrit ci-dessous :

$$log_2(S(Er_i)) = -(\sum_{o \in feuilles} \mathbb{1}_{Er \in o} d_n + c(|F_o|))/c(|F|), \tag{4.10}$$

où d_o est la profondeur du noeud o, c(|F|) = 2H(|F| - 1) - 2(|F| - 1)/|F|, $H(k)$ est le $k^{ième}$ harmonique : $H(k) = \sum_{l=1}^{k} \frac{1}{l}$.

4.4.2 Comparaison des performances de détection

Dans les expériences suivantes, nous avons comparé les performances de détection entre le *process mining* et les cinq méthodes de détection d'anomalies mentionnées précédemment. Cette comparaison a été effectuée en utilisant les trois jeux de données qui nous ont été fournis dans le cadre du projet européen SecureIoT.

Dans le but de maximiser les performances de détection, nous avons sélectionné les paramètres du *process mining*, qui ont obtenu les meilleurs résultats de détection. Pour les autres méthodes de détection nous avons appliqué une normalisation aux données lors du pré-traitement lorsque cela été nécessaire, puis nous avons fait varier leurs paramètres sur la totalité de l'intervalle possible. Le tableau 4.8 récapitule les valeurs des meilleurs paramètres pour le *process mining*. Il existe un grand nombre de métriques faisant partie de la littérature et qui visent à évaluer les performances de détection d'anomalies. Nous avons choisi d'utiliser plusieurs de ces métriques pour évaluer les méthodes de détection avec le moins de biais possible [65, 108].

Dans la figure 4.12, on trouve les courbes ROC (Receiver Operating Characteristic), qui représentent le taux de vrais positifs en fonction du taux de faux positifs pour chacun des algorithmes de détection d'anomalies considérés. Les différents points de cette courbe sont obtenus lorsque l'on fait varier le seuil de détection. On trouve également la courbe de Precision Recall (PR) qui représente l'évolution de la précision en fonction du recall (taux de vrais positifs) dans la figure 4.13.

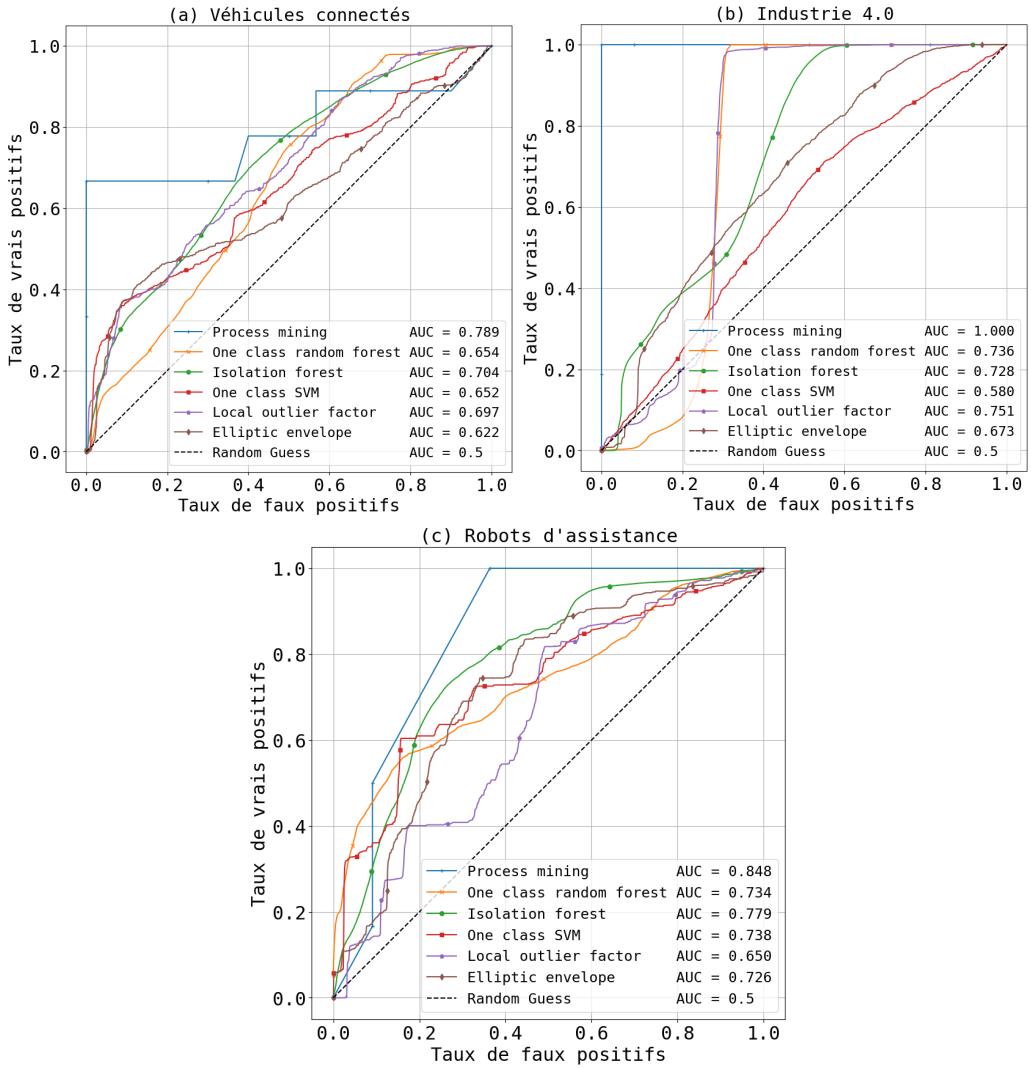


FIGURE 4.12 – Comparaison des courbes ROC obtenues par le *process mining* et les autres méthodes de détection pour les véhicules connectés (a), l'industrie 4.0 (b) et les robots d'assistance (c)

	Véhicule connecté	Machine à injection	Robot d'assistance
Découpe temporelle	30s	60s	15s
Algorithme de normalisation	min-max	min-max	min-max
Algorithme de partitionnement	K-MEANS (K = 8)	DBSCAN (ϵ = 0,4)	K-MEANS (K = 10)

TABLE 4.8 – Valeurs des paramètres de la phase de pré-traitement

En utilisant les différentes fonctions de scores définies par les équations 4.1, 4.2, 4.4, 4.7, et 4.10 ainsi que la *fitness* définit par 3.25, on sélectionne le seuil qui va maximiser le taux de vrais positifs tout en minimisant celui des faux positifs sur la courbe ROC. Après avoir sélectionné le seuil à utiliser, on le conserve, puis nous calculons la valeur des différentes métriques telle que l'aire sous la courbe ROC, l'aire sous la courbe precision recall, le F1 score (la moyenne harmonique de la precision et du recall), le coefficient de Matthews (MCC) (la corrélation entre la justesse des prédictions de détection), et la perte logistique (log loss) (l'incertitude du modèle). Les résultats obtenus pour les trois jeux de données ont été reportés dans la figure 4.14.

Puisque certains des algorithmes utilisés comportent une part d'aléatoire, le résultat présenté est la moyenne obtenue après trente expériences. La variance obtenue ne dépasse pas 6% et est même dans la plupart des cas inférieurs à 1%, c'est pourquoi nous l'avons négligée. Cependant, de façon exceptionnelle, nous avons observé que la perte logistique (log loss) du robot d'assistance pouvait varier grandement avec une variation allant jusqu'à 27%. Nous reviendrons sur cette variation dans la suite de cette section.

La figure 4.12 montre que le *process mining* présente la meilleure courbe ROC pour les trois jeux de données. En effet, comparé aux autres méthodes, le *process mining* obtient une aire sous la courbe ROC au moins supérieur de 12%, 33% et de 9% pour les données, respectivement, des véhicules connectés, des machines à injection et des robots d'assistances.

La figure 4.13 montre des résultats similaires à ceux des courbes ROC mais cette fois-ci concernant les courbes Precision Recall. En plus de l'aire sous la courbe ROC et PR, on remarque que le *process mining* obtient les meilleures performances pour l'ensemble des jeux de données pour la précision, le F1 score et le coefficient de Matthews, qui sont respectivement 68%, 56% et 47% plus élevés que ceux des autres méthodes.

La figure 4.14 nous montre les performances des méthodes de détection lors de l'analyse de chacun des jeux de données. Pour (a) les véhicules connectés et (b) les machines à injection, le *process mining* obtient les meilleures performances pour l'ensemble des métriques définies précédemment. Cependant, cela n'est pas le cas pour (c) les robots d'assistances pour lesquelles le *One Class SVM*, le *One Class Random Forest* et *Isolation Forest* obtiennent une exactitude (accuracy) et une estimation de la perte logistique (log loss) plus intéressant. Il est toutefois à noter que ce jeu de données est le plus déséquilibré entre les données normales et anormales. En effet, comme nous le montre le tableau 4.7, le ratio d'anomalies n'y est que de 10% alors qu'il est, respectivement, de 20 et 30% pour les véhicules connectés et les machines à injection.

Par conséquent, ce jeu de données est le plus vulnérable au paradoxe de l'exactitude, où la méthode de détection peut obtenir une très bonne exactitude en classant de manière agressive les points de données comme ne présentant pas un caractère anormal. Et effectivement, c'est ce phénomène que l'on rencontre ici car le recall qui montre la portion des données anormales détectées est très faible.

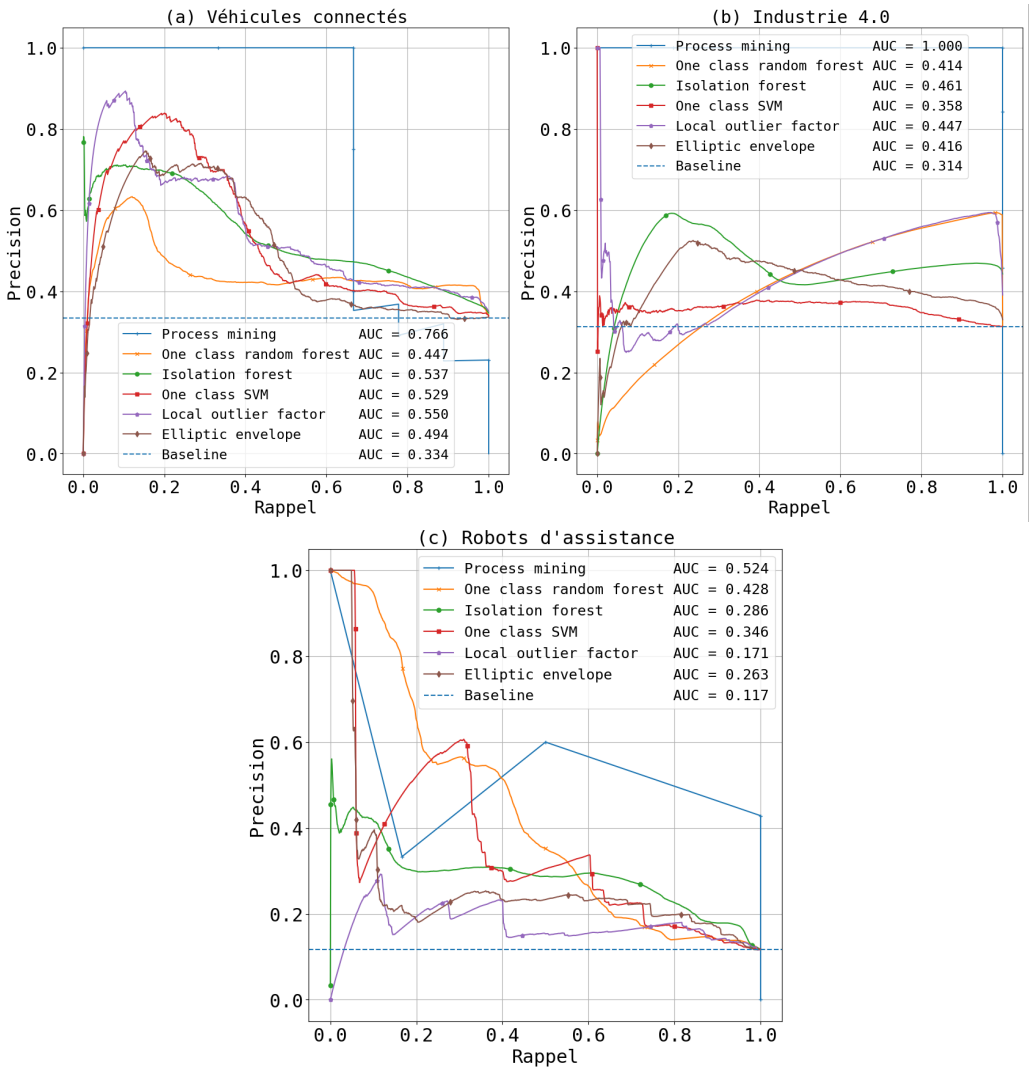


FIGURE 4.13 – Comparaison des courbes PR de l'ensemble des méthodes de détection pour les véhicules connectés (a), l'industrie 4.0 (b) et les robots d'assistance (c)

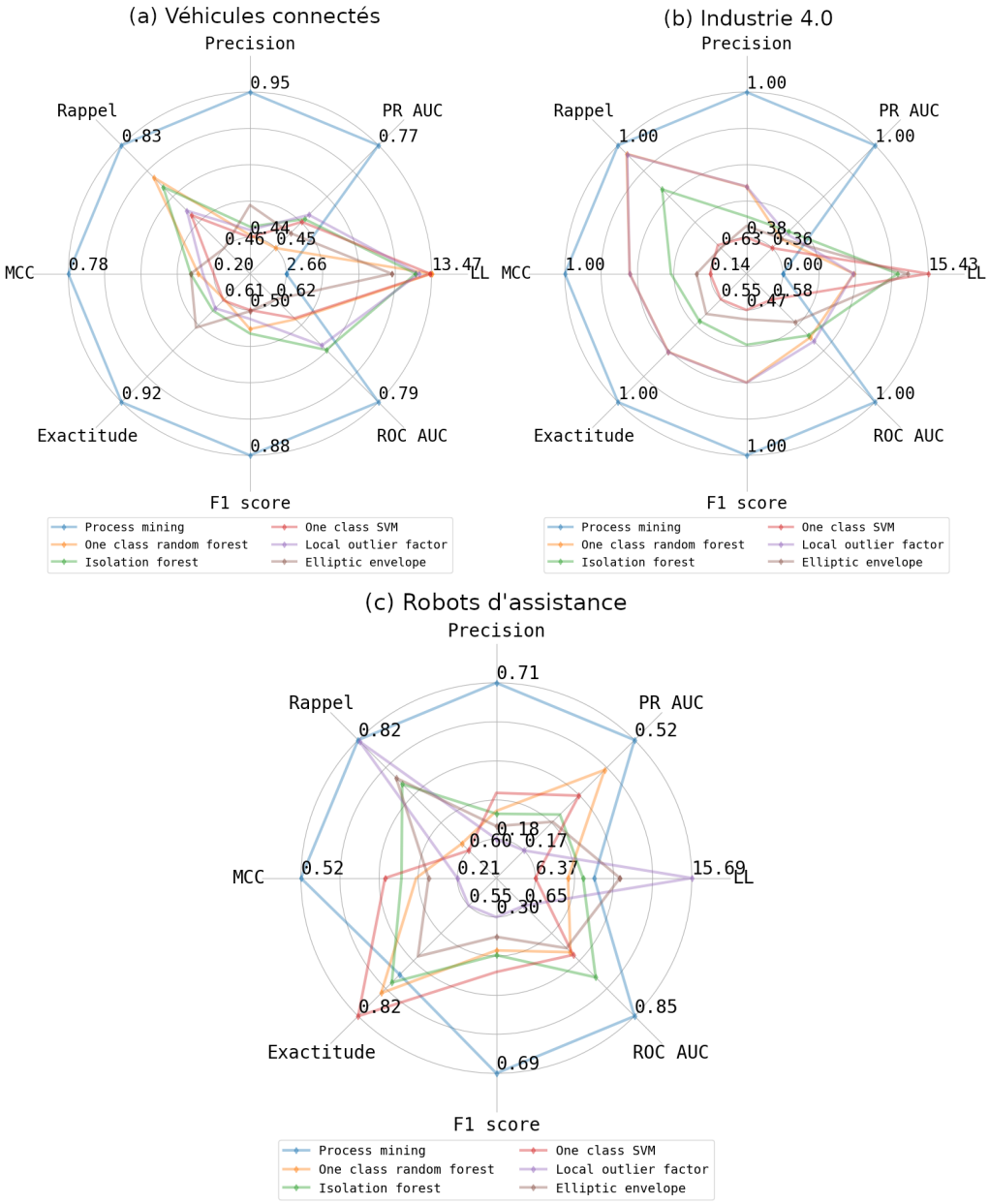


FIGURE 4.14 – Comparaison des performances du *process mining* et des autres algorithmes de détection en utilisant l'ensemble des métriques d'évaluation pour les véhicules connectés (a), l'industrie 4.0(b) et les robots d'assistance (c)

On peut remarquer ce phénomène en regardant également le MCC qui contrairement à l'exactitude prend en compte les quatre éléments de la matrice de confusion et est donc plus pertinent pour les jeux de données déséquilibrés.

La figure 4.14 nous montre que le *process mining* obtient les valeurs les plus intéressantes au niveau de la détection selon le MCC. Ainsi, le *process mining* obtient un MCC supérieur de 47%, 79% et 62% par rapport au *one class SVM*, *one class random forest* et *isolation forest*. C'est pour ces raisons que nous pensons que le *process mining* est le plus apte à analyser le jeu de données des robots d'assistances. Cette méthode est également plus pertinente que l'*elliptic envelope* et le *local outlier factor* bien que ce dernier ait un recall intéressant qui n'est que de 0,4% inférieur à celui du *process mining*. En revanche les autres métriques d'évaluation pour le *local outlier factor* ne donnent pas des scores très élevés.

Nous remarquons que notre méthode utilisant le *process mining* couplé à un pré-traitement des données a les meilleurs performances dans les différents scénarios expérimentaux que nous avons testés. Bien qu'il soit possible de sélectionner une découpe temporelle avec un intervalle différent pour la phase d'apprentissage et de détection, nous obtenons néanmoins de meilleures résultats lorsque nous considérons un intervalle identique. Pour notre méthode, le pré-traitement est un élément-clé. En effet, il permet de définir les états du système étudié, plus ceux-ci seront pertinents plus les modèles de comportement générés refléteront une réalité tangible. Le fait de considérer l'historique des états pris par le système avec l'utilisation du *process mining* nous permet d'obtenir de meilleurs résultats de détection que les cinq autres méthodes étudiées.

Nous avons observé que les points de données représentant les anomalies ne présentent pas la même répartition. Celles provenant des véhicules connectés sont très éparpillées ce qui explique en partie les mauvaises performances de la plupart des autres méthodes. En ce qui concerne les robots d'assistances, les anomalies restaient assez proches ce qui a permis à certains algorithmes, comme le *One Class Random Forest*, d'obtenir de bonnes performances d'exactitude, de recall tout en conservant une perte logistique (log loss) faible. Pour les données provenant des machines à injection, les anomalies sont proches des points de données correspondant au fonctionnement normal du système. Ainsi, les méthodes de détection considérant cette éventualité obtiennent de bonnes performances de détection, comme par exemple la méthode du *One Class Random Forest* ou bien encore celle du *Local Outlier Factor*. En revanche, les algorithmes qui construisent un espace devant contenir les points de données normales en se basant sur le calcul d'une distance, tels que l'algorithme *One Class SVM* ou l'algorithme *Elliptic Envelope* rencontrent de grandes difficultés.

Bien que notre méthode, reposant sur le *process mining*, permette de détecter les différentes anomalies contenues dans ces jeux de données, on peut tout de même constater que le temps de traitement n'est pas négligeable. En effet, l'évaluation faite par le *process mining* prend selon les jeux de données entre 30 et 60 fois plus de temps que, par exemple la méthode de l'*Isolation Forest*. Ce surcoût provient et de la phase de pré-traitement mais surtout de la partie cherchant le meilleur alignement avant l'application du *process mining*. Afin de maintenir un délai raisonnable pour le temps de détection, il serait possible d'utiliser le *process mining* couplé à une autre méthode qui permettrait d'alerter l'utilisateur de l'occurrence d'une attaque visible, tandis que le *process mining* permettrait une détection plus fine mais demandant un délai supplémentaire.

4.4.3 Conséquences de la présence de perturbations sur la détection

Afin d'évaluer la robustesse de notre solution, nous effectuons plusieurs expériences pour lesquelles nous avons soit ajouté du bruit dans les données soit fait disparaître un pourcentage des données disponibles. Dans le cas de l'ajout du bruit, nous avons injecté un bruit Gaussien

d'une intensité que nous avons fait varier, tandis que pour les pertes de données, nous avons fait varier le pourcentage de données et les avons retirées aléatoirement. Dans le but d'évaluer uniquement l'influence que peuvent avoir ces éléments, nous avons utilisé les mêmes paramètres de pré-traitement que dans la section 4.3.2 et sont donnés par le tableau 4.8.

Données bruitées. La plupart des métriques, notamment les taux de vrais ou faux positifs tout aussi bien que leurs variants négatifs, ne permettent pas séparément de pouvoir évaluer la pertinence des méthodes de détection. C'est pourquoi, nous avons décidé d'utiliser le coefficient de corrélation de Matthews (MCC) pour notre solution reposant sur le *process mining* et les autres méthodes de détection. Ces dernières ont toutes un comportement identique par rapport à l'injection de perturbations dans les données, c'est pourquoi nous n'avons représenté que l'une d'entre elles, la méthode de l'*isolation forest*.

La figure 4.15 montre l'évolution des performances de détection à la fois pour les méthodes de *process mining* et de l'*isolation forest* lorsque nous faisons varier le pourcentage de bruit dans les données. Il apparaît que le *process mining* est plus sensible au bruit que l'*isolation forest*. Cependant, le *process mining* conserve de meilleures performances de détection que l'ensemble des autres méthodes sur les jeux de données provenant de simulation (véhicules connectés et machines à injection) tant que le bruit reste inférieur à 20%. En ce qui concerne le jeu de données du robot d'assistance, nous avons pu observer une dégradation immédiate des performances de détection lors des expériences. Cette dégradation est en grande partie due au bruit initialement présent dans ce système physique. En effet, les groupes trouvés lors du partitionnement, pendant le pré-traitement des données du robot d'assistance, sont très proches. Ainsi, l'ajout de bruit, même léger, a des conséquences importantes sur la création des groupes et l'attribution des points de données à ceux-ci.

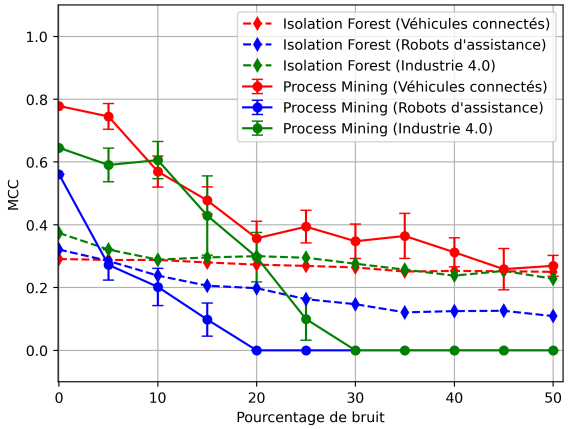


FIGURE 4.15 – Influence du bruit dans le jeu de données sur les performances de détection

Perte de données. Il est toutefois à noter que le bruit n'est pas la seule perturbation pouvant dégrader les performances d'un système de détection. On peut également se retrouver dans des cas où certaines données ne remontent pas des capteurs jusqu'au module d'analyse. Ainsi, la figure 4.16 nous montre l'évolution du coefficient de corrélation de Matthews pour le *process mining* et l'*isolation forest* lorsque nous faisons varier le taux de données manquantes. Il apparaît que notre approche est plus apte à détecter les anomalies que les autres méthodes tant que le taux de valeur manquante ne dépasse pas les 25%. Cependant, une fois ce seuil dépassé,

le *process mining* rencontre des difficultés pour mettre en avant les parties du jeu de données contenant des attaques. Cela est particulièrement vrai pour les jeux de données des véhicules connectés et des robots d'assistances. Pour la machine à injection, nous ne remarquons cette dégradation qu'à partir d'un taux de 80% de données manquantes. Il y a plusieurs raisons qui expliquent cette différence. Tout d'abord les machines à injection sont les seules qui obtiennent leurs meilleures performances en utilisant l'algorithme de partitionnement DBSCAN lors de la phase de pré-traitement. En effet, comme les données retournées sont presque exclusivement continues et numériques et que les autres éléments comme l'activation des pistons est directement liée à la valeur de l'un des paramètres numériques, nous nous retrouvons avec un modèle ne contenant qu'un seul groupe regroupant toutes les variations d'un fonctionnement normal. Ainsi, dans le cas des machines à injection, lorsque l'on observe un point trop éloigné du groupe cela signifie que nous avons affaire à une anomalie. Les résultats obtenus avec ce jeu de données sont donc à relativisés car nous nous retrouvons dans un cas où l'utilisation du *process mining* permet difficilement de détecter une utilisation malveillante du système tant que les informations envoyées par les capteurs restent suffisamment proches d'un ensemble de valeurs qui se rapprochent de l'utilisation normale du système.

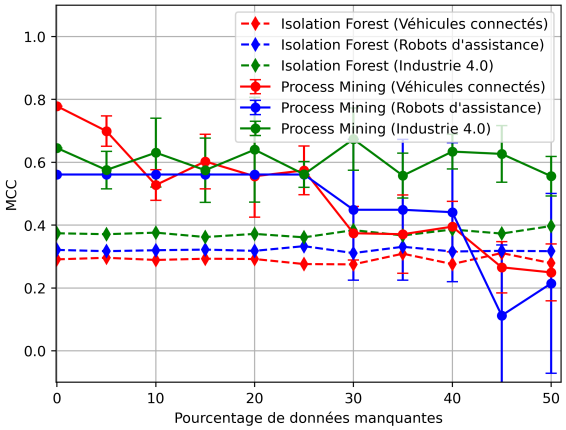


FIGURE 4.16 – Influence de la perte de données sur les performances de détection

Les expériences que nous avons réalisées montrent que notre méthode reposant sur le *process mining* peut, en partie, permettre de conserver les performances de détection d'anomalies lorsque le jeu de données contient du bruit ou que des données sont manquantes. La normalisation et le partitionnement lors du pré-traitement permettent de réduire l'influence du bruit tandis que la fréquence à laquelle sont relevées les données peut permettre de minimiser l'impact que pourraient avoir des données manquantes. Cette augmentation de fréquence de relevé augmenterait, de facto, la charge sur l'ensemble du système de détection, des capteurs en passant par le bus de transmission des données jusqu'au système de détection.

4.5 Synthèse

Dans ce chapitre, nous avons évalué l'applicabilité et les performances du *process mining* pour la détection d'anomalies avec différents systèmes IoT. Après une description des jeux de données considérés (véhicules connectés, industrie 4.0, robots d'assistance), fournis par des partenaires industriels, nous avons quantifié l'impact de différents paramètres de notre solution de détec-

tion. Lors de la phase d'apprentissage, nous avons pu construire pour chacun des cas d'usage, des modèles que nous avons ensuite testés avec des nouveaux jeux de données pour prévenir les phénomènes de sur-apprentissage. L'utilisation de méthodes de pré-traitement permet de réduire les risques d'explosion d'états, qui peuvent apparaître notamment avec des systèmes IoT hétérogènes. Les résultats obtenus montrent les bénéfices de notre solution au regard de ceux obtenus avec d'autres méthodes courantes de détection. Ainsi, le *process mining* obtient les meilleurs résultats de détection, pour la quasi-intégralité des métriques utilisées (exactitude, précision, rappel, MCC, ROC AUC, PR AUC). Lors des expérimentations menées, il détecte plus efficacement les anomalies que la méthode de l'*elliptic envelope*, la méthode à base de machines à vecteurs de support, la méthode du *local outlier factor* et la méthode de l'*isolation forest*. Son principal désavantage vient de sa complexité algorithmique qui accroît le temps de traitement nécessaire, et peut donc engendrer des temps de détection plus longs. Pour pallier ce problème, nous proposons de combiner plusieurs méthodes de détection au travers d'une approche ensembliste, dans laquelle plusieurs techniques sont appliquées simultanément.

Chapitre 5

Adaptation ensembliste de l'approche de détection

Sommaire

5.1 Introduction 82
5.2 Architecture utilisant l'apprentissage ensembliste 83
5.2.1 Modélisation des données d'entrée pour les systèmes complexes 83
5.2.2 Adaptation de l'architecture précédente à l'apprentissage ensembliste . . 85
5.3 Construction d'un graphe de dépendance 86
5.3.1 Contexte 86
5.3.2 Coefficient de Pearson 87
5.3.3 Construction manuelle par un expert 88
5.4 Apprentissage ensembliste 88
5.4.1 Choix des méthodes de détection 88
5.4.2 Définition des poids 89
5.5 Mécanisme de retour adaptatif 91
5.5.1 Fenêtre glissante temporelle 91
5.5.2 Adaptation de la taille de la fenêtre glissante 92
5.6 Synthèse 93

5.1 Introduction

L'objectif de ce chapitre est d'étendre notre solution de détection d'anomalies. En effet, la proposition que nous avons détaillée dans le chapitre précédent, bien que présentant des résultats prometteurs, possède certaines faiblesses comme notamment le temps de traitement. Nous avons donc choisi d'appliquer de manière conjointe cette solution de détection à d'autres méthodes de détection provenant des différentes familles d'algorithmes répertoriées dans [15]. Nous utilisons donc, la méthode de l'*Elliptic Envelope*, la méthode *One Class SVM*, la méthode du *Local Outlier Factor* ainsi que celle de l'*Isolation Forest*. Nous n'avons pas retenu la méthode *One Class Random Forest* car elle ne présente pas d'avantages significatifs par rapport aux autres méthodes du chapitre précédent. L'utilisation de cette diversité de méthodes doit permettre d'améliorer la détection en général. Nous pouvons envisager des cas où certaines des méthodes ne remarqueront pas une attaque ciblant le système à protéger, tandis que d'autres seront plus

efficaces. Pour cela, nous proposons une nouvelle architecture qui va nous permettre de prendre en compte les résultats de détection de l'ensemble des méthodes de détection que nous avons sélectionnées. Cette architecture s'inscrit dans une démarche d'apprentissage ensembliste pour définir une stratégie regroupant les scores de détection de ces méthodes.

Dans la suite de ce chapitre, nous décrivons la forme prise par nos données brutes qui nous proviennent directement de systèmes soit réels soit simulés. Ensuite, nous détaillons le fonctionnement de notre architecture et expliquons quels en sont les composants principaux. Nous montrons également comment ceux-ci vont nous permettre de détecter, de façon plus fine, les attaques comportant plusieurs étapes. Finalement, nous détaillons le fonctionnement de chacun de ces composants : (1) le bloc de construction du graphe de dépendance, qui va permettre d'établir les liens existant entre les différentes sources de données d'un même système ; (2) le bloc de détection utilisant l'apprentissage ensembliste, qui va définir la stratégie d'agrégation pour les scores de nos algorithmes de détection d'anomalies ; et (3) le bloc contenant le mécanisme de retour adaptatif qui va nous servir à limiter les faux positifs.

5.2 Architecture utilisant l'apprentissage ensembliste

Dans cette section, nous détaillons, tout d'abord, comment la modélisation des données d'entrée fournies à notre solution de détection d'anomalies a été modifiée. En effet, nous considérons, dans ce chapitre, des systèmes complexes pour lesquels nous devons prendre en compte une multiplicité de sources de données. Puis, nous montrons la transformation de l'architecture de notre système de détection afin d'utiliser l'apprentissage ensembliste.

5.2.1 Modélisation des données d'entrée pour les systèmes complexes

Les données brutes en entrée du système de détection sont identiques à celles que l'on a pu décrire dans le chapitre précédent. Toutefois, dans ce chapitre, nous ne nous limitons pas à un seul jeu de données pour un système. Le but est de pouvoir suivre la progression d'une attaque composée de plusieurs étapes. Ainsi les données qui, jusqu'à présent, ne comportaient qu'une seule source constituée de plusieurs enregistrements, comme défini dans l'équation 3.1, seront décrites de façon plus précise par l'équation 5.1 où q est le nombre de jeux de données, ou sources de données, pour un système et n_a est le nombre d'enregistrements pour le aième jeu de données. Ainsi, Er_{a_i} permet de faire référence au ième enregistrement du aième jeu de données.

$$Données = \{\{Er_{1_1}, \ldots, Er_{1_{n_1}}\}, \ldots, \{Er_{q_1}, \ldots, Er_{q_{nq}}\}\} \tag{5.1}$$

De la même façon, chacun des enregistrements Er_{a_i} contient m_a éléments $E_{a_{ij}}$, comme décrit dans l'équation 5.2, et là encore chaque élément est composé d'un ensemble de clés, que nous appelons attributs, et de valeurs. Une description de ces éléments est donnée par l'équation 5.3. Là aussi, il est nécessaire que l'ensemble des différents attributs de chacun des enregistrements soient identiques. En revanche, ces attributs peuvent être différents d'un jeu de données à l'autre. Les attributs doivent rester identiques pour chacun des enregistrements d'un des jeux de données, comme le montre l'équation 5.4, cela va permettre de conserver une cohérence des modèles de comportement et de l'évaluation d'un jeu de données anormal.

$$\forall i \in [\![1, n_a]\!], Er_{a_i} = \{E_{a_{i1}}, \ldots, E_{a_{im_a}}\} \tag{5.2}$$

$$\forall i \in [\![1, n_a]\!], \forall j \in [\![1, m_a]\!], E_{a_{ij}} = \{attribut_{a_{ij}} : valeur_{a_{ij}}\} \tag{5.3}$$

$$Soit\ j\ \in [\![1, m_a]\!], \forall i, k \in [\![1, n_a]\!], attribut_{a_{ij}} = attribut_{a_{kj}} \tag{5.4}$$

Afin d'illustrer au mieux cette décomposition, nous allons considérer plusieurs sources de données fournis par un véhicule connecté contenant respectivement un, deux et trois enregistrements et, étant eux-mêmes composés de plusieurs éléments. La première source de données nous donne des informations sur la partie réseau, les attributs sont l'adresse ip de la source, l'adresse ip de la destination ainsi que le nombre de bits reçus et envoyés. Pour la seconde source, nous retrouvons les écritures sur le premier disque ainsi que sur le second. Enfin la troisième contient les éléments relatifs à la charge du bus CAN du véhicule ainsi qu'au nombre de messages inattendus que celui-ci reçoit. Avec cet exemple, l'équation 5.1 va nous permettre de compter le nombre d'enregistrements pour chacun des jeux de données. La figure 5.5 est le résultat de cette adaptation dans le cas de notre exemple.

$$Données = \{\{Er_{1_1}\}, \{Er_{2_1}, Er_{2_2}\}, \{Er_{3_1}, Er_{3_2}, Er_{3_3}\}\} \tag{5.5}$$

Chacun des enregistrements du premier jeu de données est composé des mêmes trois éléments mentionnés précédemment pour la partie réseau. Et, de façon identique, les enregistrements du second jeu de données vont tous contenir les attributs relatifs à l'écriture sur les deux disques alors que ceux du troisième donneront la charge du bus CAN ainsi que le nombre de messages inattendus reçus par le système. Ainsi, l'équation 3.3 va nous permettre de lister, pour un jeu de données et un enregistrement donnés, le nom des attributs, ainsi que leurs valeurs respectives. L'équation 5.6 donne un exemple de ce que peut contenir le premier enregistrement de chacun des jeux de données.

$$E_{1_{11}} = \{adresse\_ip\_source : 192.168.0.4\} \tag{5.6}$$
$$E_{1_{12}} = \{adresse\_ip\_destination : 192.168.0.8\} \tag{5.7}$$
$$E_{1_{13}} = \{bit\_reu : 543\} \tag{5.8}$$
$$E_{1_{14}} = \{bit\_envoyé : 32\} \tag{5.9}$$

$$E_{2_{11}} = \{écriture\_disque\_1 : 243\} \tag{5.10}$$
$$E_{2_{12}} = \{écriture\_disque\_2 : 24\} \tag{5.11}$$

$$E_{3_{11}} = \{charge\_bus\_CAN : 0.2\} \tag{5.12}$$
$$E_{3_{12}} = \{message\_innatendu : 112\} \tag{5.13}$$

L'équation 5.4 est là pour illustrer la notion de cohérence que nous avons évoquée précédemment dans cette sous-section. Ainsi, pour notre exemple, nous montrons dans cette équation que l'attribut du premier élément de chacun des enregistrements du troisième jeu de données correspond à la charge du bus CAN. Là encore, il est à noter que seuls les noms des attributs sont identiques et que leurs valeurs peuvent être différentes.

$$attribut_{3_{11}} = attribut_{3_{21}} = attribut_{3_{31}} = charge\_bus\_CAN \tag{5.14}$$

Dans la sous-section suivante nous décrivons le principe de fonctionnement de notre architecture.

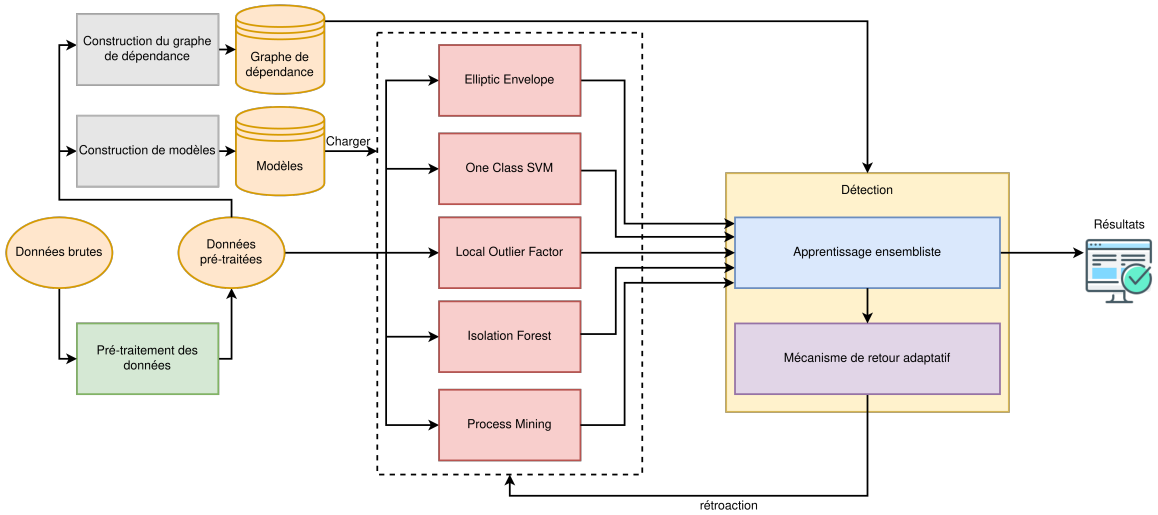


FIGURE 5.1 – Architecture de la solution de détection utilisant l'apprentissage ensembliste

5.2.2 Adaptation de l'architecture précédente à l'apprentissage ensembliste

L'architecture de notre méthode d'apprentissage ensembliste avec ses différents composants est représentée par la figure 5.1. Nous distinguons une phase d'apprentissage et une phase de détection. La phase d'apprentissage est très proche de celle présentée dans le chapitre 3. Nous allons, tout d'abord, construire les modèles de comportement pour chacun des jeux de données et pour chacun des algorithmes de détection que nous avons choisis à partir des données brutes d'entrées. Pour cela, les données vont être pré-traitées par une méthode de normalisation pour l'ensemble des méthodes de détection. Pour le *process mining*, une méthode de partitionnement est appliquée sur les données normalisées. Après cela, nous générons un ensemble de modèles de comportement à partir de ces données pré-traitées. Pour pouvoir appliquer notre solution de détection à des systèmes complexes, nous construisons un graphe de dépendances prenant en compte la propagation de l'attaque dans les systèmes IoT. Ce graphe a pour objectif de nous renseigner sur les liens pouvant exister entre les différents éléments qui composent un système sensible. De cette façon, lorsqu'un problème sera détecté sur une partie de notre système, nous serons alerté, et donc nous redoublerons de vigilance afin de nous assurer que l'attaque ne se propage pas.

En ce qui concerne la phase de détection, nous nous écartons de la proposition du chapitre 3, où une seule méthode de détection était utilisée pour cette phase. Les données pré-traitées vont être utilisées simultanément par l'ensemble des algorithmes de détection sélectionnés, puis leurs résultats vont être envoyés dans le bloc de détection. C'est le sous-bloc d'apprentissage ensembliste qui va regrouper ces différents résultats, ou scores, pour obtenir un score global qui va nous permettre de pouvoir faire la différence entre un comportement normal et une attaque de notre système. Nous avons considéré plusieurs stratégies d'agrégation des scores afin de tester leurs pertinences. Nous les décrivons plus en détail dans l'une des sections suivantes. Le résultat global obtenu par notre sous-bloc d'apprentissage ensembliste va directement influer sur le fonctionnement des algorithmes de détection. En effet, afin de réduire les faux positifs induits par de possibles variabilités dans les données, nous avons choisi de considérer la moyenne du score de détection de chaque algorithme à l'intérieur d'une fenêtre temporelle. Nous pouvons donc réduire ou agrandir cette fenêtre selon les résultats de détection obtenus par l'apprentissage ensembliste.

Dans la suite du manuscrit, nous expliquons comment peut être construit le graphe de dépendances et pourquoi nous avons utilisé les cinq méthodes de détection données dans la figure 5.1. Nous détaillons également les stratégies d'agrégation des scores que nous avons considérées pour l'apprentissage ensembliste avant d'expliquer, dans une dernière section, le principe de notre mécanisme de retour adaptatif.

5.3 Construction d'un graphe de dépendance

Dans cette section, nous expliquons pourquoi nous avons choisi d'utiliser un graphe de dépendances avant de décrire comment celui-ci est construit. Ce graphe nous permet de rendre compte des dépendances entre les différents jeux, ou sources de données. Nous pouvons automatiser cette construction en cherchant comment les attaques se propagent dans le système. Il est également possible de le construire manuellement en faisant intervenir des experts connaissant le système à protéger.

5.3.1 Contexte

La phase de construction d'un graphe de dépendance doit nous permettre d'identifier le lien entre les différentes sources de données. Ces dernières nous donnent des informations en provenance de plusieurs types de capteurs ou d'appareils qui peuvent être physiquement ou logiquement connectés. L'utilisation d'un graphe de dépendance permet à notre solution de détection d'anomalies de connaître le contexte dans lequel elle évolue. Les systèmes construits à partir d'objets connectés reposent souvent sur une architecture distribuée où l'on retrouve ces dits objets ainsi que les réseaux permettant de les relier. Par conséquent, les flux de communication peuvent montrer s'il existe une dépendance entre deux éléments et donc de mettre en évidence le lien entre eux. En connaissant les liens de dépendance entre les différents éléments qui composent le système, nous pouvons anticiper comment peut se propager une attaque. Nous pouvons alors envisager de prendre des mesures préventives pour limiter la propagation de l'attaque dans le système avant qu'elle n'atteigne une partie critique ou bien encore d'éteindre complètement le système si cela s'impose. La figure 5.2 fait référence à un exemple simplifié d'infrastructure IoT :

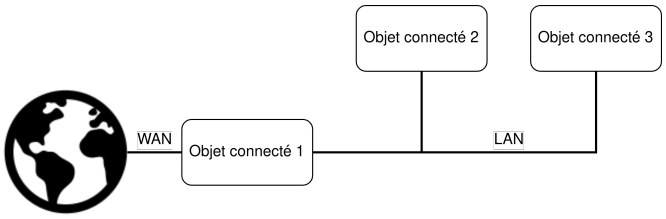


FIGURE 5.2 – Exemple d'une infrastructure IoT

un objet connecté est relié à un réseau étendu (WAN) et un réseau local (LAN), tandis que deux autres objets connectés sont eux uniquement reliés au LAN. Dans la figure 5.3, l'infrastructure précédente a été traduite en un graphe de dépendance qui nous permettra de prendre des mesures préventives dans le cas d'une attaque cherchant à compromettre l'ensemble du système. Une telle attaque aurait de grandes chances de provenir du WAN avant de se propager aux différents objets connectés en passant d'abord par l'objet connecté 1 avant d'arriver dans le LAN pour finalement atteindre les objets connectés 2 et 3.

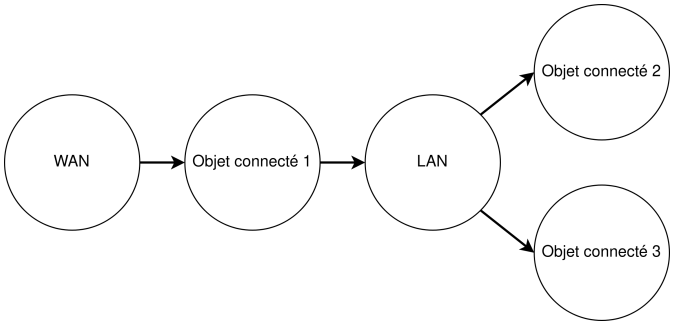


FIGURE 5.3 – Graphe de dépendance correspondant

5.3.2 Coefficient de Pearson

Nous allons à présent décrire la première manière de générer un tel graphe. Nous pouvons le construire automatiquement en calculant le coefficient de corrélation entre les scores de détection des différentes sources de données. Plus précisément, nous observons cette corrélation croisée après avoir lancé une attaque sur le système. Ainsi, nous pouvons distinguer le chemin de propagation de l'attaque en nous référant à la différence temporelle entre la détection d'une anomalie pour un premier élément du réseau et pour un second. En effet, dans le cas d'une attaque complexe, nous nous attendons à ce que celle-ci impacte plusieurs éléments du système. Nous formalisons ci-dessous la méthode de corrélation de Pearson que nous avons utilisée pour trouver les relations entre les différents éléments du système. Nous notons T comme étant l'ensemble temporel des horodatages dans l'équation 5.15.

$$T = \{value_{a_{i1}} \in \mathbb{R} \ ; \ i \in [\![1, n_a]\!] \ et \ a \in [\![1, q]\!]\} \tag{5.15}$$

Les scores que nous renvoient les méthodes de détection sont regroupés dans un ensemble S. Ces scores, décrits dans l'équation 5.16, peuvent être comparés à un ensemble de fonctions qui vont renvoyer un nombre réel pour chaque horodatage.

$$S = \{s_i \ : \ T \to \mathbb{R} \ ; \ i \in [\![1, I]\!]\} \tag{5.16}$$

Nous écrivons $s_i(t)$ le score de la méthode de détection i à l'instant t, avec $\sigma_{s_i(t)}$ qui correspond à sa déviation standard et $\mu_{s_i(t)}$ sa moyenne. La corrélation de Pearson, que l'on note $\rho_{s_i(t)s_j(t')}$ dans l'équation 5.17, correspond à la covariance de deux scores de détection $s_i(t)$ et $s_j(t')$ dans S que l'on divise par le produit de leur déviation standard.

$$\rho_{s_i(t)s_j(t')} = \frac{\mathbb{E}[\,(s_i(t) - \mu_{s_i(t)})(s_j(t') - \mu_{s_j(t')})]}{\sigma_{s_i(t)}\sigma_{s_j(t')}} \tag{5.17}$$

Soit τ le délai maximum que l'on considère pour distinguer la propagation d'une attaque entre deux éléments du système. La corrélation croisée maximum que l'on note C est la valeur maximale de corrélation de Pearson entre le premier score de détection et le second qui peut être décalé au maximum de τ. Pour trouver la corrélation croisée maximum, on calcule la corrélation de Pearson en prenant en compte un décalage pour le deuxième score entre $-\tau$ et τ qui représente le délais maximum que l'on envisage pour la propagation d'une attaque. On définit C dans l'équation 5.18.

$$C = \max_{-\tau \leq d \leq \tau} \frac{\mathbb{E}[\,(s_i(t) - \mu_{s_i(t)})(s_j(t+d) - \mu_{s_j(t+d)})]}{\sigma_{s_i(t)}\sigma_{s_j(t+d)}} \tag{5.18}$$

	WAN	Objet 1	LAN	Objet 2	Objet 3
WAN		0,8	0,5	0,2	0,6
Objet 1			0,3	0,5	0,1
LAN				0,7	0,9
Objet 2					0,4
Objet 3					

TABLE 5.1 – Application du coefficient de Pearson sur l'exemple

Une fois que nous avons accès à l'ensemble des coefficients de corrélation croisée, nous les comparons à une valeur de seuil C_{seuil}. Si la valeur du coefficient est plus élevée que le seuil, alors cela signifie qu'il existe un lien de corrélation pour le score de détection entre ces deux sources de données. Dans ce cas, on relie ces deux éléments de notre système dans le graphe de dépendance. En revanche, si le coefficient est trop faible, nous considérons que les deux éléments du système ne sont pas directement reliés.

5.3.3 Construction manuelle par un expert

Afin de construire le graphe de dépendance complet, la stratégie précédente (construction automatique) nécessite d'avoir à notre disposition une ou des attaques de tests mettant en exergue l'ensemble des liens présents dans le système. En effet, si l'une des liaisons n'est jamais utilisée lors de nos propres attaques qui visent à construire ce graphe, alors il n'existera aucune arête correspondante dans celui-ci. Ainsi, si nous reprenons l'exemple d'infrastructure donné par la figure 5.2, nous pouvons tout à fait obtenir les coefficients de corrélation du tableau 5.1. Dans ce dernier nous considérons que $C_{seuil} = 0,7$ est que donc la présence d'un coefficient supérieur ou égal à cette valeur entre deux sources d'un système signifie qu'il existe un lien direct entre elles. Le lien entre l'objet connecté 1 et le LAN ne sera jamais découvert. C'est pourquoi il semble nécessaire de laisser la possibilité à des experts de construire le graphe de dépendance en s'aidant ou non des liens mis en évidence par l'utilisation de la méthode de corrélation de Pearson.

5.4 Apprentissage ensembliste

Dans cette section, nous décrivons notre approche de détection qui utilise une stratégie d'apprentissage ensembliste. Cette stratégie consiste à agréger le score de plusieurs méthodes de détection qui sont exécutées en parallèle sur l'ensemble des sources de données. À partir de cette agrégation, nous obtenons un score global grâce auquel nous pouvons détecter, de façon plus précise, les potentielles attaques et anomalies pouvant poser des problèmes de sécurité pour notre système. Dans la suite, nous décrivons les méthodes de détection que nous avons sélectionnées dans notre approche ensembliste. Puis, nous détaillons comment leurs scores ont été agrégés afin d'obtenir un score de détection globale.

5.4.1 Choix des méthodes de détection

Nous avons inclus, dans notre architecture, quatre méthodes de détection couramment utilisées dans la détection d'anomalies et avons ajouté notre méthode reposant sur le *process mining*. L'idée est d'intégrer plusieurs méthodes de détection fonctionnant de façon différente dans notre architecture. Nous cherchons ainsi à améliorer au mieux les performances de détection. En effet, comme leurs fonctionnements sont différents, nous pouvons nous attendre, par exemple, à ce

que l'une de ces méthodes de détection donne de bons résultats pour certains types d'attaques mais pas pour d'autres. Et, dans ce cas, une autre méthode peut donner de meilleurs résultats et prendre le relais. Ainsi, en agrégeant les scores renvoyés par l'ensemble de nos méthodes, il devrait être possible de détecter une plus grande diversité de types d'attaques. Nous rappelons ci-dessous les cinq méthodes que l'on peut retrouver dans la figure 5.1.

— La méthode de détection probabiliste, *elliptic envelope*, fait l'hypothèse que les données qui lui sont fournies suivent une distribution de Gauss. Cette méthode cherche donc les paramètres de distribution qui lui permettent au mieux de coller aux données d'entrée lors de la phase d'apprentissage. Ensuite, elle définit une ellipse autour des points de données et les points en dehors de l'ellipse sont considérés comme anormaux lors de la phase de détection.

— La méthode de détection linéaire, *One Class SVM (OCSVM)*, cherche à intégrer le maximum de points de données pré-traitées dans un sous-espace de dimension inférieure à celui défini par les attributs. La phase d'apprentissage consiste à trouver ce sous-espace, tandis que les points n'en faisant pas partie sont considérés comme anormaux lors de la phase de détection.

— La méthode qui utilise la notion de proximité est appelée *Local Outlier Factor*. Elle prend en compte l'absence de densité de points de données pré-traitées et des distances entre les différents points et les densités locales. Ainsi, la méthode considère comme anormaux les points isolés, c'est-à-dire les points se trouvant dans des zones de faibles densités.

— La méthode sensible à l'isolation des données et utilisable pour des grandes dimensions, *Isolation Forest*, sépare les données de façon aléatoire en sous-ensembles de dimensions inférieures et construire un arbre pour chacun des sous-ensembles. L'arbre est obtenu en séparant récursivement les données nécessaires en choisissant aléatoirement un attribut jusqu'à ce que soit l'arbre atteigne la taille maximale, soit que les données à utiliser ne comportent qu'un élément ou bien que rien ne permette de distinguer les données restantes. En ce qui concerne la phase de détection, les nouvelles données passent à travers les arbres créés dans la phase d'apprentissage et leurs parcours dans ces arbres nous permettront d'identifier les données anormales.

— La méthode de détection qui utilise le *process mining* repose en fait sur une combinaison de deux techniques d'apprentissage machine. La première est une méthode de partitionnement qui est utilisée sur les données brutes et qui sert à extraire les états possibles que peut prendre le système. Après cela, un algorithme de *process mining* est appliqué dans le but de construire les modèles de comportement du système. Cette approche a été décrite en détail dans le chapitre 3.

5.4.2 Définition des poids

Pour chaque jeu de données, les différentes méthodes de détection fournissent un score que nous notons s_i où i fait référence à la ième méthode. Ces scores sont agrégés à l'aide d'une stratégie propre à l'apprentissage ensembliste afin d'obtenir un score global. Cette agrégation consiste à regrouper de façon linéaire, en utilisant différents types de poids pour les différentes stratégies, les scores de l'ensemble des méthodes. Ce calcul linéaire est décrit dans l'équation 5.19. Dans cette équation I fait référence au nombre total de méthodes de détection de notre stratégie d'apprentissage ensembliste, c'est-à-dire que I vaut 5. Nous notons W_i le poids associé à la ième méthode de détection au temps t.

$$score(t) = \sum_{i=1}^{I} W_i s_i(t) \tag{5.19}$$

Nous avons implémenté plusieurs stratégies pour l'apprentissage ensembliste. En effet, nous avons utilisé différents types de poids dans le calcul du score global afin de permettre d'améliorer les performances de la détection. Nous avons considéré quatre stratégies pour l'apprentissage ensembliste que nous avons appelées EL_{uni}, EL_{exa} et EL_{exp} ; les trois premières sont définies dans [153]. La stratégie supplémentaire est notée EL_{max} et nous la supposons être très réactive car elle ne prend en compte que la méthode de détection ayant le score le plus élevé.

Chacune des stratégies mentionnées précédemment a recours à un type particulier de poids qu'elle applique aux scores des différentes méthodes de détection. Ainsi, respectivement, les stratégies utilisent les poids $score_{uni}$, $score_{exa}$, $score_{exp}$, et $score_{max}$. La première stratégie considère un poids uniforme pour les différentes méthodes de détection, comme le montre la formule 5.20.

$$score_{uni}(t) = \sum_{i=1}^{I} \frac{1}{I} s_i(t) \tag{5.20}$$

La seconde stratégie donne plus d'importance aux méthodes de détection ayant une grande exactitude. La définition du calcul du score de EL_{exa} est donnée par la formule 5.21 où la variable a_i correspond à l'exactitude de la ième méthode de détection.

$$score_{exa}(t) = \sum_{i=1}^{I} \frac{a_i}{\sum_{j=1}^{I} a_j} s_i(t) \tag{5.21}$$

La troisième stratégie amplifie d'autant plus l'importance de l'exactitude en y ajoutant un paramètre exponentiel λ. Cela va réduire, grandement l'influence des méthodes de détection ne parvenant pas ou mal à détecter les anomalies. De cette façon, seules les méthodes de détection qui arrivent à obtenir les plus hauts scores d'exactitude ont un réel impact sur le score global. Nous avons utilisé la même valeur λ que celle recommandée dans [153], c'est-à-dire 10, dans la formule 5.22 qui détaille cette stratégie.

$$score_{exp}(t) = \sum_{i=1}^{I} \frac{e^{\lambda ai}}{\sum_{j=1}^{I} e^{\lambda aj}} s_i(t) \tag{5.22}$$

Enfin, la quatrième et dernière stratégie cherche à diminuer au maximum le temps de détection, et ne prend en compte que le score le plus élevé. De cette façon, le score global ne peut pas être dilué par les autres méthodes de détection. Nous supposons donc que si l'une d'entre elles repère une anomalie, nous avons bien affaire à une attaque. Cette stratégie est définie par la formule 5.23.

$$score_{max}(t) = \textstyle\sum_{i=1}^{I} W_{max_i} s_i(t) \quad avec : \tag{5.23}$$

$$W_{max_i} = \begin{cases} 1 & \text{si } s_i(t) = \max\limits_{\forall j \in I}(s_j(t)) \ et \ \sum\limits_{\substack{j=0 \\ j \neq i}}^{I} W_{max_j} = 0 \\ 0 & \text{sinon} \end{cases}$$

Une fois que le score global est calculé, il faut le comparer à une valeur de seuil S_{seuil} dans le but de distinguer les parties des données qui contiennent des anomalies de celles qui reflètent un comportement normal. L'utilisation de ces différentes stratégies pour l'apprentissage ensembliste nous permet d'adapter l'influence des méthodes de détection en prenant en compte les performances de ces dernières.

5.5 Mécanisme de retour adaptatif

L'architecture que nous avons proposée comporte un mécanisme de retour adaptatif qui modifie la façon dont sont calculés les scores des méthodes de détection et donc impacte directement le score global obtenu par notre stratégie d'apprentissage ensembliste. Il existe une boucle de rétroaction dans la figure 5.1 qui va du bloc de détection (en jaune) jusqu'aux méthodes de détection (en rouge). Ce mécanisme adaptatif utilise une fenêtre glissante temporelle et prend en compte le graphe de dépendance pour adapter sa taille en fonction des attaques. Dans la suite, nous donnons plus de détails sur cette fenêtre glissante dans laquelle la moyenne des différents scores de détection est calculée. Puis, nous expliquons comment est effectuée la modification de la taille de la fenêtre.

5.5.1 Fenêtre glissante temporelle

Nous définissons ce que nous appelons la fenêtre glissante temporelle. Sa taille est donnée par le sous-bloc mécanisme de retour adaptatif qui se trouve dans le bloc détection. Son objectif est de limiter le nombre de fausses alarmes, qui correspond au taux de faux positifs, c'est-à-dire d'anomalies détectées qui n'en sont pas. Nous cherchons donc à limiter ce phénomène à l'aide d'une fenêtre glissante qui calcule la moyenne du score de détection pour chacune des méthodes à l'intérieur de cette fenêtre. La figure 5.4 montre comment une fenêtre glissante temporelle de taille α lisse le score de détection. Nous n'utilisons pas directement les scores des méthodes de détection

FIGURE 5.4 – Utilisation d'une fenêtre glissante de taille α pour lisser le score de détection

dans notre bloc d'apprentissage ensembliste mais bien la moyenne de ces scores à l'intérieur de la fenêtre glissante. Toutefois, il nous faut légèrement adapter la définition de la fenêtre glissante pour le *process mining*. En effet, ce dernier retourne un score de détection pour un sous-ensemble du jeu de données d'évaluation et non pas un score pour un unique point de données. Ainsi, pour représenter notre fenêtre glissante temporelle de t secondes, nous considérons que la méthode de détection considérée nous renvoie p scores de détection DR, où p est inclu dans l'intervalle $[\![k, n_a]\!]$ avec respectivement k le nombre de sous-ensembles du jeu de données traité par le *process mining* et n_a le nombre d'éléments dans le aième jeu de données. Chacun de ces éléments DR contient un horodatage de début et de fin ainsi qu'un score de détection.

$$DR_p = \{début_p, fin_p, s_p\} \tag{5.24}$$

De cette façon, le nouveau score de détection obtenu en utilisant la fenêtre glissante, pour une méthode donnée, peut être résumé par la formule 5.25. Ce nouveau score représente la moyenne

des scores pris par la ième méthode de détection sur les t dernières secondes.

$$\forall j \in [\![1,p]\!], SW_j = \frac{\sum\limits_{\substack{k=0 \\ fin_j - t \leq fin_k \leq fin_j}}^{p} s_i(fin_k)}{\sum\limits_{\substack{k=0 \\ fin_j - t \leq fin_k \leq fin_j}}^{p} 1} \tag{5.25}$$

Il est toutefois à noter qu'il faut attendre d'avoir au moins un premier score de détection avant que la méthode ne soit intégrée à la stratégie d'apprentissage ensembliste. Ainsi, lorsque le *process mining* utilise un temps de découpe temporelle de 60 secondes, aucun score ne sera disponible pour cette méthode avant au minimum 60 secondes. En effet, nous devons attendre que soit écoulé un intervalle de temps avant de pouvoir accéder au premier score de détection du *process mining*.

5.5.2 Adaptation de la taille de la fenêtre glissante

Le principal objectif de la partie adaptative de la fenêtre glissante temporelle est de pouvoir jouer à la fois sur le taux de faux positifs et sur la réactivité de détection. En effet, en réduisant la taille de la fenêtre glissante, les scores de détection sont moins lissés, ce qui doit réduire le temps de détection. Plus concrètement, la taille de la fenêtre est fixée, pour toutes les sources de données, à une certaine valeur α au lancement de l'architecture. À partir du moment où une anomalie est détectée, la fenêtre est réduite à chaque fois d'une durée β pour les éléments du graphe de dépendance reliés à la source problématique. Il est possible de réduire la taille de cette fenêtre glissante jusqu'à un minimum de α_{min}. Les trois paramètres α, β et α_{min} sont définis par l'utilisateur dans la configuration. L'utilisation de la valeur α_{min} permet d'éviter de trop réduire la taille de notre fenêtre et d'arriver dans une configuration où le taux de faux positifs serait trop important.

Nous prenons comme exemple l'architecture physique décrite par la figure 5.2 et son graphe de dépendance associé de la figure 5.3. Nous y trouvons cinq noeuds correspondant aux différentes sources de données. Le premier noeud correspond au réseau étendu (WAN) et ne possède qu'un seul successeur, qui est l'objet connecté 1. Celui-ci n'a également qu'un unique successeur, qui est le réseau local (LAN). En revanche, nous pouvons constater que le noeud du réseau local (LAN) a deux successeurs possibles, les objets connectés 2 et 3. Le graphe de dépendance nous permet de mettre correctement à jour les tailles des fenêtres glissantes temporelles au sein du mécanisme adaptatif. Admettons qu'une attaque soit détectée sur l'objet connecté 1, comme le montre la figure 5.5, c'est-à-dire que le score global est supérieur à S_{seuil}. Nous allons donc réduire d'un temps β la taille de la fenêtre glissante, qui initialement était d'une longueur α. Évidemment, comme nous l'avons dit précédemment, cette réduction de la taille de la fenêtre ne pourra se faire que si la valeur $\alpha - \beta$ est supérieure à α_{min}. Dans cet exemple, nous appliquons la réduction de la taille de la fenêtre glissante à l'ensemble des éléments du graphe de dépendance arrivant après le noeud à l'origine du problème. Toutefois, comme nous avons pu l'évoquer, il est tout à fait envisageable de l'appliquer uniquement aux successeurs directs ou bien encore aux successeurs se trouvant à une distance prédéfinie dans le graphe de dépendance. Ce mécanisme, qui utilise une fenêtre glissante temporelle, cherche à améliorer les performances de détection des attaques en plusieurs étapes. Il nous permet à la fois de limiter le taux de faux positifs et d'augmenter la réactivité de la détection des éléments critiques d'un système dans le cas où les premières étapes d'une attaque auraient été repérées. La figure 5.6 montre de quelle façon la fenêtre glissante s'adapte à la détection d'une anomalie dans le réseau étendu (WAN).

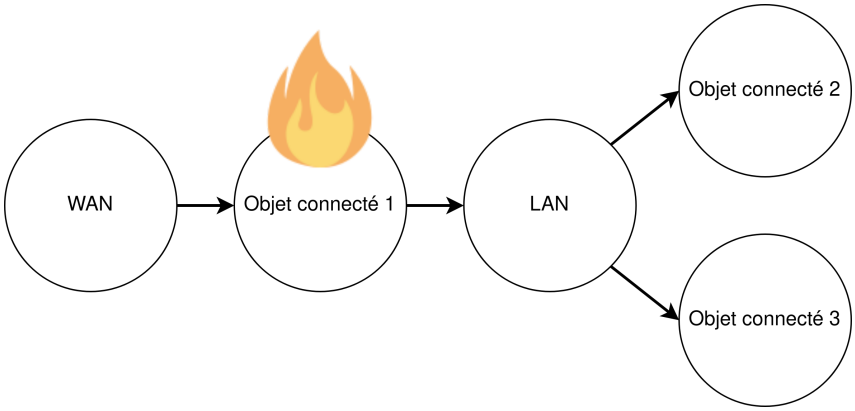


FIGURE 5.5 – Localisation de l'attaque dans le graphe de dépendance

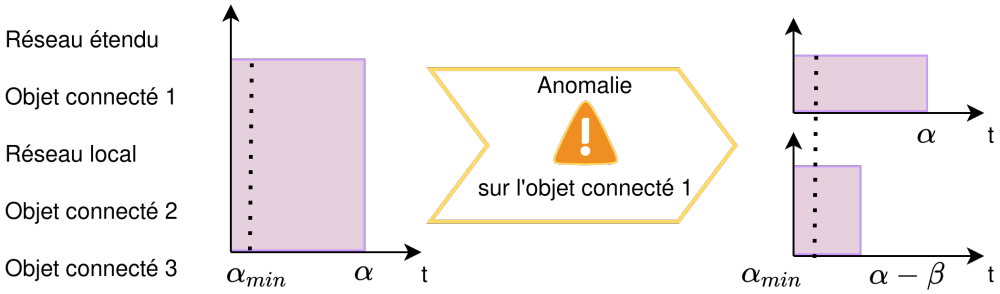


FIGURE 5.6 – Évolution de la taille de la fenêtre glissante en cas de détection d'attaque

5.6 Synthèse

Dans ce chapitre, nous présentons une adaptation ensembliste de notre solution de détection des anomalies pour les systèmes IoT hétérogènes. L'objectif est de tirer parti des performances de plusieurs méthodes de détection exécutées en parallèle. Certaines méthodes semblent plus adaptées à des types spécifiques de données, ce qui peut poser problème si l'on privilégie une unique méthode avec des données hétérogènes issues de systèmes IoT. Les attaques elles-mêmes peuvent profiter de cette hétérogénéité pour se camoufler. Nous avons tout d'abord justifié le choix des méthodes de détection que nous avons considérées pour notre solution ensembliste. Puis, nous avons présenté quatre stratégies d'agrégation des scores de détection (EL_{uni},EL_{exa},EL_{exp} et EL_{max}) permettant sa mise en oeuvre. La première stratégie donne le même poids à l'ensemble des méthodes de détection, tandis que les deuxième et troisième stratégies favorisent les méthodes ayant obtenu les meilleures exactitudes. La dernière stratégie se veut très réactive, mais peut entraîner un taux important de faux positifs. Aussi, nous avons proposé un mécanisme de retour adaptatif, dont l'objectif est d'accroître la réactivité tout en réduisant ce taux de faux positifs. Dans le chapitre suivant, nous analysons les performances que nous avons obtenues avec cette méthode ensembliste dans le cas de systèmes IoT hétérogènes.

Chapitre 6

Évaluation de l'approche ensembliste

Sommaire

6.1 Introduction 94
6.2 Description des jeux de données 95
6.3 Évaluation des performances de l'architecture 96
6.3.1 Construction du graphe de dépendances 97
6.3.2 Apprentissage ensembliste 98
6.3.3 Mécanisme de retour adaptatif 100
6.4 Influence des perturbations sur les performances de détection . . . 103
6.4.1 Données bruitées 103
6.4.2 Perte de données 105
6.5 Synthèse 108

6.1 Introduction

Dans ce chapitre, nous travaillons sur l'évaluation des performances de détection de notre solution reposant sur l'apprentissage ensembliste. Tout d'abord, nous décrivons les jeux de données qui nous ont servi lors de notre évaluation. Ils proviennent d'un véhicule connecté mais ne correspondent pas à ceux que nous avons pu décrire précédemment dans la section 4.2 du chapitre 4. Nous considérons deux scénarios dans ce chapitre : un fonctionnement normal et une attaque en plusieurs étapes qui passe par la recherche d'une faille à partir du réseau jusqu'à l'exécution d'une attaque de type déni de service sur le véhicule. Après avoir décrit les données, nous voyons comment a été construit le graphe de dépendances de ce système IoT. Ensuite, nous étudions les performances de notre architecture ensembliste utilisant quatre stratégies d'agrégation des scores de détection. Lors des évaluations de performances, nous abordons, tout d'abord, les résultats obtenus par notre solution de détection en fonction de la taille de la fenêtre glissante temporelle. Puis, nous observons l'impact du mécanisme de retour adaptatif sur la réactivité et la pertinence de la détection. Finalement, nous étudions l'influence de deux types de perturbations sur notre système de détection d'anomalies. Dans un premier temps, nous utilisons des jeux de données comprenant plusieurs niveaux de bruit. L'objectif est de montrer la robustesse de l'apprentissage ensembliste par rapport au bruit pouvant provenir des capteurs ou du média de transmission. Dans un second temps, nous nous sommes focalisés sur les pertes de données pouvant là aussi intervenir dans le cas de systèmes réels.

6.2 Description des jeux de données

Dans cette section, nous décrivons le scénario de l'attaque considérée ainsi que les différents jeux de données que nous utilisons dans le reste de ce chapitre. Le scénario que nous avons retenu se concentre exclusivement sur les véhicules connectés avec de nombreux capteurs et appareils installés à l'intérieur de celui-ci permettant d'apporter une aide à la conduite pour le conducteur et d'améliorer l'expérience des passagers. Ces données, contiennent une attaque en trois étapes. La sécurité des véhicules connectés est d'une importance capitale car une attaque peut avoir des conséquences graves pour les passagers. En particulier, les attaques qui cherchent à corrompre le bus de transmission CAN peuvent empêcher la communication entre les différents éléments du système. Le scénario qui a été retenu avec notre partenaire consiste en une attaque du réseau local de commande (CAN) où l'attaquant va surcharger les capacités du bus de transmission et donc empêcher la remontée des informations. Avant de pouvoir effectuer cette partie de l'attaque, il faut que l'attaquant accède à l'unité de bord en commençant par un scan des ports afin de trouver une vulnérabilité lui permettant d'atteindre le système et d'y installer un fichier malveillant contenant le script qui lance le déni de service sur le bus CAN. Pour observer cette attaque, nous avions à notre disposition trois sources de données. La figure 6.1 montre où sont récupérées les informations pour chacune d'entre elles sur le système physique. Dans la première

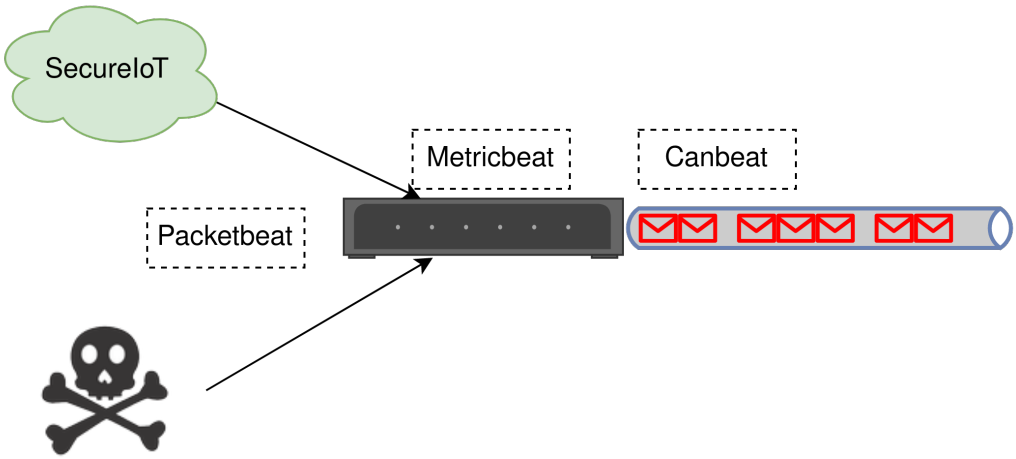


FIGURE 6.1 – Figure illustrant le positionnement des trois sources de données sur le véhicule connecté

source de données, intitulée *Packetbeat*, nous trouvons les données réseau du bus CAN du véhicule contenant les adresses IP de l'expéditeur et du destinataire, le nombre de paquets ainsi que le port utilisé lors des échanges d'informations entre les différentes entités du système IoT. Ces éléments sont synthétisés dans le tableau 6.1 dans lequel nous avons trié les données de la même façon que dans la section 4.2 du chapitre 4. Nous retrouvons la distinction entre les données à valeurs discrètes (VD), bornées ou non, les données à valeurs numériques continues (VN), bornées ou non, les données booléennes (VB), les données textuelles (VT), les données catégorielles (VC) ainsi que les données d'agrégation ou de regroupement de paramètres (RP). Nous notons également la variabilité de ces données avec des changements de valeurs continues (CC), liés à un événement (CE) ou tout simplement sans changement, donc avec une valeur fixe (VF).

La seconde source de données, *Metrcibeat*, nous renseigne sur l'utilisation, physique et logique, du matériel. Elle nous indique, entre autres, l'usage du CPU, de la mémoire vive (RAM) mais

	VD	VN	VB	VC	VT	RP	VF	CC	CE
Protocole de transport utilisé				X				X	
Protocole réseau utilisé				X				X	
Nombre d'octets échangés	X							X	
Nombre de paquets échangés	X							X	
Adresse MAC source					X			X	
Adresse MAC destination					X			X	
Port source				X				X	
Port destination				X				X	
Adresse IP source					X			X	
Adresse IP destination					X			X	
Flag de fin			X						X

TABLE 6.1 – Tableau explicatif des données relatives au véhicule connecté et provenant du capteur *Packetbeat*

elle décrit également comment sont utilisés les disques et quels sont les processus en cours. Le nombre d'attributs est très important et ils ne sont pas tous pertinents. Par exemple, nous trouvons des attributs qui ne changeront jamais, comme le système d'exploitation installé sur le système, et d'autres qui sont difficilement interprétables, comme le nombre de processus en sommeil. Ainsi contrairement aux autres sources de données, nous ne listons pas l'ensemble des attributs disponibles. De plus, il est à noter que dans le cadre de nos expériences, nous avons filtré les données. En effet, pour détecter la seconde phase de l'attaque, qui correspond à l'envoi et l'installation d'un fichier malveillant, nous avons considéré exclusivement les données relatives à l'utilisation en écriture des disques. Ce filtrage était nécessaire dû au nombre très important d'attributs présents dans ce jeu de données, afin d'affiner la détection pour cibler des phases particulières de l'attaque. Finalement, le dernier jeu de données à notre disposition nous provient de la source *Canbeat*. Celle-ci nous transmet les informations provenant du bus CAN du véhicule telles que le temps de réponse, le nombre d'erreurs observées ou bien encore le pourcentage d'utilisation du bus réseau. Le tableau 6.2 donne les différents attributs de ce jeu de données.

6.3 Évaluation des performances de l'architecture

Dans cette section, nous présentons les résultats que nous avons obtenus à l'aide de plusieurs séries d'expériences. Ainsi, dans un premier temps, nous cherchons à générer le graphe de dépendance d'un véhicule connecté. Puis, nous analysons l'influence des différentes stratégies

	VD	VN	VB	VC	VT	RP	VF	CC	CE
Charge du bus CAN	X(B)							X	
Contrôle de redondance invalide	X								X
Système d'évaluation invalide	X								X
Mauvaise taille des données	X								X
Erreur de transmission	X								X

TABLE 6.2 – Tableau explicatif des données relatives au véhicule connecté et provenant du capteur *Canbeat*

d'agrégation sur le temps de détection et sur l'exactitude. Ensuite, dans une troisième section, nous étudions l'impact du mécanisme de retour adaptatif sur la réactivité et les performances de détection.

6.3.1 Construction du graphe de dépendances

Dans le but de créer le graphe de dépendances d'un véhicule connecté, nous avons calculé le coefficient de corrélation croisée entre les différentes sources de données pour chaque méthode de détection. Nous avons également considéré une méthode de détection parfaite (PC) dans un but illustratif. La figure 6.2 donne les résultats obtenus avec les sources de données (*packetbeat, metricbeat, canbeat*) de notre véhicule connecté.

		Metricbeat					Canbeat				
		EE	IF	LOF	OCSVM	PC	EE	IF	LOF	OCSVM	PC
Packetbeat	EE	0.124	0.049	0.052	0.124	0.124	0.062	0.173	0.523	0.522	0.522
	IF	0.061	0.108	0.111	0.061	0.060	0.218	0.203	0.666	0.665	0.668
	LOF	0.103	0.111	0.102	0.103	0.103	0.252	0.211	0.761	0.760	0.761
	OCSVM	0.307	0.083	0.089	0.307	0.307	0.050	0.125	0.235	0.236	0.235
	PC	0.088	0.071	0.085	0.088	0.088	0.037	0.174	0.543	0.544	0.542
Metricbeat	EE	-	-	-	-	-	0.001	0.225	0.045	0.045	0.045
	IF	-	-	-	-	-	0.126	0.095	-0.022	-0.020	-0.020
	LOF	-	-	-	-	-	0.148	0.115	0.010	0.012	0.011
	OCSVM	-	-	-	-	-	0.001	0.225	0.045	0.045	0.045
	PC	-	-	-	-	-	-0.002	0.225	0.047	0.047	0.047

FIGURE 6.2 – Valeur maximale de corrélation croisée observées pour l'ensemble des méthodes de détection lors des différentes phases de l'attaque

Les coefficients de corrélation ne dépassent pas 0,307 entre les sources de données *packetbeat* et *metricbeat* ce qui ne nous permet pas de découvrir automatiquement le lien entre elles. Ce coefficient est plus faible entre le *metricbeat* et le *canbeat* avec une valeur maximum de 0,225. Ces valeurs de corrélation s'expliquent par la faible durée de la seconde étape de l'attaque visible dans les données du *metricbeat* par rapport aux autres étapes. En effet, l'envoie et l'installation d'un fichier malveillant dure environ 2 secondes, tandis que la première et la troisième étape de

l'attaque, pour lesquelles nous avons utilisé le *packetbeat* et le *canbeat*, durent plusieurs minutes. Il n'est donc pas surprenant d'éprouver des difficultés à trouver un lien de corrélation entre la première et la seconde étape ainsi qu'entre la seconde et la troisième étape. En revanche, notre méthode de détection automatique des dépendances permet de mettre en exergue un lien entre le *packetbeat* et le *canbeat* avec un coefficient de corrélation de Pearson atteignant 0,761.

En appliquant le calcul des coefficients de Pearson dans ce cas précis, nous remarquons que les liens mis en avant ne sont pas nécessairement pertinents. Dans la suite de ce chapitre, nous utilisons le graphe de dépendances de la figure 6.3 que nous avons déduit de l'architecture physique du véhicule connecté La figure détaille les dépendances existant entre les trois sources que nous avons considérées pour les véhicules connectés. Ainsi, nous trouvons la connexion à

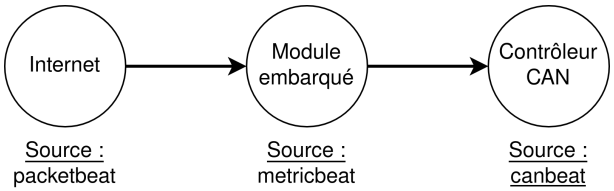


FIGURE 6.3 – Graphe de dépendance final pour le véhicule connecté

Internet avec le réseau étendu (WAN) connecté au module embarqué, qui est l'objet connecté du réseau, lui-même relié au contrôleur CAN. Les noms des capteurs correspondant aux différentes sources de données sont respectivement le packetbeat, le metricbeat et le canbeat.

Dans cet exemple, nous avons associé la recherche automatique de liens entre les différentes sources de données d'un système avec le calcul du coefficient de corrélation de Pearson avec une recherche manuelle de liens. En effet, l'utilisation d'un seuil C_{seuil} de 0,7, permettant de justifier de la présence d'un rapprochement entre deux sources, ne nous permet pas de mettre en évidence le lien entre le réseau étendu et le module embarqué. Il n'est pas non plus possible de détecter le lien entre le module embarqué et le contrôleur CAN. Ainsi, dans ce cas nous devons spécifier manuellement que ces liens existent. Dans les cas où les jeux de données ne permettent pas d'établir correctement l'existence des liens, au moins deux choix s'offrent à nous. Le premier est de créer le graphe de dépendances manuellement, c'est-à-dire de laisser à un utilisateur la possibilité de renseigner l'intégralité des liens existants. Le second est d'injecter une attaque dans le système pour suivre sa propagation, ce qui permet d'augmenter le coefficient de corrélation de Pearson lors de la propagation de l'attaque et donc de pouvoir repérer les liens empruntés par celle-ci. L'utilisation du coefficient de corrélation nous permet d'automatiser la détection des liens existant entre les sources de données. Toutefois, il est nécessaire de valider la présence de ces liens et d'ajouter manuellement ceux manquants.

6.3.2 Apprentissage ensembliste

Dans une première série d'expériences, nous nous sommes intéressés aux performances des approches d'apprentissage ensembliste intégrées dans l'architecture. Notre objectif est de quantifier les apports de cette approche par rapport à l'utilisation d'une unique méthode de détection. Nous avons comparé les résultats de l'apprentissage ensembliste avec les méthodes de l'*eliptic envelope*, du *one class support vector machine*, du *local outlier factor*, de l'*isolation forest* et du *process mining*. Nous avons réalisé ces expériences à l'aide de notre prototype, que nous décrivons plus en détail dans le chapitre 7, et de trois sources de données (le *packetbeat*, le *metricbeat* et le *canbeat*) provenant de la simulation d'un véhicule connecté. L'idée derrière l'utilisation de ces

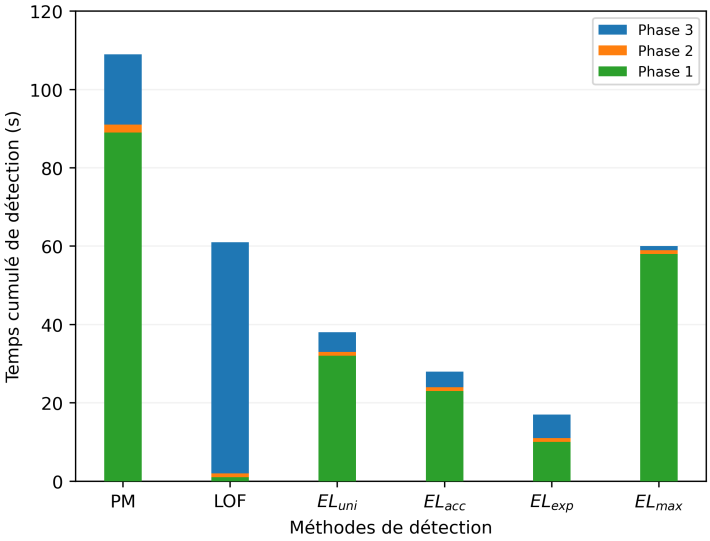


FIGURE 6.4 – Comparaison du temps de détection cumulé obtenu pour les trois phases de l'attaque par différentes méthodes de détection (*process mining*, *local outlier factor* et les quatre méthodes d'agrégation des scores utilisées par l'apprentissage ensembliste)

trois sources de données hétérogènes est de pouvoir suivre la propagation d'une attaque complexe composée de plusieurs étapes. Nous nous intéressons au temps de détection, et pour cela, nous mesurons le temps cumulatif de détection ainsi que l'exactitude de notre solution dans sa détection des trois phases de l'attaque. Nous faisons cela pour toutes les méthodes de détection et pour les différentes stratégies d'agrégation des scores de l'apprentissage ensembliste, détaillées dans le chapitre 5. Il est à noter que les méthodes de détection ne sont pas directement appliquées par les objets connectés mais bien par un système externe. Celui-ci reçoit les données de la part des capteurs et possède de plus grandes capacités de calcul que les objets connectés, ce qui lui permet de traiter efficacement les informations. Il est, en effet, tout à fait possible de considérer l'utilisation d'un tel système externe en bordure (*network edge*) pour permettre des traitements lourds sur les données que les systèmes IoT ne peuvent pas effectuer eux-mêmes.

La figure 6.4 représente le temps cumulé de détection des trois phases de détection de l'attaque sur le véhicule connecté. Nous avons utilisé une fenêtre temporelle glissante de 60 secondes. La détection d'anomalies est effectuée par les quatre stratégies d'agrégation des scores, notées $\mathrm{EL}_{\mathrm{uni}}$, $\mathrm{EL}_{\mathrm{exa}}$, $\mathrm{EL}_{\mathrm{exp}}$ et $\mathrm{EL}_{\mathrm{max}}$ déjà définies dans la section 5.4. À côté d'elles, nous avons affiché les deux méthodes de détection ayant donné les résultats les plus extrêmes, au niveau du temps de détection : le *process mining*, qui est la moins réactive, et le *local outlier factor*, qui détecte l'anomalie des deux premières phases le plus rapidement parmi l'ensemble des méthodes. De plus, ces deux méthodes ont été capables de repérer toutes les phases de l'attaque contrairement à d'autres méthodes comme l'*isolation forest*, par exemple, qui n'a pas pu détecter la seconde étape. Lorsque l'on observe les résultats obtenus, on remarque que la stratégie EL_{exp} est la méthode la plus réactive pour détecter les trois phases de l'attaque avec un temps cumulé de détection de 18 secondes. Les autres stratégies, c'est-à-dire $\mathrm{EL}_{\mathrm{uni}}$, $\mathrm{EL}_{\mathrm{exa}}$ et $\mathrm{EL}_{\mathrm{max}}$, obtiennent respectivement un temps de détection cumulé de 38 secondes, 28 secondes et 60 secondes. Ces temps restent meilleurs que ceux obtenus par les autres algorithmes de détection, avec le *local outlier factor* qui met environ 60 secondes à détecter les trois phases de l'attaque, et le *process mining* qui lui met 109 secondes. Les performances de certaines des stratégies d'agrégation de

Méthodes	Phase 1	Phase 2	Phase 3
PM	82.61%	84.78%	97.78%
LOF	86.20%	13.51%	87.74%
EL_{uni}	56.05%	85.93%	99.15%
EL_{acc}	70.91%	85.93%	99.37%
EL_{exp}	97.24%	85.92%	98.94%
EL_{max}	60.08%	84.01%	99.79%

TABLE 6.3 – Exactitudes obtenues avec les différentes méthodes de détection pour les trois phases de l'attaque considérée

l'apprentissage ensembliste sont décevantes. En effet, il est difficile de détecter correctement la première phase de l'attaque à cause de la nature des données qui comportent beaucoup de bruit. Cela a pour conséquence de perturber certaines des méthodes de détection d'anomalies. Seules les méthodes *one class support vector machine* et *local outlier factor* obtiennent un temps de détection de l'ordre de la seconde pour cette étape.

La rapidité de détection ne peut pas être la seule métrique d'évaluation des performances d'une solution de détection. C'est pourquoi, nous observons également les valeurs de l'exactitude obtenues par les différentes méthodes. Le tableau 6.3 donne les valeurs d'exactitude des méthodes pour les différentes sources de données et nous pouvons voir que cette valeur peut grandement varier en fonction de la source de données. Par exemple, l'algorithme *local outlier factor* obtient des performances intéressantes pour la phase 1 de l'attaque, avec une exactitude de 86,20%, tandis qu'elles sont significativement dégradées lors de la seconde phase, avec une exactitude de 13,51%. Le *process mining*, quant à lui, conserve une exactitude relativement élevée pour les trois phases. En effet, pour les trois phases de l'attaque, le *process mining* obtient une exactitude supérieure à 80%. Ses performances sont même excellentes dans la détection de la troisième phase de l'attaque avec une exactitude de 97,78%. Mais comme nous avons pu le voir précédemment, l'une des caractéristiques de cette méthode est son important temps de détection. Afin de proposer une évaluation plus pertinente, nous considérons à la fois le temps de détection et l'exactitude des méthodes de détection et des stratégies d'agrégation de l'apprentissage ensembliste. Lorsque nous prenons en compte ces deux métriques, nous remarquons que c'est la stratégie EL_{exp} qui obtient les résultats les plus intéressants. Elle consiste à grandement réduire l'influence des méthodes de détection peu performantes, obtient le temps de détection cumulé le plus bas de nos expériences en plus d'avoir une exactitude élevée pour toutes les phases de l'attaque. En effet, elle obtient une exactitude de 97% pour la première phase, d'environ 85% pour la seconde et de 99% pour la dernière phase de l'attaque.

6.3.3 Mécanisme de retour adaptatif

Dans une seconde série d'expériences, nous nous sommes intéressés à l'évaluation des performances de notre solution de détection lorsque celle-ci utilise le mécanisme de retour adaptatif. Nous avons utilisé les mêmes jeux de données que ceux décrits dans la section 6.2. Nous comparons les résultats de détection obtenus par les deux méthodes nous servant de base de comparaison méthodes (le *process mining* et l'*isolation forest*) avec les quatre stratégies utilisées par l'apprentissage ensembliste et décrites dans la section 5.4. Notre objectif est de montrer l'impact du changement de la taille de la fenêtre glissante temporelle sur les performances de la détection

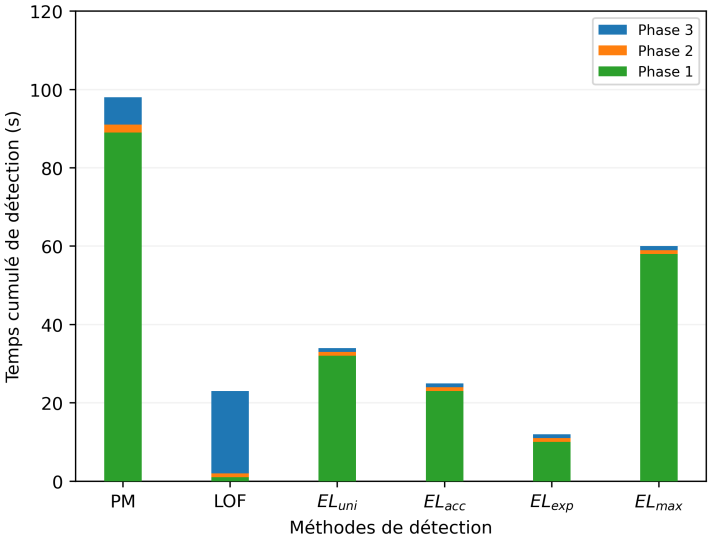


FIGURE 6.5 – Comparaison du temps de détection cumulé pour les trois phases de l'attaque avec l'utilisation du mécanisme de retour adaptatif ($\alpha = 60s$ et $\beta = 40s$

des différentes phases de l'attaque. Pour cela, nous utilisons, dans ces expériences, une taille de fenêtre glissante initiale de 60 secondes, qui correspond au paramètre α, ainsi que la valeur par laquelle nous réduisons cette fenêtre en cas de détection, qui est de 40 secondes ($\beta = 40$). L'influence de la taille initiale de la fenêtre glissante temporelle est également évaluée dans nos expériences.

La figure 6.5 montre le temps cumulé de détection qui est obtenu pour les trois phases de l'attaque pour les différentes stratégies d'agrégation des scores. Les résultats du *process mining* et du *local outlier factor* nous servent de base de comparaison entre les stratégies d'agrégation et les méthodes de détection plus classiques. Nous remarquons que le mécanisme de retour adaptatif permet de réduire le temps total cumulé de détection dans tous les cas. En effet, le *process mining* voit son temps de détection passer de 109 secondes, sans adaptation, à environ 99 secondes en appliquant le retour adaptatif. De la même façon, nous observons, sur la figure 6.5, cette réduction de temps pour les différentes stratégies qui passe de 18 secondes à 11 secondes pour la stratégie EL_{exp}. Celle qui profite le moins de ce mécanisme est EL_{max} qui obtient le même temps cumulé de détection d'environ 60 secondes dans les deux cas, c'est-à-dire en l'absence du mécanisme de retour adaptatif et en sa présence. Nous revenons plus tard dans cette section sur les conséquences d'une trop grande réduction de taille pour notre fenêtre glissante.

Initialement, notre but était d'étudier l'influence de la réduction de la fenêtre, à la fois sur le temps de détection et sur l'exactitude. Ainsi, après avoir montré que la modification de la taille de la fenêtre glissante temporelle a un impact sur le temps de détection, nous avons continué nos expériences en faisant varier la taille de cette fenêtre entre 10 et 60 secondes par pas de 10 secondes. Nous obtenons les deux figures 6.6 et 6.7 qui correspondent aux phases pour lesquelles l'influence de la taille de la fenêtre est la plus importante à la fois pour le temps de détection et pour l'exactitude. Un phénomène identique peut être observé pour les autres phases mais de façon moins marquée. La figure 6.6 concerne la troisième phase de l'attaque, le déni de service sur le bus CAN (CANDoS), et correspond à la phase où le mécanisme adaptatif a la plus grande influence sur le temps de détection. Les valeurs d'exactitudes des différentes méthodes de détection pour cette étape de l'attaque restent quasiment identiques, c'est pourquoi nous ne l'avons pas utilisée

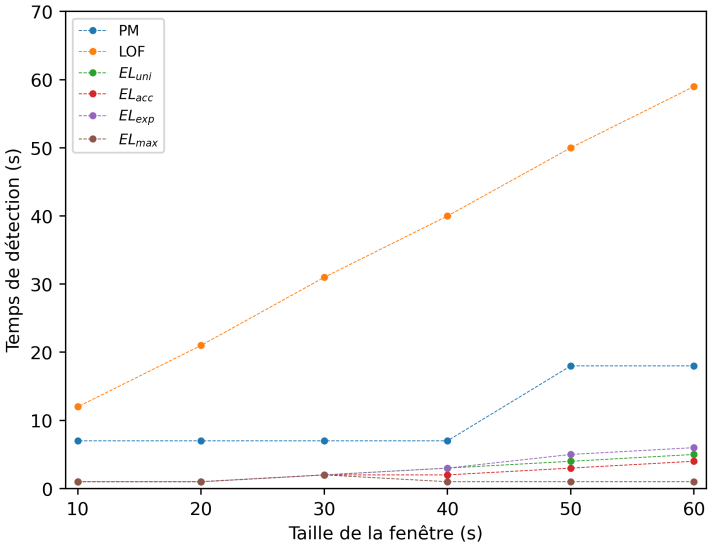


FIGURE 6.6 – Influence de la fenêtre glissante temporelle sur le temps de détection pour les différentes méthodes de détection pour la première phase de l'attaque (variation de la fenêtre de 10 à 60 secondes)

pour illustrer l'évolution de l'exactitude en fonction de la taille de la fenêtre glissante. Nous remarquons que plus la taille de la fenêtre glissante est réduite plus nous trouvons un temps de détection faible. Ce constat est fortement visible sur les méthodes de détection individuelle où le *process mining* obtient un temps de détection aux alentours de 8 secondes, pour cette phase particulière de l'attaque. La méthode du *local outlier factor* quant à elle détecte l'anomalie de cette troisième phase en un peu plus de 10 secondes. Cette réduction du délai de détection est moins marquée sur les méthodes utilisant des stratégies d'agrégation des scores, même si elle reste présente. Il faut toutefois prendre en compte que le temps de détection des stratégies utilisées par l'apprentissage ensembliste était déjà inférieur aux autres méthodes de détection et que donc la marge d'amélioration était moins importante. La figure 6.7 met en avant l'évolution de l'exactitude, pour différentes tailles de fenêtres glissantes lors de la première phase de l'attaque, qui correspond au scan des ports. C'est pour cette phase que la variation de la taille de la fenêtre a le plus d'impact sur l'exactitude et c'est donc pourquoi nous avons choisi de la représenter. Nous n'avons pas utilisé cette phase de l'attaque lors de notre évaluation du temps de détection en fonction de la taille de la fenêtre. Précédemment, la présence de faux positifs quelques secondes avant le début de l'attaque favorisait les tailles de fenêtre englobant les faux positifs et le début de l'attaque. Les fenêtres plus petites prenaient du retard à cause du phénomène de lissage de la fenêtre glissante. Nous observons, sur la figure, que l'exactitude diminue, pour toutes les méthodes, lorsque nous diminuons la fenêtre glissante temporelle. Cependant, cette diminution ne se fait pas de façon identique pour toutes les méthodes. En effet, nous remarquons que cette diminution est limitée pour la stratégie EL_{exp} tant que nous considérons une taille d'au moins 30 secondes. Nous passons donc d'une exactitude de 0,97 pour une taille de fenêtre de 60 secondes à une exactitude de 0,96 pour une fenêtre de 30 secondes. Ces différentes expériences nous permettent de mettre en avant les avantages de notre solution utilisant l'apprentissage ensembliste couplé à un mécanisme de retour adaptatif. Nous mesurons, pour différentes tailles de fenêtres, l'amélioration du temps de détection avec l'exactitude correspondante. Nous montrons également

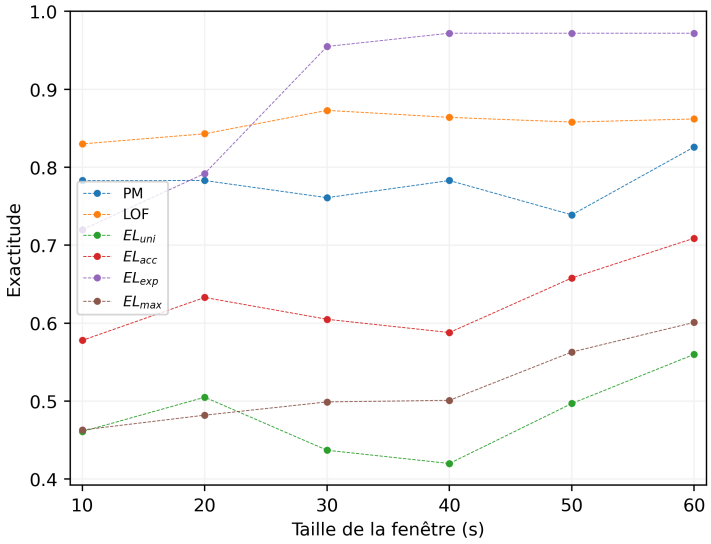


FIGURE 6.7 – Influence de la fenêtre glissante temporelle sur l'exactitude pour les différentes méthodes de détection pour la troisième phase de l'attaque (variation de la fenêtre de 10 à 60 secondes)

qu'il existe une limite basse, pour la taille de fenêtre, au-delà de laquelle les performances de détection se détériorent de façon importante. Par exemple, pour la méthode utilisant la stratégie EL_{exp}, l'exactitude diminue de 24%, passant de 96% à 72%, lorsque nous considérons une taille de fenêtre de 10 secondes à la place d'une de 30 secondes.

6.4 Influence des perturbations sur les performances de détection

Dans cette section, nous étudions l'influence de deux types de perturbations sur la détection d'anomalies. Dans un premier temps, nous regardons comment l'introduction de bruit dans les données modifie les performances de notre solution de détection. Puis, nous étudions la robustesse de la solution pouvant affecter un système réel. Nous ne considérons pas le jeu de données du *packetbeat* pour effectuer la détection du scan des ports. En effet, bien que l'on trouve de multiples flux de réseaux rendant inopérant le *process mining*, ce qui justifierait l'utilisation d'autres méthodes de détection, la construction même du jeu de données complique la bonne détection des anomalies. Les attributs des différents flux réseaux y sont répétés tout au long du jeu de données. À cause de ce phénomène, nous trouvons des données générées lors du fonctionnement normal du système qui sont répétées durant l'attaque. L'inverse est également vrai, nous observons des flux provenant de l'attaque sur les ports après la fin de cette étape.

6.4.1 Données bruitées

Il est tout à fait possible, dans le cas de systèmes IoT réels, que la remontée des informations soit perturbée ou bien tout simplement que le manque de précision des capteurs pose problème. Pour tester la robustesse des stratégies d'agrégation pour l'apprentissage ensembliste, nous les avons confrontées à des jeux de données d'évaluation comportant différents niveaux de bruit. Ce bruit gaussien a été introduit de la même façon que dans le chapitre 4 lorsque nous avons évalué le comportement du *process mining* face à un jeu de données bruitées. Dans le cas présent, nous

avons fait cette analyse sur les données provenant du bus CAN du véhicule (*canbeat*) ainsi que de deux sous-ensembles fournis par le système de bord (*metricbeat*) sur des données matérielles. Le premier de ces sous-ensembles comporte des informations sur les écritures disques, tandis que le second remonte des éléments relatifs à l'utilisation CPU. Les données de l'utilisation CPU nous ont servi à détecter, entre autres, le scan des ports, qui est la première étape de l'attaque. Sur la figure 6.8, nous observons l'évolution du coefficient de Matthews (MCC), qui est la métrique qui prend en compte l'ensemble de la matrice de confusion, ainsi que l'exactitude pour les données sur l'utilisation CPU. Le bruit a une influence limitée, on n'observe pas de changement drastique de la valeur du coefficient de Matthews en moyenne qui reste proche de 0,6 pour un bruit allant de 0 à 50%. En revanche, selon les cas, on remarque une plus grande variation de cette métrique lorsque le bruit augmente, ce qui est également le cas pour l'exactitude. Ainsi, bien qu'en moyenne la détection reste identique, l'introduction de bruit permet parfois de détecter des anomalies dont le score de détection était proche, sans le dépasser, du seuil de détection. À l'inverse, ce bruit peut également faire passer une anomalie pour un point de données normal. Plus la force du bruit est importante plus ces deux phénomènes sont marqués, et donc plus les performances de détection peuvent varier. Les quatre stratégies d'agrégation des scores utilisées par l'apprentissage ensembliste donnent des résultats assez similaires. La stratégie utilisant un poids exponentiel $\mathrm{El}_{\mathrm{exp}}$ est légèrement inférieure aux autres mais semble très stable, c'est-à-dire que jusqu'à un bruit de 40% nous n'observons pas de variation, en moyenne, au niveau du coefficient de Matthews. Pour l'exactitude, nous ne détectons aucune variation, en moyenne, sur l'ensemble du graphe et donc celle-ci n'est pas affectée par un bruit inférieur à 50%.

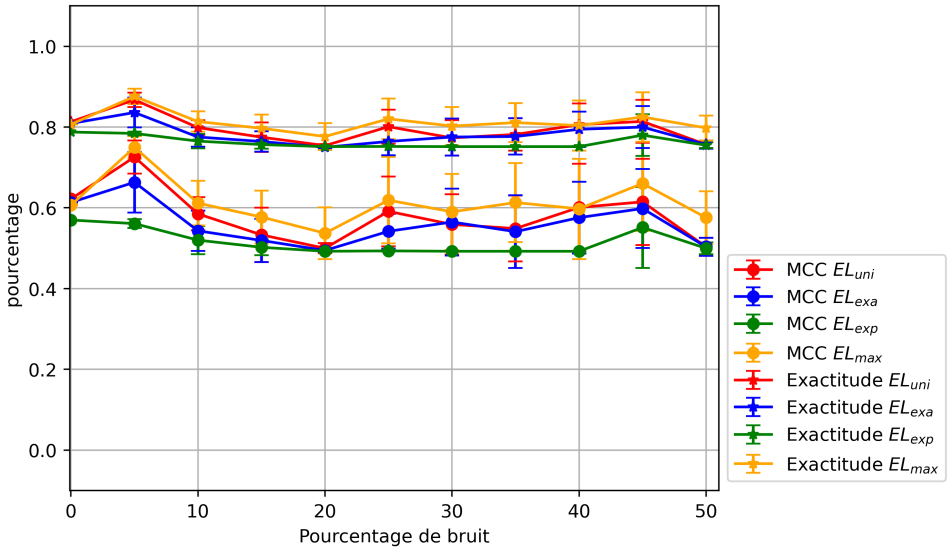


FIGURE 6.8 – Influence du bruit dans la détection d'anomalies lors de l'utilisation de stratégies d'agrégation sur les données du capteur metricbeat (cpu) avec une fenêtre de 60 secondes

Dans le cas des données relatives aux écritures disques, que l'on peut voir sur la figure 6.9, la dégradation des valeurs obtenues par le coefficient de Matthews peut s'expliquer par la nature de la seconde étape de l'attaque qui consiste à envoyer un fichier malveillant au système. Ce transfert est rapide, moins de 2 secondes, et n'est pas la seule action qui va toucher aux écritures disques. L'introduction de bruit va immédiatement faire diminuer la valeur du coefficient de Matthews avec l'apparition d'un grand nombre de faux positifs. Il est même tout à fait envisageable de rencontrer des faux négatifs, si le bruit diminue la valeur de certains attributs car, dans

ces jeux de données, la frontière entre les données normales et anormales est ténue. En ce qui concerne la seconde métrique d'évaluation, nous nous trouvons dans un cas typique du paradoxe de l'exactitude visible sur les jeux de données déséquilibrés. En effet, dans cette configuration, celle-ci est toujours comprise entre 80% et 100%. La proportion des anomalies étant très faible (inférieur à 1%), une méthode de détection ne détectant aucune de ces anomalies peut obtenir une exactitude supérieure à 99%.

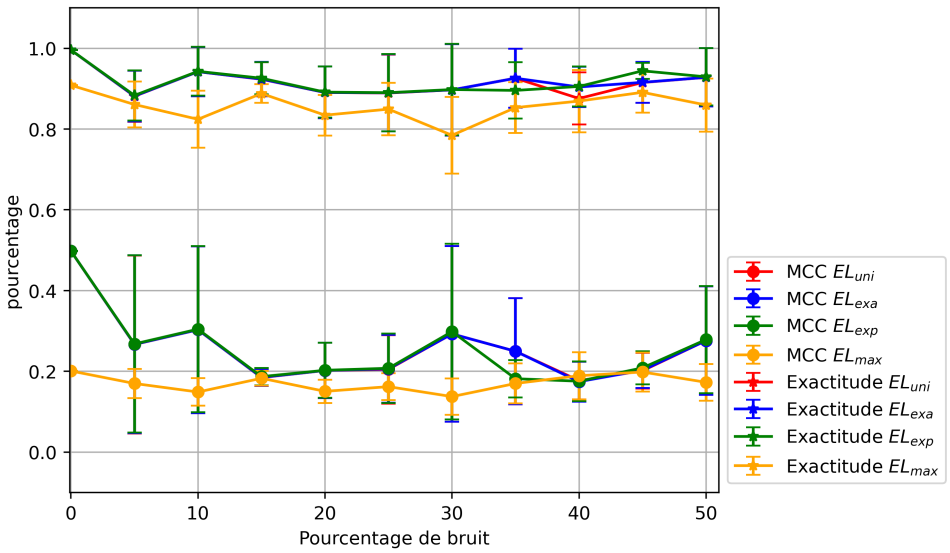


FIGURE 6.9 – Influence du bruit dans la détection d'anomalies lors de l'utilisation de stratégies d'agrégation sur les données du capteur metricbeat (diskio) avec une fenêtre de 60 secondes

Finalement, on peut remarquer sur la figure 6.10 que la détection de la troisième phase de l'attaque, le déni de service sur le bus CAN, obtient de très bonnes performances avec un coefficient de Matthews toujours au-dessus de 80% et une exactitude proche de 95% pour des valeurs de bruits comprises entre 0 et 50%.

La dégradation des performances reste minime bien que l'introduction de bruit soit à l'origine d'une légère dégradation de la détection de cette phase de l'attaque. En effet, la stratégie d'agrégation utilisant un poids exponentiel, qui est la stratégie obtenant les meilleurs résultats de détection, voit son coefficient de Matthews passer de 100% en l'absence de bruit à une moyenne de 98% avec un bruit de 50%. Le déni de service sur le bus CAN est l'étape de l'attaque la plus facilement mise en évidence avec l'ensemble des méthodes de détection. De plus, les données obtenues pour le comportement normal et anormal sont très différentes avec, par exemple, une charge du bus CAN allant respectivement d'environ 10% à plus de 90%. Il n'est donc pas surprenant d'obtenir ces résultats.

6.4.2 Perte de données

Après avoir introduit du bruit dans les jeux de données, nous étudions le comportement des différentes stratégies d'agrégation en cas de perte de données. En effet, là où il est possible, pour des systèmes IoT réels, que la remontée des informations soit soumise à du bruit, il est également envisageable qu'une partie de ces informations ne parviennent pas jusqu'à l'architecture chargée de protéger le système. Dans le but de tester la robustesse des stratégies d'agrégation face à la perte de données, nous considérons des jeux de données d'évaluation dans lesquels, de façon aléa-

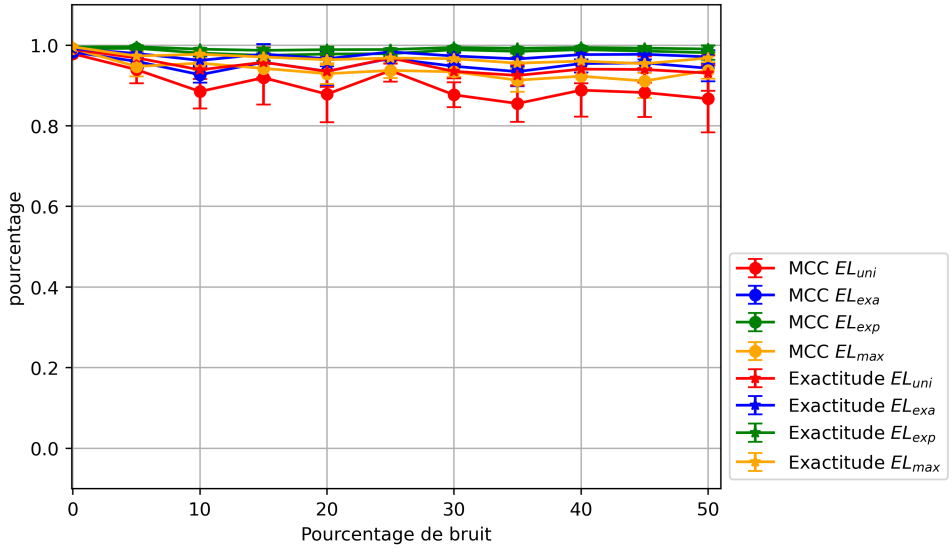


FIGURE 6.10 – Influence du bruit dans la détection d'anomalies lors de l'utilisation de stratégies d'agrégation sur les données du capteur canbeat avec une fenêtre de 60 secondes

toire, un certain pourcentage des données ne sont jamais transmis aux algorithmes de détection. Nous analysons les jeux de données provenant du bus CAN du véhicule (*canbeat*) ainsi que les deux sous-ensembles qui remontent les informations matérielles du système embarqué (*metricbeat*) que nous avons décrits précédemment. Le premier sous-ensemble de données du *metricbeat* décrit les informations relatives aux écritures disques, tandis que le second regroupe les données sur l'utilisation CPU. Encore une fois, nous utilisons les jeux de données CPU pour détecter la première étape de l'attaque. Sur la figure 6.11, nous pouvons voir l'évolution du coefficient de Matthews pour les données CPU. Les résultats de cette métrique ici sont proches de 60% lorsqu'aucune donnée n'est manquante là où l'exactitude est autour des 80%. Toutefois, nous remarquons un phénomène un peu particulier pour ces jeux de données. En effet, il semble que la perte d'informations améliore les performances de détection de toutes les stratégies d'agrégation utilisées par l'apprentissage ensembliste avec un coefficient de Matthews avoisinant les 80% et une exactitude proche de 90% lorsque la moitié des informations sont perdues. Cela s'explique par la présence d'une faible quantité de points catalogués comme faux positifs avant le début de l'attaque. Ainsi, plus la proportion de perte est importante, moins on retrouve ces points problématiques et donc, comme le taux de faux positifs diminue, les stratégies de détection dans leur ensemble obtiennent de meilleurs résultats.

Pour les données relatives aux écritures disques, les scores du coefficient de Matthews sont aussi inférieurs à ceux obtenus pour les autres sources de données. Cela s'explique par le faible nombre de points de données caractéristiques de l'attaque ainsi que par la présence d'un état correspondant à un comportement normal qui est proche de celui de la seconde étape de l'attaque. En effet, le transfert du fichier malveillant est rapide, moins de 2 secondes au total, et donc ne produit pas beaucoup de points de données. Pour certains des algorithmes de détection que nous utilisons, la disparition des points de données anormaux peut changer ce que les algorithmes considèrent comme normal et anormal. Nous observons sur la figure 6.12 que la perte de données introduit une grande variabilité au niveau de la détection. En revanche, les valeurs moyennes du coefficient de Matthews restent assez similaires et proche de 30% pour des pertes comprises entre 10 et 50% pour les différentes stratégies d'agrégation utilisées par l'apprentissage ensembliste.

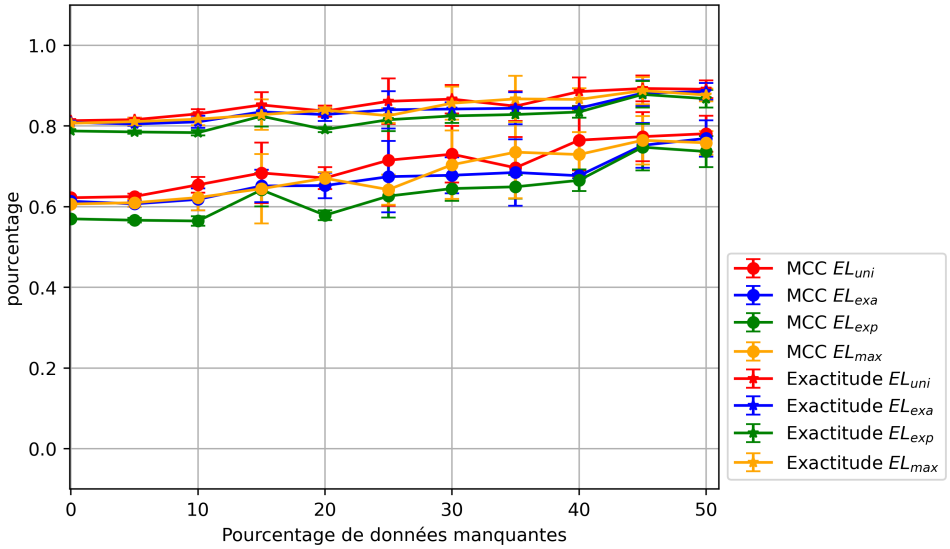


FIGURE 6.11 – Influence de la perte de données dans la détection d'anomalies lors de l'utilisation de stratégies d'agrégation sur les données du capteur metricbeat (cpu) avec une fenêtre de 60 secondes

La stratégie EL_{max} fait office d'exception avec un coefficient proche de 15%. L'exactitude, quant à elle, conserve une valeur élevée tant que la perte des données n'est pas trop importante. De la même façon que pour le coefficient de Matthews, nous observons une forte dégradation au delà de 15% de perte.

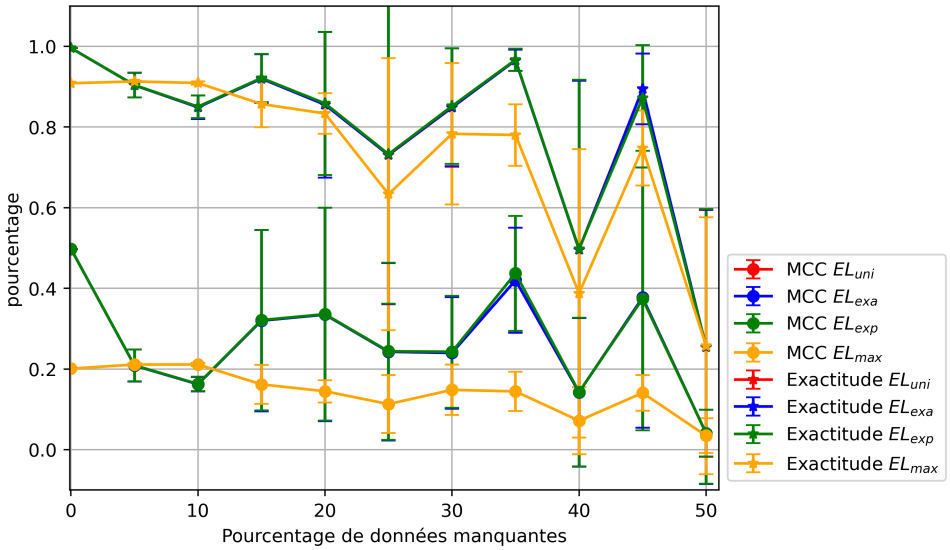


FIGURE 6.12 – Influence de la perte de données dans la détection d'anomalies lors de l'utilisation de stratégies d'agrégation sur les données du capteur metricbeat (diskio) avec une fenêtre de 60 secondes

Finalement, nous remarquons, sur la figure 6.13, que la détection de la dernière étape de l'attaque est peu influencée par la perte des données. En effet, le coefficient de Matthews et

l'exactitude se maintiennent toujours au-dessus de 95%, et même avec une perte de données comprise entre 0 et 50%. La perte de points de données au moment du début de l'attaque peut éventuellement faire manquer aux différentes méthodes une faible proportion des points anormaux et donc augmenter le taux de faux négatifs de façon extrêmement légère. Par exemple, la stratégie EL_{exp} voit son coefficient de Matthews passer de 100% lorsqu'il n'y a aucune perte à 98% en perdant la moitié des données.

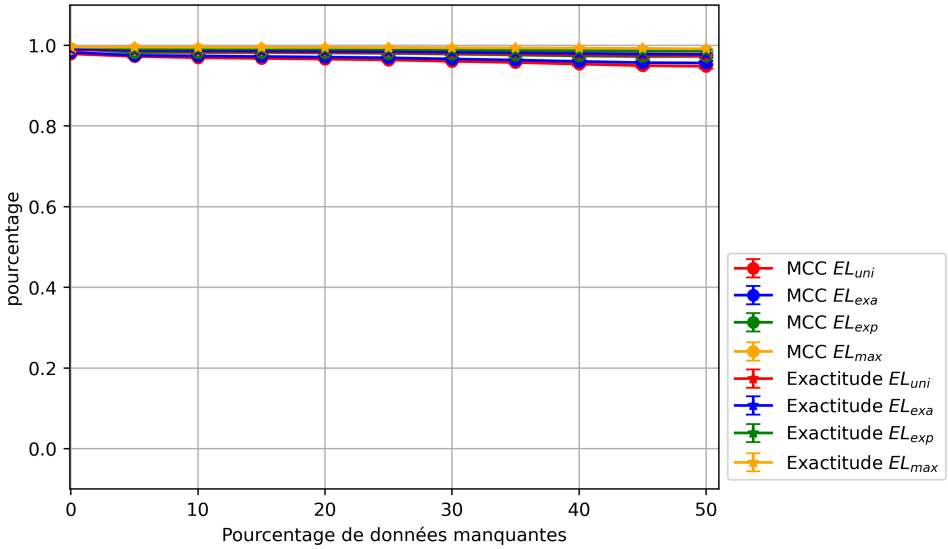


FIGURE 6.13 – Influence de la perte de données dans la détection d'anomalies lors de l'utilisation de stratégies d'agrégation sur les données du capteur canbeat avec une fenêtre de 60 secondes

Nous notons, comme dans le cas du bruitage des données, qu'il est plus facile de repérer l'attaque du bus CAN du véhicule que la seconde étape de l'attaque, car la surcharge de la capacité du bus de transmission est très nette. Nous trouvons, dans le cas du déploiement du fichier malveillant sur le véhicule, la même difficulté que dans la section précédente. En effet, en plus des difficultés à détecter les anomalies il est possible qu'elles fassent partie des données supprimées. La dégradation des performances pour de telles données intervient rapidement. Dans notre cas, nous l'observons pour des jeux de données ayant subi 5% de perte.

6.5 Synthèse

Dans ce chapitre, nous avons analysé les performances de notre méthode de détection ensembliste. Nous avons commencé par décrire les scénarios considérés, notamment une attaque complexe constituée de trois étapes à l'encontre de véhicules connectés, et étudier la construction de graphes de dépendances, qui permet de lier les différentes sources de données. L'attaque se décompose en (1) un scan des ports afin de détecter un port vulnérable sur le véhicule, (2) l'installation et l'exécution d'un fichier malveillant, et enfin (3) la réalisation d'un déni de service sur le bus CAN du véhicule, qui perturbe la bonne transmission des commandes. Une première série d'expériences nous a permis de comparer la pertinence des différentes stratégies d'agrégation des scores permettant de mettre en oeuvre notre méthode ensembliste, et de les confronter aux méthodes classiques de détection. Une seconde série d'expériences nous a permis de quantifier l'impact du mécanisme de retour adaptatif à base de fenêtre glissante. Les expériences ont montré

que la stratégie d'agrégation $\mathrm{EL}_{\mathrm{exp}}$ est la plus performante pour détecter les différentes étapes de l'attaque considérée en un minimum de temps. Nous avons également observé une diminution du temps de détection, lorsque nous réduisons la taille de la fenêtre glissante au détriment de l'exactitude, en particulier lorsque nous descendons sous une valeur limite pour la taille de la fenêtre. Enfin, nous avons évalué la robustesse de notre approche ensembliste au regard de perturbations, à savoir le bruit et la perte de données, pour les différentes étapes de l'attaque.

Chapitre 7

Développement et intégration du prototype

Sommaire

7.1 Introduction 110
7.2 Implémentation de la solution de détection 111
7.2.1 Transformation des données d'entrée 112
7.2.2 Application du *process mining* 113
7.2.3 Adaptation de la solution à l'apprentissage ensembliste 115
7.3 Conteneurisation de la solution de détection 116
7.3.1 Explicitation des entrées et des sorties de la phase d'apprentissage . . . 116
7.3.2 Explicitation des entrées et des sorties de la phase de détection 119
7.3.3 Choix des paramètres de pré-traitement 121
7.4 Intégration de l'approche à la plateforme SecureIoT 123
7.4.1 Présentation de la plateforme SecureIoT 123
7.4.2 Déploiement d'une API REST 126
7.5 Synthèse 128

7.1 Introduction

Dans ce chapitre, nous présentons notre prototype et ses différents éléments constitutifs. Celui-ci nous a servi à montrer les performances de nos solutions de détection d'anomalies dans les deux chapitres d'expérimentation de ce manuscrit. Dans le cadre du projet européen SecureIoT, plusieurs acteurs ont été amenés à travailler sur différents éléments d'une plateforme visant à améliorer la sécurité des objets connectés. De notre côté, nous avons mis à disposition notre solution de détection en l'intégrant à la plateforme globale. Son rôle est de fournir des services de sécurité (SECaaS) à des objets connectés. Afin d'illustrer le fonctionnement de cette plateforme, présenté dans le livrable public D2.5 [2] du projet SecureIoT, nous rappelons ses principaux modules dans la figure 7.1. Notre solution de détection des anomalies se situe dans la partie analyse des données et interagit directement avec le module de collecte et d'agrégation des données, celui de gestion des risques, mais également avec le répertoire global et le bus de données. Lors de la démonstration finale, nous avons pu montrer un exemple pratique dans lequel les valeurs de capteurs d'un objet connecté étaient récupérées par la partie de collecte et d'agrégation

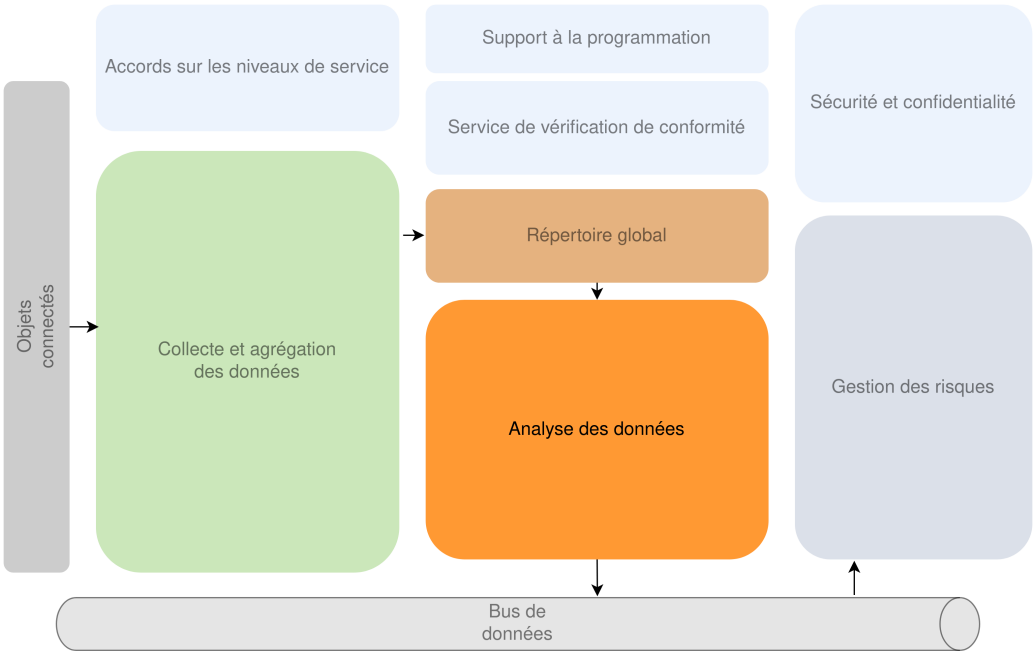


FIGURE 7.1 – Plateforme de sécurité du projet SecureIoT

des données pour être envoyées sur le répertoire global. Après cela, notre solution de détection est capable de récupérer ces données, pour soit construire des modèles de comportement, soit chercher à détecter les anomalies. Quand une anomalie est détectée, une alerte est transmise au module de gestion des risques via le bus de données. Avant d'aborder l'intégration de notre solution de détection, nous expliquons son fonctionnement et comment l'utiliser. Nous détaillons également comment nous avons adapté notre solution à l'apprentissage ensembliste. Puis, nous justifions notre choix de conteneuriser les différents éléments de notre solution de détection. Ainsi, nous décrivons les entrées et sorties des phases d'apprentissage et de détection d'un conteneur incluant notre solution. Finalement, dans une troisième section, nous voyons de quelle façon nous avons intégré notre solution de détection à la plateforme du projet européen SecureIoT. Nous détaillons comment la solution que nous avons proposée interagit avec les autres modules de la plateforme. Puis, nous décrivons l'interface que nous avons développée pour en faciliter l'intégration. Elle permet de mettre notre solution de détection à la disposition de l'ensemble de nos partenaires par le moyen d'un service web.

7.2 Implémentation de la solution de détection

Afin d'automatiser le traitement des données et ainsi de réaliser nos expériences, nous avons réalisé un prototype. Dans le reste de cette section, nous revenons sur l'ensemble des étapes de traitement des données et listons quelles technologies et formats de données ont été utilisés. Une démonstration de notre solution a été faite pendant la conférence internationale IEEE|IFIP NOMS en 2020 [78]. Nous montrons également comment ont été ajoutés les éléments propres à l'apprentissage ensembliste.

7.2.1 Transformation des données d'entrée

Dans le cadre de nos travaux, nous avons choisi d'implémenter notre solution à l'aide d'un outil de *process mining* couramment utilisé, facile d'accès et *open source*. Cet outil, appelé ProM, contient plusieurs algorithmes qui requièrent un certain format pour les fichiers d'entrée. Ainsi, dans le cas de données hétérogènes, il est nécessaire d'ajouter une étape de pré-traitement avant d'utiliser les algorithmes de *process mining* pour transformer les données d'entrée en un format adéquat. En effet, les différents algorithmes que nous avons étudiés fonctionnent grâce au traitement d'un fichier au format XML eXtensible Event Stream (XES), qui représente le journal d'événements d'un système. La figure 7.2 montre les différentes technologies qui ont été utilisées dans le pré-traitement. Elle fait directement le lien entre celles-ci et la présentation du bloc de pré-traitement de la figure 3.3 présentée dans le chapitre 3.

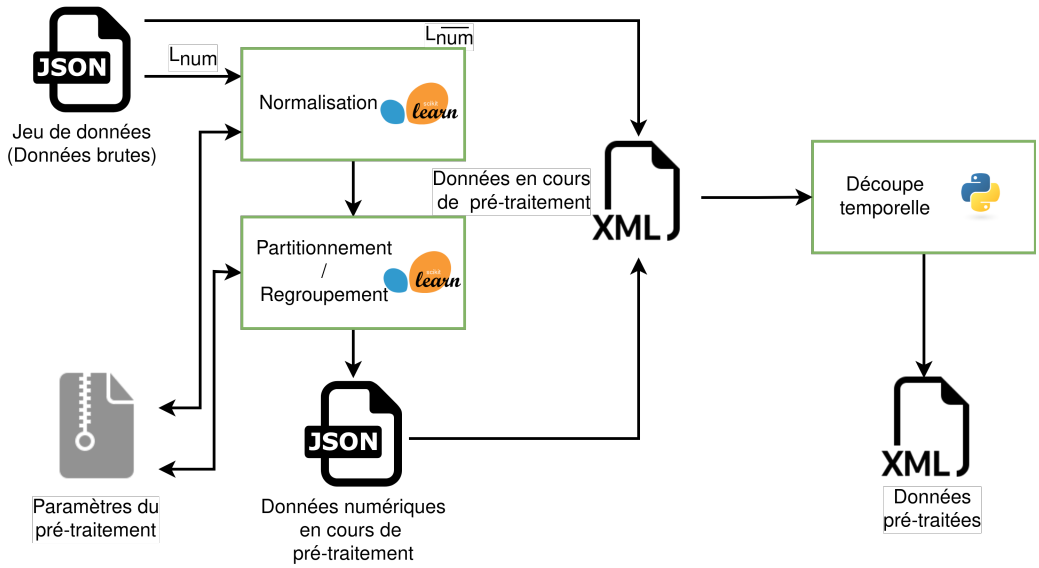


FIGURE 7.2 – Présentation du bloc de pré-traitement du prototype

Le sous-bloc de *normalisation* et de *partitionnement/regroupement* utilisent les méthodes de la bibliothèque scikit-learn [133], qui regroupe de façon *open source* un grand nombre de méthodes permettant le traitement des données et dont la documentation rend facile sa prise en main. Nous pouvons observer que le changement de format de nos données, pour obtenir le format XES, se fait au moment du regroupement des données de $L_{\overline{num}}$ avec celles passées par la normalisation et le partitionnement. Si l'on met à part l'entête du fichier XES, nous y retrouvons les noms et les valeurs des différents attributs de chacun des points de données en plus d'une valeur qui fait référence au numéro de groupe (Gr_i) obtenu par le partitionnement des données numériques lors du pré-traitement. Un point de données contient les informations de l'ensemble des différents attributs pour un temps donné, il représente donc l'état ($état_i$) du système à cet instant. L'exemple de la figure 7.3, obtenu lors de nos expériences sur les véhicules connectés, montre le format d'une partie du contenu d'un fichier XES. Les balises <*event*> nous permettent de faire la distinction entre deux points de données successifs.

```
<trace>
    <event>
        <date key="horodatage" value = "2020-07-29 08 :45 :14.900"/>
        <float key="vitesse_enclenchee" value = "3"/>
        <float key="angle_du_volant" value = "180"/>
        <float key="rotation_moteur" value = "2055"/>
        <float key="groupe" value = "0"/>
        <float key="etat" value = "0"/>
    </event>
    <event>
        <date key="horodatage" value = "2020-07-29 08 :45 :15.900"/>
        <float key="vitesse_enclenchee" value = "3"/>
        <float key="angle_du_volant" value = "20"/>
        <float key="rotation_moteur" value = "3148"/>
        <float key="numero_groupe" value = "1"/>
        <float key="etat" value = "1"/>
    </event>
</trace>
```

FIGURE 7.3 – Exemple de données présentes dans un fichier XES pour un véhicule connecté

C'est après cette étape de transformation que la découpe du jeu de données en sous-ensembles intervient. Elle permet de réduire le temps de traitement des données en réduisant la taille des modèles que nous allons générer, ainsi que celle des sous-ensembles des données de la phase de détection. En effet, nous allons chercher à éviter l'augmentation du temps de traitement induit par la recherche de l'alignement optimal, décrite dans la section 3.4.4 du chapitre 3, pour permettre la détection des anomalies. Une fois que nos données brutes ont été transformées au format XES, elles peuvent être utilisées pour générer des modèles de comportement du système étudié. Lors du traitement par l'algorithme de *process mining*, seul l'*etat* est utilisé. En effet, cet élément synthétise l'ensemble des données d'un point.

7.2.2 Application du *process mining*

À la suite de la transformation des données d'entrée, nous pouvons appliquer l'algorithme de *process mining* afin, soit de générer les modèles de comportement, pour la phase d'apprentissage, soit de chercher les anomalies qui caractérisent les attaques, dans la phase de détection. L'algorithme utilisé dans la phase d'apprentissage est l'*inductive miner* qui va, à partir de la succession des points de données, extraire des modèles de comportement sous la forme de réseaux de Petri. Nous avons choisi cet algorithme pour sa capacité à créer des modèles sous la forme de graphes non bloquants et à ne pas faire le choix arbitraire d'éliminer certaines transitions visibles dans les données. Notre prototype utilise la bibliothèque ProM, plus exactement sa version 6.8, pour à la fois l'algorithme de l'*inductive miner* et pour celui de vérification de conformité. La description précise du fonctionnement de cet algorithme a été réalisée dans le chapitre 2, où nous nous

Données d'évaluation	Modèle correspondant	Fitness
sous_ensemble_1	modèle_14	0,93
sous_ensemble_2	modèle_16	0,70
sous_ensemble_3	modèle_16	0,60
sous_ensemble_4	modèle_43	0,61
sous_ensemble_5	modèle_35	0,68
sous_ensemble_6	modèle_17	0,97
sous_ensemble_7	modèle_39	0,69
sous_ensemble_8	modèle_36	0,97
sous_ensemble_9	modèle_16	0,69
sous_ensemble_10	modèle_43	0,93
sous_ensemble_11	modèle_16	0,81
sous_ensemble_12	modèle_35	0,86

TABLE 7.1 – Exemple d'un fichier résultat obtenu lors de la phase de détection

sommes appuyés sur un exemple pour décrire ses étapes successives. Dans le cas de la phase de détection, nous utilisons un autre algorithme qui compare les données en entrée aux différents modèles disponibles. Cela permet de calculer un score de similarité entre ces données d'entrée et les modèles et donc, avec l'utilisation d'une valeur de seuil, de pouvoir faire la distinction entre les données normales et anormales. La figure 7.4 fait le rapprochement entre la figure 3.1 du chapitre 3, qui présente notre solution de détection, et le choix des technologies liées à son implémentation.

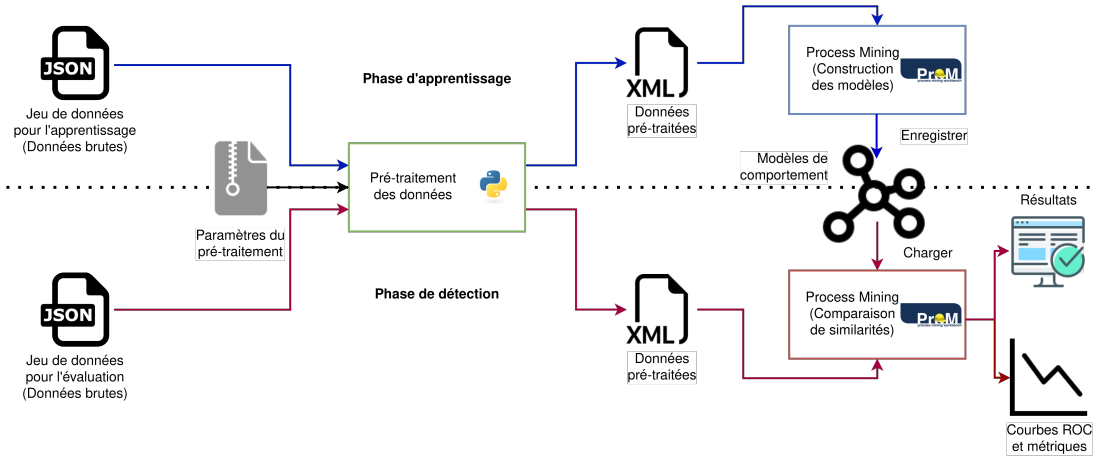


FIGURE 7.4 – Présentation de l'architecture du prototype

Le tableau 7.1 présente l'un de nos fichiers de résultats, on y trouve une liste des sous-ensembles obtenus après la phase de découpe associés au modèle le plus proche ainsi que la valeur de la *fitness*. Cette dernière est la métrique nous permettant d'évaluer la similarité entre les deux premiers éléments. Ainsi, la première ligne signifie que le premier sous-ensemble du jeu de données de la phase de détection est proche du quatorzième modèle créé dans la phase d'apprentissage. Nous observons que la correspondance entre les deux est de 93%. Dans le cas où notre seuil de détection est de 70%, les données représentées par le sous_ensemble_1 sont considérées comme normales. En revanche, ce n'est pas le cas pour le troisième sous-ensemble de données, qui obtient au mieux 60% de correspondance avec un modèle.

7.2.3 Adaptation de la solution à l'apprentissage ensembliste

Jusqu'à présent, nous avons pu voir de quelle façon nous avons développé notre solution de détection utilisant le *process mining*, pour détecter les comportements anormaux. Nous avons cherché à améliorer les performances de cette solution. Pour cela, nous avons fait le choix d'ajouter, à notre solution initiale, d'autres méthodes de détection et de regrouper leurs résultats à l'aide de stratégies d'apprentissage ensembliste. Cette partie du prototype est décrite par la figure 7.5.

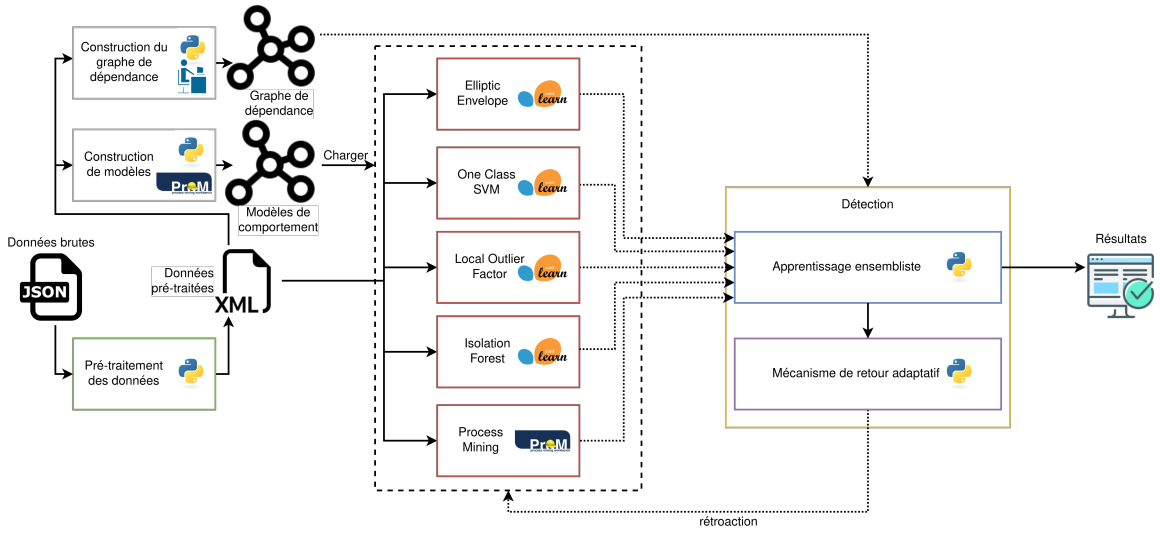


FIGURE 7.5 – Présentation de l'architecture du prototype incluant l'apprentissage ensembliste

Pour ces autres méthodes, nous avons utilisé à nouveau la bibliothèque scikit-learn qui intègre des méthodes telles que OCSVM, IF ou bien encore OCRF. Notre module génère un fichier de résultats pour chacune des méthodes de détection, dans lequel nous avons accès à un score de détection. Pour le *process mining*, nous obtenons un score pour chaque sous-ensemble de données, alors que pour les autres algorithmes, ce score est présent pour chaque point de données. Pour éviter d'éventuelles difficultés lors du calcul du score global, nous avons considéré le score du *process mining* comme une fonction constante par morceau. Une fois les différents fichiers à notre disposition, nous avons agrégé les différents résultats à l'aide de stratégies d'apprentissage ensembliste qui permettent de calculer un score de détection global qui est utilisé pour distinguer les données normales des données anormales. En plus de cela, nous avons ajouté un mécanisme de fenêtre glissante temporelle dans le but de lisser ces scores, et ainsi de limiter le taux de faux

positifs. Finalement, lorsque les résultats de l'apprentissage ensembliste générés par le prototype sont disponibles, il nous est possible de visualiser l'évolution de ces scores agrégés dans des graphes.

7.3 Conteneurisation de la solution de détection

Afin d'exporter plus facilement notre solution tout en conservant son efficacité au niveau des ressources, nous avons choisi d'utiliser la conteneurisation via la solution Docker. Elle présente un avantage de sécurité avec l'isolation des processus, qui empêche par défaut le conteneur d'échanger avec les autres éléments sur le serveur. De plus cette solution est plus légère en termes de ressources que la solution de virtualisation classique qui repose sur le déploiement de machines virtuelles. Il est également facile de créer plusieurs instances de notre image Docker avec différents paramètres pour traiter simultanément les données. En effet, nous avons facilement pu créer plusieurs conteneurs pour réduire le temps de sélection des meilleures associations de paramètres de pré-traitement (normalisation, partitionnement et découpe temporelle) lors de nos expériences. La conteneurisation de notre solution de détection nous a également permis de la mettre facilement à disposition de nos partenaires du projet SecureIoT.

Dans la suite du chapitre, nous détaillons la structure du dossier nécessaire à l'utilisation des méthodes définies dans l'image Docker. Pour cela, nous décrivons les entrées et les sorties des deux phases, à savoir la phase d'apprentissage et la phase de détection qui peuvent être lancées depuis un conteneur Docker. Puis, nous expliquons comment a été utilisée notre image Docker lors du choix des paramètres de pré-traitement.

7.3.1 Explicitation des entrées et des sorties de la phase d'apprentissage

La figure 7.6 illustre les composants de la phase d'apprentissage. Dans cette phase, nous construisons les modèles de comportement à partir de jeux de données ne contenant pas d'anomalie.

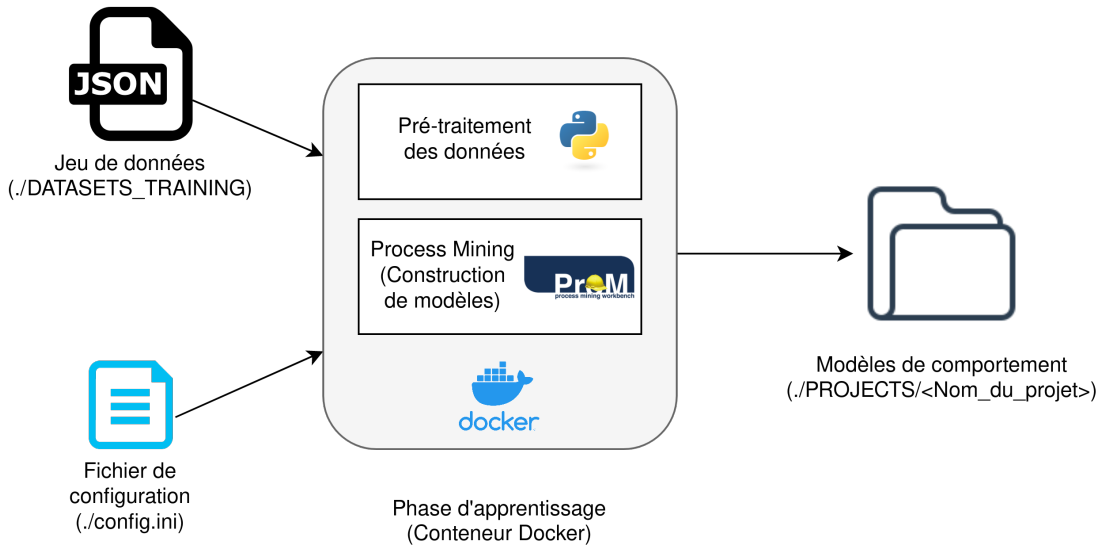


FIGURE 7.6 – Illustration de la phase d'apprentissage de notre prototype intégré dans un conteneur Docker

Sur la figure 7.7, qui liste les entrées nécessaires à la phase d'apprentissage, nous pouvons voir les jeux de données d'apprentissage qui sont stockés dans le dossier *./DATASETS_ TRAINING*, ainsi que le fichier de configuration *./config.ini* qui contient la valeur des paramètres du pré-traitement à prendre en compte, comme par exemple le choix de la méthode de normalisation des données.

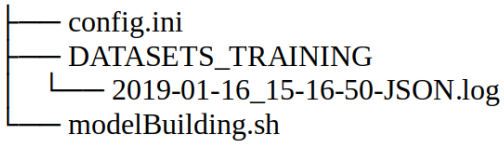


FIGURE 7.7 – Liste des données d'entrée pour la phase d'apprentissage

Il existe 6 sous-catégories dans notre fichier de configuration, comme le montre la figure 7.8.

```
[DATA]
dataset = idiada
normalization = min-max
clustering = KMEANS
project = PROJECTS/testP2

[KMEANS]
nb_clusters = 4
random_state = 0

[DBSCAN]
eps = 0.7
min_samples = 10

[BIRCH]
branching_factor = 100
n_clusters = None
threshold = 0.4

[BUILDING]
relativefolderpath = DATASETS_MODELS_BUILDING
timesplit = 10

[EVALUATION]
relativefolderpath = DATASETS_EVALUATION/DATASET
pm_algo = InductiveMiner
timesplit = 10
```

FIGURE 7.8 – Fichier de configuration (config.ini)

Dans la première d'entre elles, c'est-à-dire la catégorie *[DATA]*, il y a quatre paramètres : la provenance des données (*dataset*) qui nous permet d'indiquer le type de données que nous devons traiter ; le type de normalisation que notre architecture doit utiliser (*normalisation*), ici il est possible de choisir soit la méthode min-max soit z-score ; la méthode de partitionnement qui doit être appliquée (*clustering*), par exemple K-MEANS, BIRCH ou DBSCAN ; le nom du dossier dans lequel l'ensemble des informations sont sauvegardées (*project*).

Les trois catégories suivantes (*[KMEANS]*, *[BIRCH]* et *[DBSCAN]*) vont nous permettre de définir les paramètres de l'algorithme de partitionnement choisi. Seule la méthode correspondant au paramètre *clustering* de la catégorie *[DATA]* est prise en compte lors du traitement. Par exemple, si nous précisons que nous souhaitons utiliser K-MEANS alors seuls les paramètres

de la catégorie *[KMEANS]* seront pris en compte lors du partitionnement. L'avant-dernière catégorie (*[BUILDING]*) contient deux paramètres. Le premier correspond au temps de découpe en secondes (*timesplit*) qui est appliqué lors de la phase d'apprentissage. Le second paramètre (*relativefolderpath*), quant à lui, indique dans quel dossier se trouve le jeu de données utilisé dans la phase d'apprentissage. La dernière catégorie (*[EVALUATION]*) n'a aucune influence lors de la phase d'apprentissage.

Une fois que le jeu de données d'apprentissage est à notre disposition et que nous avons correctement renseigné les paramètres que nous souhaitons appliquer, il ne nous reste plus qu'à lancer le script bash "modelBuilding.sh" en précisant le nom que nous allons donner au projet, c'est-à-dire au dossier dans lequel nous allons sauvegarder l'ensemble des entrées et sorties de la phase d'apprentissage.

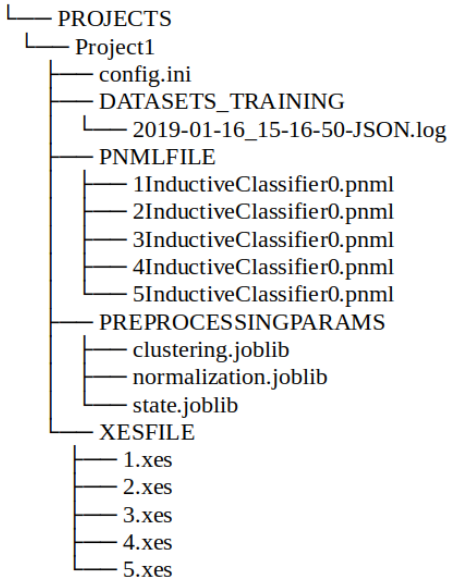


FIGURE 7.9 – Structure d'un dossier projet à la fin de la phase d'apprentissage

Comme le montre la figure 7.9, les sorties, qui sont placées dans un dossier projet, sont au nombre de trois, nous trouvons trois fichiers sérialisés, dans le dossier "PREPROCESSINGPARAMS", qui contiennent l'ensemble des caractéristiques des méthodes de normalisation et de partitionnement qui ont été utilisées sur le jeu de données d'apprentissage. Cela permet, lors de la phase de détection, de conserver une cohérence entre les données traitées en utilisant les mêmes paramètres avec les mêmes méthodes. Par exemple, si nous avons utilisé la normalisation min-max lors de la phase d'apprentissage, nous devons conserver les valeurs minimales et maximales des attributs. Ce sont ces valeurs que nous utiliserons lors de la normalisation de la phase de détection. De cette façon, nous sommes certains que deux valeurs identiques après normalisation correspondent bien à deux valeurs identiques avant normalisation. De plus, pour le partitionnement, nous aurons également accès au résultat complet de l'algorithme appliqué aux données d'apprentissage, ce qui nous assure l'homogénéité des différents groupes mis en avant par la méthode de partitionnement. Les autres éléments qui composent les sorties sont les fichiers XES, résultat de la phase de pré-traitement, que l'on trouve dans le dossier "XESFILE", ainsi que les modèles de comportement, fournis sous la forme de réseaux de Petri, dans le dossier "PNMLFILE".

Afin de s'assurer de la reproductibilité de nos expériences, nous conservons les entrées qui nous ont donné ces résultats. Ainsi, nous remarquons que l'on trouve, sur la figure 7.9, le jeu de données qui a été utilisé pour créer nos modèles de comportement, présent dans le dossier "DATASETS_TRAINING", ainsi que le fichier de configuration "config.ini".

7.3.2 Explicitation des entrées et des sorties de la phase de détection

La figure 7.10 permet de visualiser à la fois les entrées et les sorties de la phase de détection. Durant cette phase, nous allons nous appuyer sur des modèles générés lors de la phase d'apprentissage et calculer le taux de similarité (*fitness*) entre les modèles de comportement et les différentes parties du jeu de données à analyser.

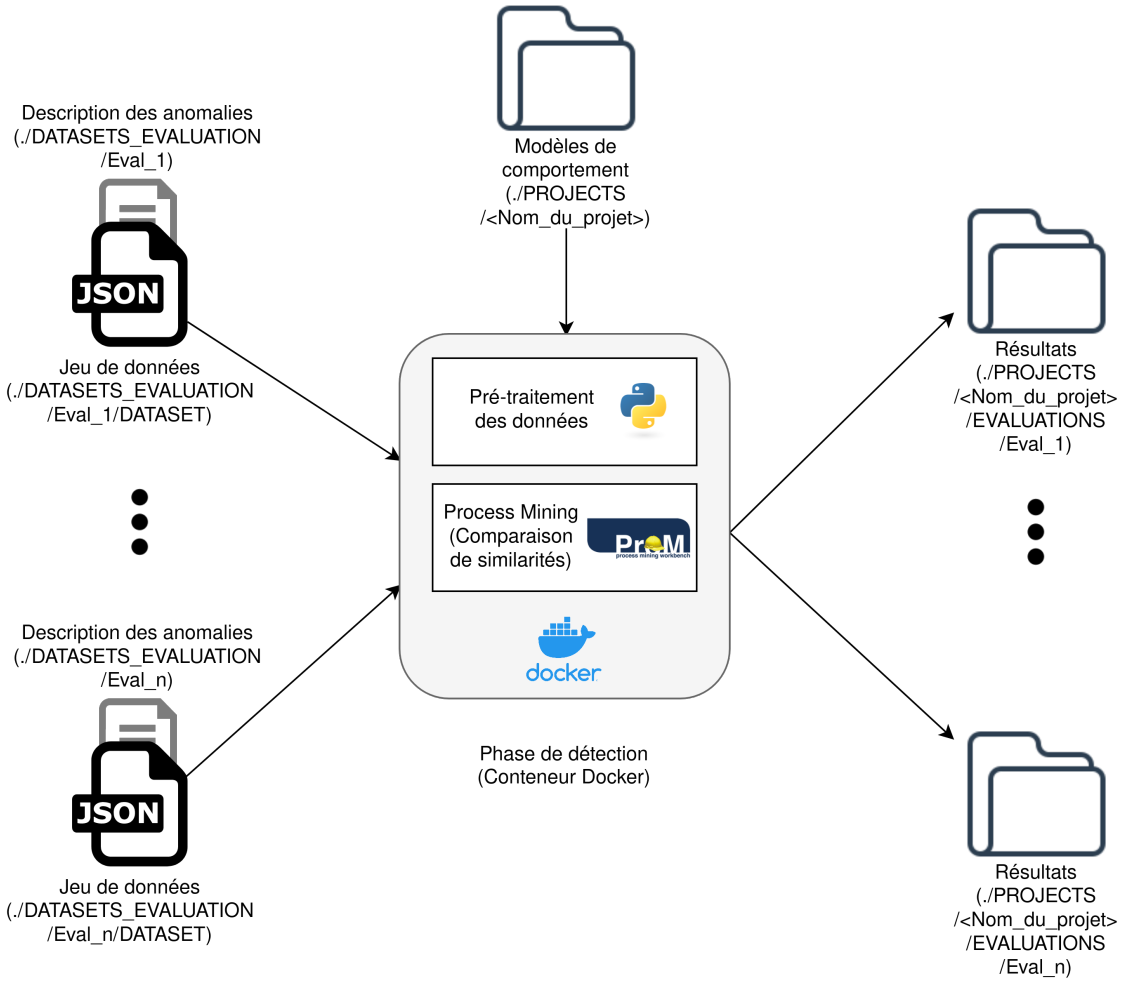


FIGURE 7.10 – Illustration de la phase de détection de notre prototype intégré dans un conteneur Docker

Lors de la phase de détection, nous allons avoir besoin de trois entrées. La première est un dossier "PROJECT", résultat de la phase d'apprentissage. Les deux autres, dont la liste est donnée dans la figure 7.11 sont les jeux de données d'évaluation. Nous retrouvons ces jeux de données dans le dossier "DATASETS_EVALUATION/<nom_evaluation>/DATASET". La description des anomalies correspondantes se situe dans le fichier "summary.txt". Toutefois, ce

fichier est optionnel pour la phase de détection, mais il nous permet de calculer des métriques telles que l'exactitude ou bien encore le coefficient de Matthews et donc de pouvoir choisir le paramétrage de la solution de détection.

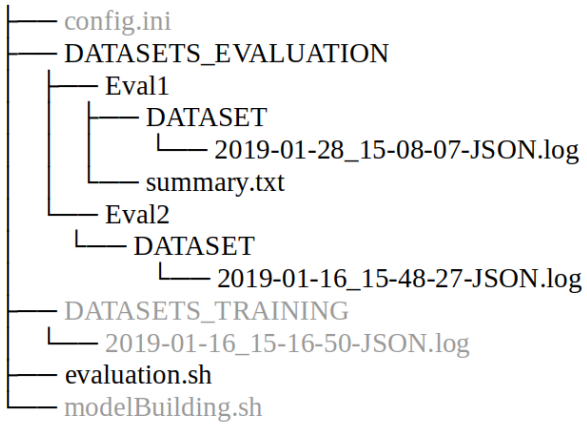


FIGURE 7.11 – Liste des données d'entrée pour la phase de détection

Il est possible d'appliquer notre méthode de détection sur plusieurs jeux de données différents tant que ces derniers correspondent bien au type de données qui ont été utilisées lors de la phase d'apprentissage. Pour cela, il suffit d'ajouter autant de dossiers "<nom_evaluation>", contenant chacun leur jeu de données et le fichier de description des anomalies, que nécessaire. Si l'on regarde la figure 7.11, on remarque que nous avons placé deux jeux de données pour la phase de détection "Eval_1" et "Eval_2". À partir du moment où ces différents éléments sont donnés en entrée, il est possible d'utiliser le script "evaluation.sh" en précisant bien le dossier projet qui doit être utilisé. De cette façon, nous allons compléter le projet avec les différents éléments qui apparaissent en noirs sur la figure 7.12.

Ainsi, en plus de copier les jeux de données de la phase de détection dans le dossier du projet, nous allons trouver l'ensemble des fichiers XES, dans le dossier "XESFILETOCHECK", qui sont obtenus à la fin de la partie de pré-traitement. Ce sont ces fichiers qui vont être rejoués sur les modèles de comportement. Nous allons également trouver un fichier "labels.json" qui désigne quelles parties du jeu de données contiennent des anomalies. Les autres éléments qui sont générés dans cette phase de détection se trouveront dans un dossier "RESULTS" du projet. Ce dossier contient trois documents, le premier est un fichier qui indique, le modèle le plus proche pour chaque sous-ensemble des jeux de données de détection. Il donne également la valeur de la *fitness* correspondante.

Dans le cas où le fichier de description des anomalies, qui correspond au fichier "summary.txt" de la figure 7.12, n'est pas vide, nous allons également avoir accès à la valeur de l'exactitude et des différentes métriques qui composent la matrice de confusion pour plusieurs valeurs de seuil de détection. On a, par exemple, accès aux valeurs de ces métriques pour un seuil fixé à 0,5 ou bien encore à 0,95. Les deux images dans notre dossier "RESULTS" vont être créées en utilisant ces valeurs. La première, "accuracy.png", montre l'évolution de l'exactitude, du taux de vrais et de faux positifs et négatifs en fonction de la valeur de seuil utilisée pour la détection. La seconde correspond à la courbe ROC.

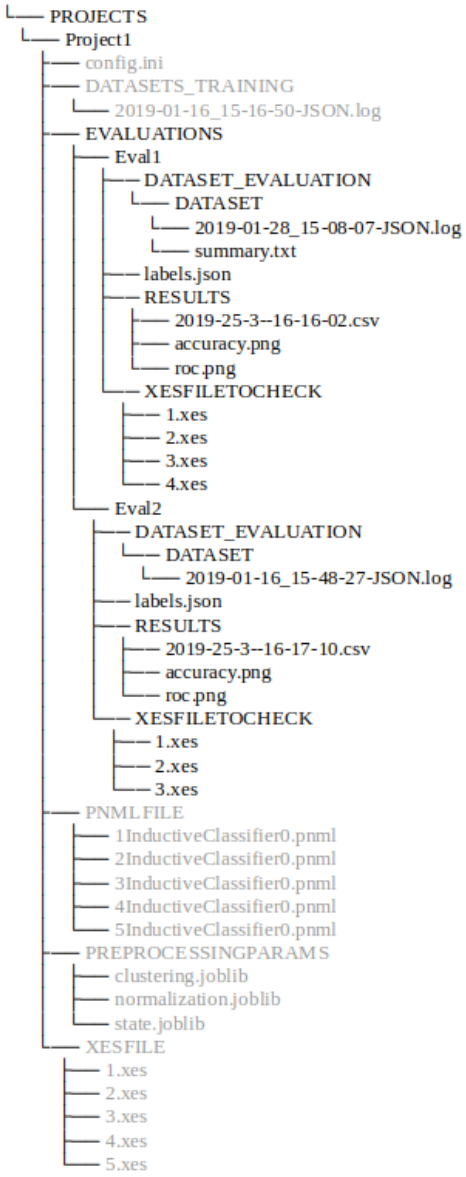


FIGURE 7.12 – Structure d'un dossier projet à la fin de la phase de détection

7.3.3 Choix des paramètres de pré-traitement

Lors de la recherche des paramètres de pré-traitement dans nos expériences, nous avons exploité l'un des avantages de la conteneurisation, qui est sa faculté à pouvoir facilement créer de nouvelles instances d'une image, pour générer quatre conteneurs. Le but recherché était de réduire le temps de traitement de l'ensemble des associations retenues pour les paramètres de pré-traitement. Comme le montre la figure 7.13, chacun des quatre conteneurs traite les données avec un ensemble de paramètres déterminés. De plus, nous pouvons voir que dès la fin de son traitement, le conteneur (ici le troisième) rend disponible ses résultats.

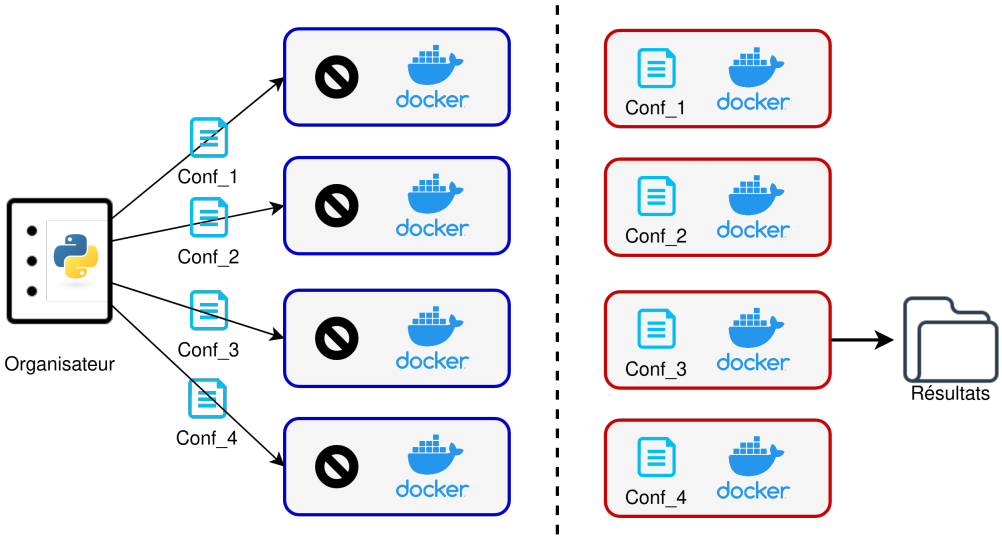


FIGURE 7.13 – Début de la recherche des paramètres de pré-traitement

Pour illustrer l'organisation de la répartition du travail, nous allons considérer qu'un conteneur dont le contour est rouge indique que celui-ci est en cours de traitement et donc n'est pas disponible. En revanche, lorsque son contour est bleu, il est disponible pour démarrer une nouvelle analyse. Ainsi, la figure 7.14 montre ce qu'il se passe une fois que le troisième conteneur a rendu disponibles ses résultats. Ce conteneur redevient disponible et va donc recevoir un nouveau fichier de configuration, comportant de nouveaux paramètres de pré-traitement, qu'il va devoir traiter.

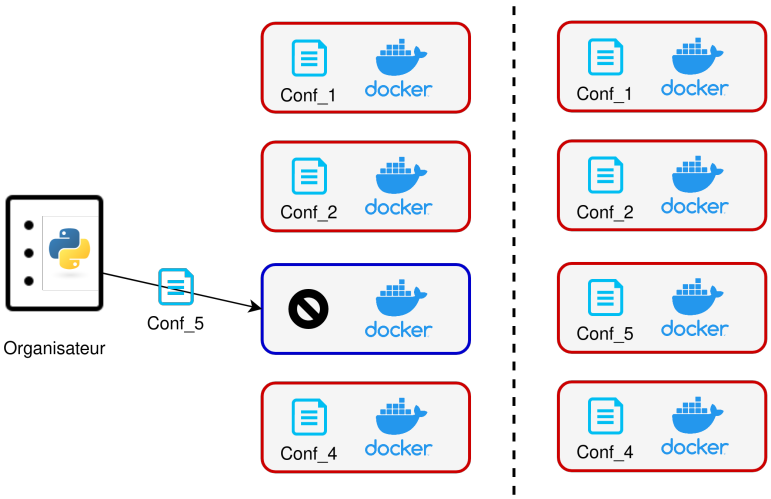


FIGURE 7.14 – Étape intermédiaire de la recherche des paramètres de pré-traitement

Cette étape s'arrête une fois que l'ensemble des combinaisons de paramètres retenus pour les expériences ont été testées. À partir des résultats, nous sélectionnons les paramètres de pré-traitement qui ont obtenu les meilleures performances. Ces paramètres seront utilisés pour traiter les prochaines données provenant de ce même système IoT.

7.4 Intégration de l'approche à la plateforme SecureIoT

Dans cette section, nous montrons comment a été intégrée notre solution dans la plateforme de sécurité du projet européen SecureIoT. Nous expliquons de quelle façon les données des objets connectés ont été collectées, ce qui correspond aux données d'entrée brutes que nous devons traiter. Puis, nous montrons comment les alertes sont construites et transmises lors de la phase de détection d'anomalies. Nous décrivons également le composant qui reçoit les alertes et qui doit faire le lien entre les analystes responsables de la sécurité du système et le module d'analyse. Dans le cadre du projet SecureIoT, nous étions responsables du traitement des données dans le pôle analyse le module analyse des données. D'autres acteurs, notamment IDIADA, LuxAI,IT'S OWL et INTRASOFT, ont travaillé sur la collecte et l'agrégation des données ainsi que sur le module de gestion des risques. Dans une dernière partie, nous détaillons l'interface que nous avons utilisée pour intégrer notre solution de détection à la plateforme du projet. Elle a été déployée sur les serveurs du projet et mettait à disposition de l'ensemble des acteurs les méthodes disponibles dans l'image Docker.

7.4.1 Présentation de la plateforme SecureIoT

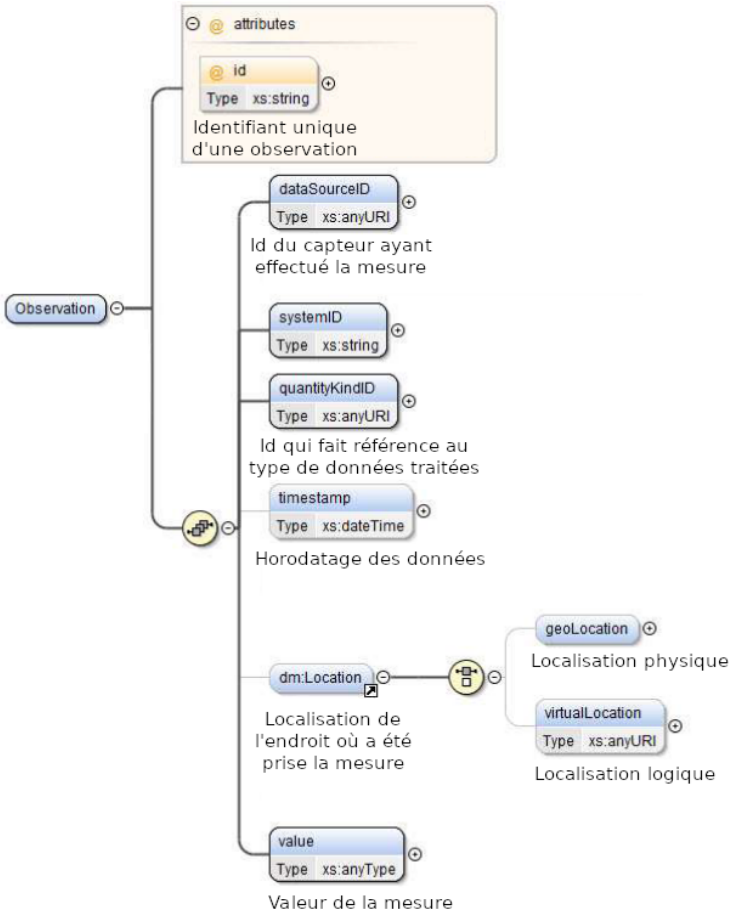


FIGURE 7.15 – Modélisation d'une observation

Nous avons pu avoir un aperçu de l'organisation de la plateforme du projet SecureIoT dans l'introduction de ce chapitre. Nous y avons également expliqué que notre solution de détection des anomalies fait partie prenante de la partie analyse. Elle va, d'une part, interagir avec le module de gestion des données et avec le répertoire global afin d'accéder aux données d'apprentissage et de détection. D'autre part, notre solution prévient le module de gestion des risques, en passant par le bus de données, lorsqu'elle détecte une anomalie.

En ce qui concerne la partie gestion des données, elle récolte les informations provenant des capteurs d'objets connectés. Ces dernières sont stockées sous la forme d'observations dans Elasticsearch [8], qui est la technologie de stockage retenue pour le répertoire global, sous un format normalisé résumé par la figure 7.15. Ainsi, à la fin du projet, les données en entrée de notre solution de détection suivent ce format dans lequel on retrouve les valeurs des capteurs installés sur les systèmes ainsi que l'horodatage correspondant. D'autres informations qui décrivent le contexte dans lequel évolue le système IoT sont également disponibles telles que l'identifiant du capteur ou bien encore sa localisation physique.

Dans le but de faciliter l'utilisation de ces données, une instance Kibana [11], qui est un greffon de visualisation pour Elasticsearch, a été mise en place par les partenaires du projet. Cette instance nous permet de facilement visualiser les données du répertoire global dans un navigateur web, comme on peut le voir sur la figure 7.16. Dans l'exemple présenté sur la figure, nous pouvons remarquer la présence de données à 10:03:34,2 que nous pouvons directement récupérer car nous connaissons leurs horodatages. Pour faire fonctionner notre solution de détection lors de l'intégration globale, nous devions récupérer les données à partir d'Elasticsearch avant de pouvoir y chercher des anomalies.

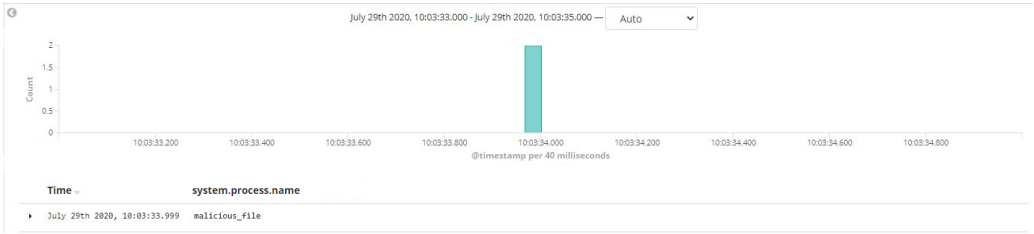


FIGURE 7.16 – Instance kibana utilisé dans le projet SecureIoT

Une fois les données récupérées, nous pouvons utiliser la solution de détection décrite dans le chapitre 3. Cette solution de détection se situe dans la partie d'analyse. Elle nous permet d'utiliser des méthodes relatives soit à la phase d'apprentissage contenue dans notre image docker, soit à la phase de détection. Pour la détection, nous distinguons deux cas. Dans le premier, nous considérons que la phase de détection se fait a posteriori, c'est-à-dire que nous cherchons à détecter des anomalies sur des données qui ont déjà été complètement récoltées. Dans le second cas envisagé, qui est le cas le plus courant et celui utilisé dans la démonstration finale du projet, la détection doit être faite au plus vite et les données sont traitées quasiment en temps réel. Pour que la détection soit la plus rapide possible, nous avons appliqué régulièrement la phase de détection sur de plus petits ensembles de données. Afin de simplifier l'utilisation de notre solution de détection et permettre son intégration dans la plateforme du projet SecureIoT, nous avons proposé une API REST qui fait appel à une instance de notre image Docker.

observation_56d614f8-b94d-4376-8dea-5ab801571582

{"dataSourceID": "08fa2ecc-d571-475b-94a3-0c4395ea550e", "systemID": "11ca6948-2cee-4ee6-abeb-4aa894a96133", "quantityKindID": "5e9ee151-b229-421c-aa08-b1540ef0c7a2", "timestamp": "2021-02-15 10:16:05", "location": {"geolocation": {"latitude": "52.2375786", "longitude": "0.1583927"}, "virtualLocation": "-"}, "observationValue": {"assetID": "38a13e2d-a044-4d3c-b491-d4cf90b591a5", "attackID": "CAPEC-94", "algorithm": "ProcessMining_Inria", "timestamp": "2021-01-21 11:03:19"}}

FIGURE 7.17 – Exemple d'une alerte envoyée sur le bus de données pour être traitée par la gestion des risques

Si des données sont identifiées comme problématiques, nous transmettons une alerte à la partie de gestion des risques. Cette alerte, dont un exemple est donné par la figure 7.17, est construite sur le même format que les observations que nous avons décrites dans la figure 7.15. Son identifiant correspond au champ supérieur de la figure 7.17 tandis que nous trouvons un ensemble d'informations permettant de savoir quel capteur a relevé les données anormales à l'aide de son identifiant "dataSourceID" et sa localisation "location". Ces alertes sont traitées par le module de gestion des risques qui, à partir du contexte défini dans le répertoire global, quantifie le niveau de risque. L'évaluation du niveau de risque doit permettre aux utilisateurs, ayant accès à la page du tableau de bord, de pouvoir appliquer un certain nombre de mesures pour assurer la protection du système, comme par exemple l'isolement de l'équipement compromis. La figure 7.18 donne une estimation du niveau de risque actuel. Ce niveau est amené à augmenter lorsque des alertes sont reçues par le module de gestion des risques. La valeur utilisée lors de l'incrémentation du score de risque est décidée par plusieurs éléments tels que la fréquence de réception des alertes ou bien encore du capteur qui a relevé les données. Cette jauge est l'un des éléments présents sur la page d'accueil du tableau de bord. Sur celui-ci, il est également possible de trouver un graphique de l'évolution au cours du temps du niveau de risque, comme le montre la figure 7.19.

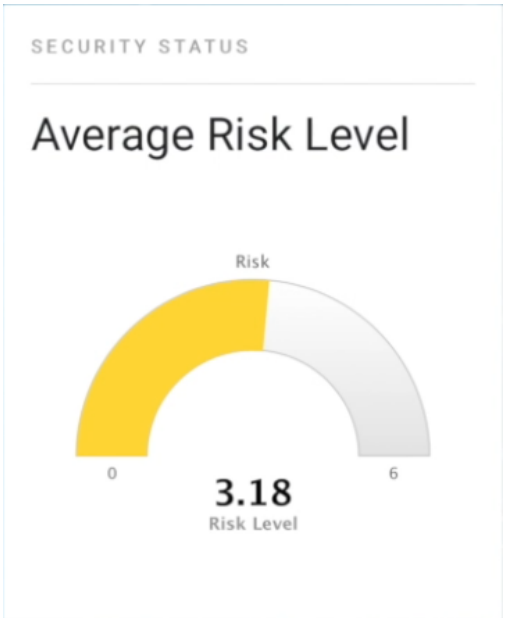


FIGURE 7.18 – Page d'accueil du tableau de bord

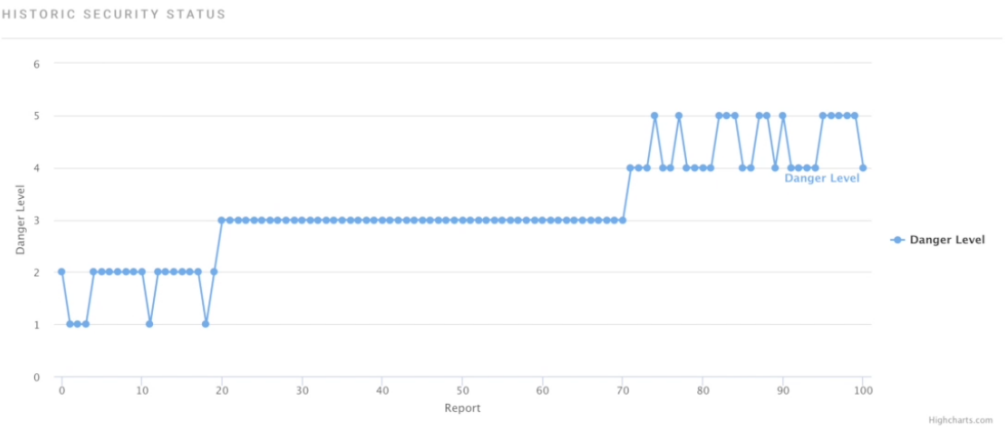


FIGURE 7.19 – Évolution du niveau de risque

7.4.2 Déploiement d'une API REST

Dans cette sous-section, nous détaillons l'API REST intégrant notre solution de détection à la plateforme du projet SecureIoT. Cette API est une interface de programmation d'application qui met à disposition de l'ensemble des acteurs notre solution pour construire les modèles de comportement et détecter des anomalies sur un système IoT. Son rôle est de faciliter l'accès à notre image Docker, déployée sur les serveurs du projet, à l'aide de méthodes documentées.

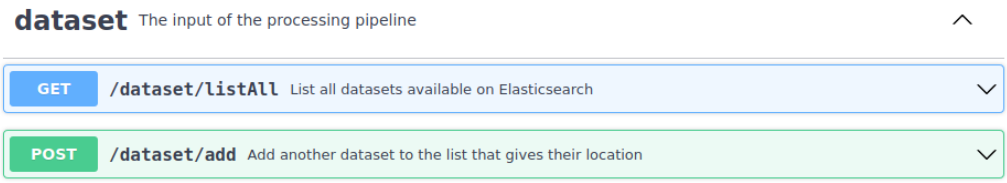


FIGURE 7.20 – Méthodes relatives aux jeux de données de l'API REST

Les méthodes de l'API REST que nous avons spécifiées sont au nombre de quatorze et sont réparties en cinq catégories. Il existe deux méthodes relatives aux jeux de données qui représentent notre première catégorie, comme indiqué sur la figure 7.20. La première méthode permet de lister l'ensemble des jeux de données disponibles et la seconde de renseigner la localisation dans le répertoire global d'un nouveau jeu de données. La gestion et la connaissance des jeux de données reposent sur la mise à jour d'un index qui contient les informations nécessaires pour accéder aux données comme par exemple leur intervalle temporel ou bien encore le nom de la sonde qui les a collectées sur les objets.

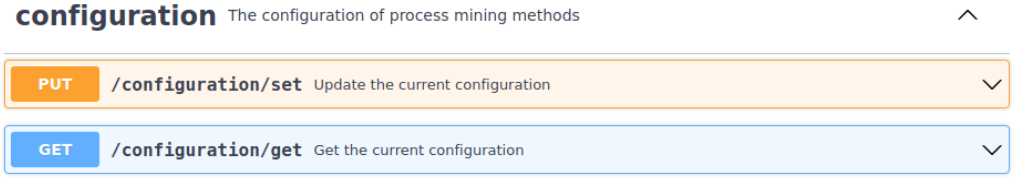


FIGURE 7.21 – Méthodes relatives à la configuration de l'API REST

Notre seconde catégorie contient les méthodes propres à la configuration de notre solution reposant sur le *process mining* présentée dans la figure 7.21. On y trouve une méthode permettant de récupérer la configuration actuelle, et une autre qui nous donne la possibilité de la modifier.

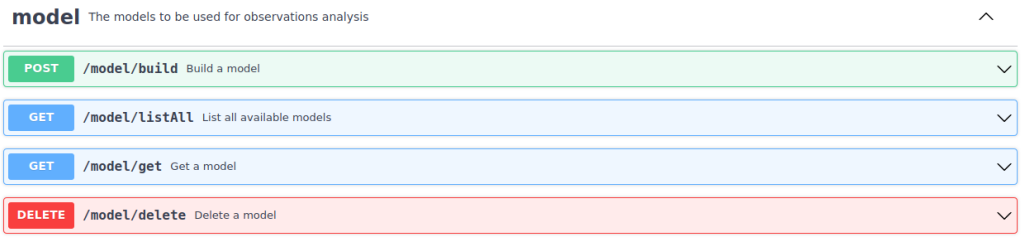


FIGURE 7.22 – Méthodes relatives à la construction des modèles de comportement de l'API REST

La troisième catégorie de notre API REST est composée de quatre méthodes qui sont visibles sur la figure 7.22. La première permet de construire, à partir du fichier de configuration évoqué précédemment et d'un jeu de données sélectionné, des modèles de comportement. Ces modèles doivent représenter le fonctionnement normal du système et pourront être utilisés, par la suite, dans la phase de détection. La seconde méthode nous permet de lister les modèles disponibles et, ainsi, de pouvoir choisir ceux que nous utiliserons dans la phase de détection. Les deux dernières méthodes de cette sous-catégorie vont nous permettre soit de récupérer les modèles disponibles qui ont été, dans un premier temps, créés par la méthode "model/build", soit de supprimer un ensemble de modèles. Dans ces deux derniers cas, il est intéressant de lister les modèles disponibles avec la méthode "/model/listAll".

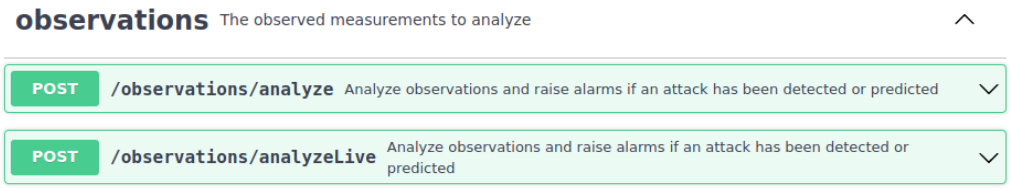


FIGURE 7.23 – Méthodes relatives à la détection des anomalies de l'API REST

Dans une quatrième catégorie, nous avons implémenté deux méthodes qui vont nous servir pour la phase de détection. Pour cela, il faut fournir l'emplacement de modèles obtenus lors d'une phase d'apprentissage ainsi que le jeu de données à évaluer. L'une de ces méthodes, "/observations/analyze", considère l'intégralité des données entre deux dates et appliquer la phase de détection immédiatement. On peut, de cette façon, faire une analyse a posteriori puisque cette méthode nécessite que l'intégralité des données entre ces deux dates soient présentes au lancement de la méthode. En effet, même si de nouvelles données sont injectées dans Elasticsearch, elles ne seront pas considérées pour la détection. La seconde méthode,"observations/analyzeLive", quant à elle permet de traiter la phase de détection quasiment en temps réel. L'utilisateur doit fournir un paramètre supplémentaire à la méthode qui correspond au temps entre deux récupérations de données depuis Elasticsearch. Ainsi, cette méthode récupère un sous-ensemble de données qu'elle fait passer par la phase de détection jusqu'à atteindre la date de fin.

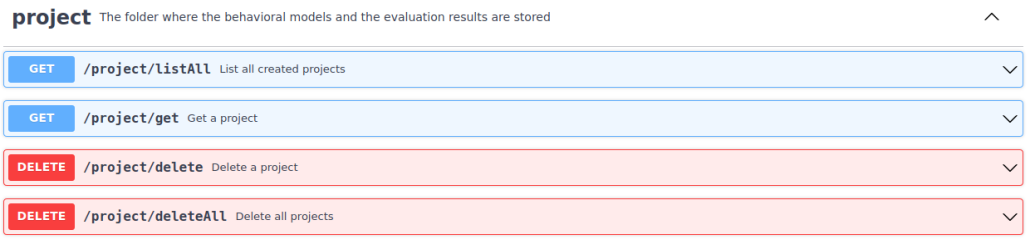


FIGURE 7.24 – Méthodes relatives à la gestion des projets de l'API REST

Finalement, dans la dernière catégorie, nous trouvons les méthodes de gestion des projets qui sont au nombre de quatre, comme présenté par la figure 7.24. La première des méthodes permet à l'utilisateur de lister l'intégralité des projets disponibles en affichant leurs noms. La contenance des projets peut varier. En effet, pour exister, un utilisateur doit nécessairement utiliser au minimum la méthode de construction de modèles de la troisième sous-catégorie afin de construire des modèles. En revanche, il n'est pas dit que tous les projets présents possèdent un jeu de données d'évaluation ou un dossier de résultat. Ces deux éléments, en plus du dossier contenant les fichiers XES du jeu de données d'évaluation, ne sont créés que lorsque nous utilisons l'une des méthodes de détection, c'est-à-dire "observations/analyze" ou "observations/analyzeLive". Les trois méthodes restantes vont nous permettre soit de récupérer un projet, d'en supprimer un ou de supprimer l'ensemble des projets disponibles.

7.5 Synthèse

Nous avons présenté dans ce chapitre les travaux de prototypage de notre solution de détection. Nous avons tout d'abord décrit la conception et l'implantation du prototype, du prétraitement des données à l'application du *process mining*, et détaillé comment la solution avait été adaptée pour permettre une approche ensembliste en exploitant plusieurs méthodes de détection exécutées en parallèle. Nous avons ensuite présenté la conteneurisation de la solution en utilisant l'environnement docker. Cette conteneurisation permet un déploiement facilité avec une isolation du service et l'intégration des bibliothèques et dépendances requises pour son fonctionnement dans une image docker, dont nous avons décrit la structuration. Elle accroît la flexibilité et la portabilité d'exécution de notre prototype, pour lui permettre de fonctionner de façon fiable et prévisible sur une variété de systèmes. Nous avons enfin détaillé l'intégration de notre solution au sein de la plateforme de sécurité SecureIoT, et notamment l'interface de programmation (API) qui a servi de support pour cette intégration, en offrant des accès à différentes méthodes permettant notamment le paramétrage de la solution, la création de nouveaux modèles et la détection des anomalies à partir de jeux de données collectés par la plateforme. Par ailleurs, le prototypage a servi lors de plusieurs démonstrations (conférence internationale, évaluation de fin de projet), et a montré la pertinence de notre solution. Lors de ces démonstrations, nous avons pu montrer comment se déroulent la collecte, l'analyse des données, l'affichage du niveau de risque et des propositions de contre-mesures.

Chapitre 8

Conclusions et perspectives

Sommaire

8.1 Bilan des travaux réalisés 129
8.2 Publications 131
8.3 Perspectives de recherche 132

Le développement de l'Internet s'est caractérisé par le large déploiement d'objets connectés, dont les vulnérabilités constituent un vecteur d'attaques important. Cette thèse porte sur la conception de nouvelles méthodes de détection pour les systèmes IoT hétérogènes. Elle s'est inscrite dans le cadre du projet européen SecureIoT, dont l'objectif était de construire et implanter une plateforme de sécurité pour la protection de tels systèmes. Nous avons travaillé en particulier sur l'analyse et la détection des anomalies affectant les objets connectés à partir de données collectées sur ces systèmes. Notre solution est adaptée à des systèmes hétérogènes, et fournit une aide rapide et compréhensible à l'administrateur/opérateur pour choisir des contre-mesures adaptées. Nous avons privilégié les méthodes de détection semi et non supervisées, car elles nous permettent d'utiliser des données non labellisées lors de la phase de détection.

8.1 Bilan des travaux réalisés

Les travaux menés lors de cette thèse ont porté sur trois contributions principales. La première contribution porte sur une méthode de détection reposant sur le *process mining* couplé à des méthodes de pré-traitement. La seconde contribution est une adaptation de notre précédente proposition dans laquelle nous faisons intervenir une approche ensembliste prenant en considération plusieurs méthodes de détection simultanément. La troisième contribution correspond au prototypage de notre solution de détection, ainsi que son intégration dans une plateforme de sécurité.

Exploitation du *process mining* pour la détection. Nous avons proposé une méthode de détection pour les systèmes IoT hétérogènes, qui repose sur le couplage d'un algorithme de *process mining* à des techniques de pré-traitement. Le *process mining* seul est rapidement confronté à une explosion d'états, lorsqu'il est appliqué sur des données hétérogènes. Les techniques de pré-traitement permettent de réduire ce phénomène. Nous avons décrit les deux phases importantes de la solution, à savoir une phase d'apprentissage et une phase de détection. La première phase permet de générer des modèles de comportement du système sous la forme de réseaux de

Petri à partir des données pré-traitées. Ces représentations permettent d'améliorer la compréhension du fonctionnement du système par l'administrateur et de l'aider dans la sélection des contre-mesures à appliquer en cas d'anomalies. Lors de la phase de détection, notre méthode calcule automatiquement un score de similarité (*fitness*) entre les modèles de comportement et les données à tester pour mettre en évidence les anomalies potentielles. Nous avons réalisé plusieurs séries d'expériences afin d'évaluer notre solution à partir de données fournies par des partenaires industriels, qui permettent de montrer la pertinence de ce couplage pour la détection d'anomalies sur des données hétérogènes provenant d'objets connectés. Nous avons également comparé les performances de détection de notre solution avec cinq autres méthodes couramment utilisées (*elliptic envelope, support-vector machine, local outlier factor*, *isolation forest* et *one class random forest*). Notre approche obtient les meilleures performances de détection avec une précision, un F1 score et un coefficient de Matthews respectivement plus élevés de 68%, 56% et 47% par rapport aux autres méthodes. Nous avons également quantifié la robustesse de notre solution et avons observé que sa détection restait meilleure que celles des autres méthodes comparatives, lorsque nous ajoutions des perturbations (bruit gaussien de 20% ou perte de 25% des données). Toutefois, il apparaît que le temps de détection de notre méthode est parfois trop long ce qui nous a amenés à recourir à une adaptation ensembliste.

Adaptation ensembliste de notre solution. Nous avons ensuite adapté notre solution de détection à travers une approche ensembliste, qui permet d'intégrer un ensemble de méthodes de détection simultanément. Cette approche est associée à un mécanisme de retour adaptatif visant à améliorer la réactivité de la détection. Le principe de cette solution ensembliste réside dans l'agrégation des scores de plusieurs méthodes de détection pour obtenir de meilleures performances que les méthodes prises individuellement. Nous avons décrit quatre stratégies d'agrégation des scores de détection, ainsi que le mécanisme de retour adaptatif s'appuyant sur une fenêtre glissante. Nous avons appliqué cette solution à différents scénarios, notamment un scénario d'attaque en plusieurs étapes affectant un véhicule connecté, intégrant un scan des ports du véhicule, le déploiement d'un fichier malveillant sur le véhicule, et son exécution résultant en un déni de service sur le bus CAN. Nos expériences ont montré que la stratégie d'agrégation EL_{exp}, qui attribue un poids en fonction de l'exponentielle de l'exactitude à chaque méthode de détection, est au moins 56% plus rapide que les autres méthodes et son exactitude est jusqu'à 17% plus élevée. La diminution de la taille de la fenêtre glissante permet de réduire encore le temps de détection, avec globalement une réduction supplémentaire pouvant aller jusqu'à 63% pour les différentes agrégations. Nous avons aussi évalué la robustesse de l'approche ensembliste, et avons remarqué une dégradation rapide des performances lors de l'introduction de bruit gaussien quand les données anormales sont proches des données normales. Concernant la perte de données, nous observons cette dégradation lorsqu'une étape de l'attaque est de faible durée.

Prototypage et intégration à une plateforme. Nous avons également réalisé un prototype de notre solution de détection, qui nous a servi pour les expérimentations et pour la mise en oeuvre dans la plateforme du projet européen SecureIoT. Nous avons décrit la conception et l'implantation du prototype, dont toute la chaîne de la phase d'apprentissage et de la phase de détection a été développée. Nous avons aussi assuré la conteneurisation du prototype en utilisant l'environnement docker. Cette conteneurisation permet une plus grande portabilité de la solution, ainsi qu'une plus grande isolation au regard des bibliothèques, puisque celles-ci peuvent être directement intégrées à l'image docker. Nous avons également développé une interface de programmation (API) de type REST, qui permet facilement le paramétrage de la solution pour la

création des modèles de comportement et pour la détection des anomalies. Le prototype interagit avec différents modules de la plateforme de sécurité, à savoir le répertoire global où sont stockées les données collectées par le module de gestion des données à partir des objets connectés du système, et un module de gestion des risques auquel sont envoyées des alertes, lorsque des anomalies sont détectées par notre solution. Le prototype a été présenté lors de plusieurs démonstrations, notamment lors d'une conférence internationale [78] et lors de l'évaluation finale du projet européen SecureIoT. Ces démonstrations ont permis d'en présenter les différentes fonctionnalités à partir des données collectées sur les systèmes IoT de partenaires industriels européens.

8.2 Publications

Les travaux menés dans cette thèse ont abouti à plusieurs publications dans des conférences et journaux internationaux, ainsi qu'à la publication d'un chapitre pour un ouvrage international. Nous présentons ces publications ci-dessous dans l'ordre chronologique :

— Alexandru Vulpe, Ali Paikan, Razvan Craciunescu, Pouyan Ziafati, Sofoklis Kyriazakos, Adrien Hemmer, Rémi Badonnel. IoT Security Approaches in Social Robots for Ambient Assisted Living Scenarios. In Proceedings of the International Symposium on Wireless Personal Multimedia Communications (IEEE WPMC), 2019.

— Adrien Hemmer, Remi Badonnel, and Isabelle Chrisment. A Process Mining Approach for Supporting IoT Predictive Security. In Proceedings of the Network Operations and Management Symposium (IEEE/IFIP NOMS), 2020.

— Adrien Hemmer, Rémi Badonnel, Jérôme François, and Isabelle Chrisment. A Process Mining Tool for Supporting IoT Security. In Proceedings of the Network Operations and Management Symposium (IEEE/IFIP NOMS), Demonstration Paper, 2020.

— Adrien Hemmer, Mohamed Abderrahim, Rémi Badonnel, Jérôme François, and Isabelle Chrisment. Comparative Assessment of Process Mining for Supporting IoT Predictive Security. IEEE Transactions on Network and Service Management (IEEE TNSM), 18(1), 2021.

— Jürgen Neises, Spyridon Evangelatos, John Soldatos, Thomas Walloschke, George Moldovan, Hendrik Eikerling, Bianca Popovici, Cosmin Grigoras, Daniel Calvo and Adrien Hemmer. Trustworthy Human–machine Assistance by Dynamic Process Security Monitoring in Industrial Environments. Soft Computing in Smart Manufacturing : Solutions toward Industry 5.0, Tatjana Sibalija and J. Paulo Davim, Berlin, Boston, De Gruyter, 2021.

— Adrien Hemmer, Mohamed Abderrahim, Rémi Badonnel, and Isabelle Chrisment. An Ensemble Learning-Based Architecture for Security Detection in IoT Infrastructures. In Proceedings of the International Conference on Network and Service Management (IFIP/IEEE CNSM), pages 180–186, 2021.

8.3 Perspectives de recherche

Plusieurs perspectives de recherche ont été identifiées au regard des travaux réalisés pendant cette thèse sur les méthodes de détection pour la sécurité des systèmes IoT hétérogènes :

Intégration de méthodes de détection complémentaires. Dans le cadre du projet européen SecureIoT, nous nous devions de proposer une solution de détection des anomalies dont les modèles soient facilement compréhensibles. En outre, il devait être simple pour l'administrateur/opérateur d'enquêter sur les alertes et de pouvoir comprendre la raison de leurs apparitions. C'est pourquoi nous avons fait le choix de nous focaliser sur le *process mining*, couplé à des techniques de pré-traitement. Par la suite, nous avons adapté la solution pour intégrer d'autres algorithmes dans le but d'améliorer les performances de détection à partir des données hétérogènes. Dans ce contexte, nous sommes intéressés par l'intégration de nouvelles méthodes de détection, comme la méthode LSTM (Long Short Term Memory) [47] qui s'appuie sur l'usage de réseaux de neurones. Il sera intéressant de déterminer comment la complémentarité des méthodes peut faciliter la compréhension des alertes, notamment dans le cas où certaines méthodes fournissent des résultats difficilement explicables, et également de prendre en compte les coûts de traitement induits par l'utilisation simultanée de plusieurs méthodes.

Gestion des alertes et contre-mesures automatiques. Lors du projet SecureIoT, nous avons utilisé nos méthodes de détection sur trois cas d'usage. Nous avons donc pu vérifier que l'utilisation du *process mining* était possible pour détecter les anomalies de véhicules connectés, de machines à injection plastique (industrie 4.0), ainsi que de robots d'assistance. Bien que la gestion des alertes et le choix des contre-mesures à prendre sortent du cadre de notre travail, il est nécessaire de considérer ces deux aspects. En effet, choisir les contre-mesures adéquates peut s'avérer complexe pour les systèmes IoT en fonction de leur nature. Par exemple, dans la plupart des cas, il n'est pas envisageable d'arrêter immédiatement un système IoT industriel sans conséquences importantes. A contrario, l'installation d'une mise à jour de sécurité et le redémarrage des appareils chez un particulier présentent, dans la majeure partie des cas, des risques moindres. C'est pourquoi la sélection de contre-mesures doit être adaptée, et celles-ci doivent chercher à réduire également l'impact sur le fonctionnement des systèmes. Aussi, la sélection automatique de contre-mesures à partir des alertes générées et du contexte considéré est un défi important à investiguer.

Exploitation de la *threat intelligence*. Dans ce manuscrit, nous avons présenté une solution de détection qui repose sur l'analyse des données collectées à partir de systèmes IoT hétérogènes. La *threat intelligence* consiste en l'utilisation et le partage d'informations de sécurité d'autres systèmes informatiques, notamment sur les nouvelles menaces. En effet, pour tenir compte du contexte de sécurité à une plus large échelle, il est important de s'appuyer sur des bases de connaissances additionnelles mises à disposition par d'autres organismes, tels que typiquement le MITRE [1] ou le NIST (*National Institute of Standards and Technology*) [6], ou encore celles disponibles à partir de la plateforme MISP (*Malware Information Sharing Platform*) développée par le CIRCL [155]. Ces bases de connaissances recensent des informations importantes sur les menaces et vulnérabilités des systèmes informatiques, incluant ceux de l'Internet des Objets. Leur prise en compte permettra d'ajuster le paramétrage de nos méthodes de détection en fonction des vulnérabilités potentielles présentes sur les systèmes IoT et leurs objets connectés.

Glossaire

API REST ou REST API Representational State Transfer Application Program Interface : interface de programmation d'application qui respecte les contraintes de l'architecture REST

BIRCH Balanced Iterative Reducing and Clustering using Hierarchies : algorithme de partitionnement hiérarchique

CAN Controller Are Network : système de communication interne à la plupart des véhicules

CANDoS Controller Are Network Denial of Service : attaque par saturation de service sur le bus CAN du véhicule

DBSCAN Density-Based Spatial Clustering of Applications with Noise : algorithme de partitionnement de données bruitées faisant intervenir la notion de densité spatiale

DDoS Distributed Denial of Service : attaque collective par saturation/déni de service

DNS Domain Name System : service de traduction de noms de domaines

DT Decision Tree : arbre de décision

EE Elliptic Envelope : algorithme de l'enveloppe elliptique pour le partitionnement

HMM Hidden Markov Model : modèle de Markov caché

IDS Intrusion Detection System : système de détection des intrusions

IF Isolation Forest : algorithme de la forêt d'isolement

IIoT Industrial Internet of Things : Internet des objets industriels

IISF Industrial Internet Security Framework : modèle d'architecture pour les systèmes IoT

IM Inductive Miner : algorithme de *process mining* utilisant le *directly-follow graph* pour construire un modèle de comportement

IoT Internet of Things : Internet des objets

K-NN K-Nearest Neighbors : algorithme de partitionnement des K plus proches voisins

LL Log Loss : métrique permettant d'évaluer l'incertitude d'un modèle

LOF Local Outlier Factor : algorithme du facteur aberrant local

LSTM Long Short Term Memory : algorithme de mémoire longue à court terme

MCC Matthews Correlation Corefficient : coefficient de corrélation de Matthews

MitM Man-in-the-Middle ou homme au milieu : attaque consistant à se placer entre deux entités en train de communiquer

OCRF One Class Random Forest : algorithme de la forêt aléatoire à une classe

OCSVM One Class Support Vector Machine : machine à vecteurs de support

ORA OpenFog Reference Architecture : modèle d'architecture pour les systèmes IoT

OSI Open Systems Interconnection Model : modèle de communication pour les systèmes informatiques

OUTRES OUTlier ranking in RElevant Subspace projections : méthode pour trier les données aberrantes

OWASP Open Web Application Security Project : organisation à but non lucratif qui travaille sur la sécurité des applications Web

PCA Principal Component Analysis : analyse en composantes principales

PR Precision Recall : caractéristique de performance donnant l'évolution de la précision en fonction du rappel

Rec NN Recurrent Neural Network : réseau de neurones récurrent

Rep NN Replicator Neural Network : réseau de neurones réplicateurs

REST Representational State Transfer Application Program Interface : ensemble de contraintes architecturales

RF Random Forest : algorithme de la forêt aléatoire

ROC Receiver Operating Characteristic : caractéristique de performance donnant l'évolution du taux de vrais positifs en fonction du taux de faux positifs

RSA Rivest Shamir Adleman : algorithme de cryptographie asymétrique

SPOT Stream Projected Outlier deTector : détecteur de données aberrantes en temps réel

SQL Structured Query Language : langage permettant d'exploiter les bases de données relationnelles

SSL Secure Sockets Layer : protocole permettant l'authentification et le chiffrement des données sur le Web

TCP Transmission Control Protocol : protocole de contrôle des transmissions

TSM Transition System Miner : algorithme de *process mining* reposant sur les régions pour créer un modèle de comportement

V2X Vehicle-to-Everything : module permettant la communication de véhicules entre eux, entre les véhicules et les infrastructures ainsi qu'entre les véhicules et les piétons

XES eXtensible Event Stream : adaptation du format XML pour représenter un journal d'évènements

XML eXtensible Markup Language : langage de balisage extensible

Z-Wave protocole radio conçu pour la domotique

Table des figures

1.1 Plateforme de sécurité utilisée dans le projet européen SecureIoT [2] 2
1.2 Principaux chapitres du manuscrit et leurs relations 5

2.1 Composition d'un systèmes IoT 8
2.2 Caractéristiques propres aux systèmes IoT 9
2.3 Classification des méthodes de détection 15
2.4 Exemple d'un arbre de décision 17
2.5 Exemple d'une forêt aléatoire 18
2.6 Détection d'anomalies en utilisant DBSCAN 19
2.7 Détection des anomalies par l'elliptic envelope (EE) 21
2.8 Exemple d'anomalie dans une séquence temporelle 22
2.9 Réduction de l'espace de représentation des données 23
2.10 Détection des anomalies pour le OCSVM 23
2.11 Détection des anomalies pour les réseaux de neurones 24
2.12 Composition d'une cellule de la méthode LSTM [47] 24
2.13 Illustration du fonctionnement du partitionnement hiérarchique 25
2.14 Illustration du fonctionnement du *process mining* 26
2.15 Graphe des suivants directs obtenu à partir des données 28
2.16 Opérateurs servant à la découpe du graphe des suivants directs 29
2.17 Première étape de découpe du graphe des suivants directs avec l'opérateur séquentiel 29
2.18 Seconde étape de découpe du graphe des suivants directs avec l'opérateur du choix exclusif 30
2.19 Troisième étape de découpe du graphe des suivants directs avec l'opérateur boucle 30
2.20 Quatrième étape de découpe du graphe des suivants directs avec l'opérateur séquentiel 31
2.21 Cinquième étape de découpe du graphe des suivants directs avec l'opérateur parallèle 32
2.22 Conversion de l'arbre de processus (*process tree*) en réseau de Petri 32
2.23 Réseau de Petri obtenu par l'inductive miner 33
2.24 Paramètres permettant de caractériser un état 33
2.25 Système de transitions obtenu en considérant les deux précédents points de données pour en déduire l'état 34
2.26 Définition d'une région S_1 vers S_2 où S_2 est dans S' 35
2.27 Définition d'une région S_1 vers S_2 où S_1 est dans S' 35
2.28 Définition d'une région S_1 vers S_2 où S_1 et S_2 sont ou non dans S' 35
2.29 Réseau de Petri obtenu par le *transition mining system* 36

3.1 Vue d'ensemble de l'architecture du système de détection d'anomalies 39

3.2 Exemple d'un réseau de Petri 41
3.3 Vue détaillée du sous-bloc de pré-traitement des données 42
3.4 Exemple de réseau de Petri 49

4.1 Étude de l'influence du temps de découpe sur les performances de détection à l'aide des courbes ROC pour les véhicules connectés (a), l'industrie 4.0 (b) et les robots d'assistance (c) 61
4.2 Étude de l'influence des paramètres de partitionnement sur les performances de détection à l'aide des courbes ROC pour les véhicules connectés (a), l'industrie 4.0 (b) et les robots d'assistance (c) 63
4.3 Modèle de comportement au démarrage du véhicule 65
4.4 Détection des anomalies sur le véhicule connecté dans un cas de sur-apprentissage 65
4.5 Détection d'anomalies sur le véhicule connecté 66
4.6 Courbe ROC obtenue pour le véhicule connecté 67
4.7 Modèle d'un cycle du processus d'injection 67
4.8 Détection d'anomalies sur le système d'injection (Industrie 4.0) 68
4.9 Modèle d'un enchaînement de geste du robot d'assistance 69
4.10 Détection d'anomalies sur le robot d'assistance 69
4.11 Courbe ROC obtenue pour le robot d'assistance 70
4.12 Comparaison des courbes ROC obtenues par le *process mining* et les autres méthodes de détection pour les véhicules connectés (a), l'industrie 4.0 (b) et les robots d'assistance (c) 74
4.13 Comparaison des courbes PR de l'ensemble des méthodes de détection pour les véhicules connectés (a), l'industrie 4.0 (b) et les robots d'assistance (c) 76
4.14 Comparaison des performances du *process mining* et des autres algorithmes de détection en utilisant l'ensemble des métriques d'évaluation pour les véhicules connectés (a), l'industrie 4.0(b) et les robots d'assistance (c) 77
4.15 Influence du bruit dans le jeu de données sur les performances de détection . . . 79
4.16 Influence de la perte de données sur les performances de détection 80

5.1 Architecture de la solution de détection utilisant l'apprentissage ensembliste . . . 85
5.2 Exemple d'une infrastructure IoT 86
5.3 Graphe de dépendance correspondant 87
5.4 Utilisation d'une fenêtre glissante de taille α pour lisser le score de détection . . . 91
5.5 Localisation de l'attaque dans le graphe de dépendance 93
5.6 Évolution de la taille de la fenêtre glissante en cas de détection d'attaque 93

6.1 Figure illustrant le positionnement des trois sources de données sur le véhicule connecté 95
6.2 Valeur maximale de corrélation croisée observées pour l'ensemble des méthodes de détection lors des différentes phases de l'attaque 97
6.3 Graphe de dépendance final pour le véhicule connecté 98
6.4 Comparaison du temps de détection cumulé obtenu pour les trois phases de l'attaque par différentes méthodes de détection (*process mining*, *local outlier factor* et les quatre méthodes d'agrégation des scores utilisées par l'apprentissage ensembliste) 99
6.5 Comparaison du temps de détection cumulé pour les trois phases de l'attaque avec l'utilisation du mécanisme de retour adaptatif ($\alpha = 60s$ et $\beta = 40s$ 101

6.6 Influence de la fenêtre glissante temporelle sur le temps de détection pour les différentes méthodes de détection pour la première phase de l'attaque (variation de la fenêtre de 10 à 60 secondes) . . . 102
6.7 Influence de la fenêtre glissante temporelle sur l'exactitude pour les différentes méthodes de détection pour la troisième phase de l'attaque (variation de la fenêtre de 10 à 60 secondes) . . . 103
6.8 Influence du bruit dans la détection d'anomalies lors de l'utilisation de stratégies d'agrégation sur les données du capteur metricbeat (cpu) avec une fenêtre de 60 secondes . . . 104
6.9 Influence du bruit dans la détection d'anomalies lors de l'utilisation de stratégies d'agrégation sur les données du capteur metricbeat (diskio) avec une fenêtre de 60 secondes . . . 105
6.10 Influence du bruit dans la détection d'anomalies lors de l'utilisation de stratégies d'agrégation sur les données du capteur canbeat avec une fenêtre de 60 secondes . 106
6.11 Influence de la perte de données dans la détection d'anomalies lors de l'utilisation de stratégies d'agrégation sur les données du capteur metricbeat (cpu) avec une fenêtre de 60 secondes . . . 107
6.12 Influence de la perte de données dans la détection d'anomalies lors de l'utilisation de stratégies d'agrégation sur les données du capteur metricbeat (diskio) avec une fenêtre de 60 secondes . . . 107
6.13 Influence de la perte de données dans la détection d'anomalies lors de l'utilisation de stratégies d'agrégation sur les données du capteur canbeat avec une fenêtre de 60 secondes . . . 108

7.1 Plateforme de sécurité du projet SecureIoT . . . 111
7.2 Présentation du bloc de pré-traitement du prototype . . . 112
7.3 Exemple de données présentes dans un fichier XES pour un véhicule connecté . . 113
7.4 Présentation de l'architecture du prototype . . . 114
7.5 Présentation de l'architecture du prototype incluant l'apprentissage ensembliste . 115
7.6 Illustration de la phase d'apprentissage de notre prototype intégré dans un conteneur Docker . . . 116
7.7 Liste des données d'entrée pour la phase d'apprentissage . . . 117
7.8 Fichier de configuration (config.ini) . . . 117
7.9 Structure d'un dossier projet à la fin de la phase d'apprentissage . . . 118
7.10 Illustration de la phase de détection de notre prototype intégré dans un conteneur Docker . . . 119
7.11 Liste des données d'entrée pour la phase de détection . . . 120
7.12 Structure d'un dossier projet à la fin de la phase de détection . . . 121
7.13 Début de la recherche des paramètres de pré-traitement . . . 122
7.14 Étape intermédiaire de la recherche des paramètres de pré-traitement . . . 122
7.15 Modélisation d'une observation . . . 123
7.16 Instance kibana utilisé dans le projet SecureIoT . . . 124
7.17 Exemple d'une alerte envoyée sur le bus de données pour être traitée par la gestion des risques . . . 125
7.18 Page d'accueil du tableau de bord . . . 125
7.19 Évolution du niveau de risque . . . 126
7.20 Méthodes relatives aux jeux de données de l'API REST . . . 126
7.21 Méthodes relatives à la configuration de l'API REST . . . 126

7.22 Méthodes relatives à la construction des modèles de comportement de l'API REST 127
7.23 Méthodes relatives à la détection des anomalies de l'API REST 127
7.24 Méthodes relatives à la gestion des projets de l'API REST 128

Liste des tableaux

2.1 Tableau récapitulatif des différences entre IoT et IIoT [4] 10
2.2 Tableau récapitulatif des différentes attaques associées à chaque catégorie OWASP pour les systèmes IoT [7, 121] 12
2.3 Tableau récapitulatif de différentes approches de sécurisation des systèmes IoT [146] 13
2.4 Tableau d'association entre les catégories des travaux de sécurisation et la classification des vecteurs d'attaques pour les systèmes IoT proposée par OWASP 14

3.1 Coût d'alignement maximal 50
3.2 Coût d'alignement optimal 50

4.1 Tableau explicatif des données relatives aux informations de conduite du véhicule 56
4.2 Tableau explicatif des données relatives au système CAN du véhicule 56
4.3 Tableau explicatif des données relatives au système d'injection plastique (Industrie 4.0) 57
4.4 Tableau explicatif des données relatives aux moteurs du robot d'assistance . . . 58
4.5 Tableau explicatif des données relatives aux routeurs wifi visibles par le robot d'assistance 59
4.6 Meilleur résultat expérimental pour chacun des jeux de données 60
4.7 Description des caractéristiques des jeux de données utilisés dans l'évaluation des performances 64
4.8 Valeurs des paramètres de la phase de pré-traitement 75

5.1 Application du coefficient de Pearson sur l'exemple 88

6.1 Tableau explicatif des données relatives au véhicule connecté et provenant du capteur *Packetbeat* 96
6.2 Tableau explicatif des données relatives au véhicule connecté et provenant du capteur *Canbeat* 97
6.3 Exactitudes obtenues avec les différentes méthodes de détection pour les trois phases de l'attaque considérée 100

7.1 Exemple d'un fichier résultat obtenu lors de la phase de détection 114

Bibliographie

[1] Common Vulnerabilities and Exposures. https://cve.mitre.org/. Accessed : 2022-09-08.

[2] DELIVERABLE D2.5 - Architecture and Technical Specifications of SecureIoT Services_Final version. http://secureiot.eu/D2.5.pdf. Accessed : 2022-08-17.

[3] France : plus de 9 millions d'attaques DDoS en 2021. https://www.sekurigi.com/2022/05/france-plus-de-9-millions-dattaques-ddos-en-2021/. Accessed : 2022-06-22.

[4] Good Practices for Security of Internet of Things in the Context of Smart Manufacturing. https://www.enisa.europa.eu/publications/good-practices-for-security-of-iot/at_download/fullReport. Accessed : 2022-07-6.

[5] "Je suis le père Noël" : un hacker pirate une caméra Ring dans une chambre d'enfant. https://www.clubic.com/domotique/video-surveillance/actualite-879620-suis-pere-noel-hacker-pirate-camera-ring-chambre-enfant.html. Accessed : 2022-08-04.

[6] National Vulnerability Database. https://nvd.nist.gov/. Accessed : 2022-09-08.

[7] OWASP Internet of Things Project. https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project#tab=Main. Accessed : 2022-07-6.

[8] The heart of the free and open Elastic Stack. https://www.niceideas.ch/roller2/badtrash/entry/lambda-architecture-with-kafka-elasticsearch. Accessed : 2020-03-16.

[9] Une attaque informatique majeure a paralysé une partie du Web pendant plusieurs heures. https://www.lemonde.fr/pixels/article/2016/10/21/une-cyber-attaque-massive-perturbe-de-nombreux-sites-internet-aux-etats-unis_5018361_4408996.html. Accessed : 2022-08-10.

[10] What is the Internet-of-Things (IoT) ? https://aws.amazon.com/what-is/iot/. Accessed : 2022-07-5.

[11] Your window into the Elastic Stack. https://www.elastic.co/kibana/. Accessed : 2022-11-01.

[12] Z-Wave For Consumers. https://z-wavealliance.org/z-wave_for_consumers/. Accessed : 2022-08-10.

[13] Étude : il y a eu 9,75 millions d'attaques DDoS en 2021. https://siecledigital.fr/2022/03/24/etude-il-y-a-eu-975-millions-dattaques-ddos-en-2021/. Accessed : 2022-06-22.

[14] W.M.P. Aalst, van der. *Process Mining : Discovery, Conformance and Enhancement of Business Processes*. Springer, Germany, 2011.

[15] Charu C Aggarwal. Outlier Analysis. In *Data mining*, pages 237–263. Springer, 2015.

[16] David Airehrour, Jairo Gutierrez, and Sayan Kumar Ray. Securing RPL Routing Protocol from Blackhole Attacks using a Trust-Based Mechanism. In *Proceedings of the International Telecommunication Networks and Applications Conference*, pages 115–120, 2016.

[17] Rima Al-Ali, Robert Heinrich, Petr Hnetynka, Adrian Juan-Verdejo, Stephan Seifermann, and Maximilian Walter. Modeling of Dynamic Trust Contracts for Industry 4.0 Systems. In *Proceedings of the European Conference on Software Architecture : Companion Proceedings*, New York, NY, USA, 2018. Association for Computing Machinery.

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

[19] Asayel AlAbdullatif, Kholood AlAjaji, Norah Saad Al-Serhani, Rachid Zagrouba, and Maryam AlDossary. Improving an Identity Authentication Management Protocol in IIoT. In *Proceedings of the International Conference on Computer Applications & Information Security*, pages 1–6, 2019.

[20] Max Alaluna, Luís Ferrolho, José Rui Figueira, Nuno Neves, and Fernando M.V. Ramos. Secure Multi-Cloud Virtual Network Embedding. *Computer Communications*, 155 :252–265, 2020.

[21] Cristina Alcaraz, Giuseppe Bernieri, Federica Pascucci, Javier Lopez, and Roberto Setola. Covert Channels-Based Stealth Attacks in Industry 4.0. *IEEE Systems Journal*, 13(4) :3980–3988, 2019.

[22] Salwa Alem, David Espes, Eric Martin, Laurent Nana, and Florent De Lamotte. A Hybrid Intrusion Detection System in Industry 4.0 Based on ISA95 Standard. In *Proceedings of the International Conference on Computer Systems and Applications*, pages 1–8, 2019.

[23] James Allan, Jaime G Carbonell, George Doddington, Jonathan Yamron, and Yiming Yang. Topic Detection and Tracking Pilot Study Final Report. 1998.

[24] Ralph Ankele, Stefan Marksteiner, Kai Nahrgang, and Heribert Vallant. Requirements and Recommendations for IoT/IIoT Models to Automate Security Assurance through Threat Modelling, Security Analysis and Penetration Testing. In *Proceedings of the International Conference on Availability, Reliability and Security*, New York, NY, USA, 2019. Association for Computing Machinery.

[25] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, et al. Understanding the Mirai Botnet. In *Proceedings of the USENIX Security Symposium*, pages 1092–1110, 2017.

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

[27] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things : A survey. *Computer networks*, 54(15) :2787–2805, 2010.

[28] Philipp Autenrieth, Christian Lörcher, Christian Pfeiffer, Tobias Winkens, and Ludwig Martin. Current Significance of IT-Infrastructure Enabling Industry 4.0 in Large Companies. In *Proceedings of the International Conference on Engineering, Technology and Innovation*, pages 1–8. IEEE, 2018.

[29] Zeinab Bakhshi, Ali Balador, and Jawad Mustafa. Industrial IoT Security Threats and Concerns by Considering Cisco and Microsoft IoT Reference Models. In *Proceedings of the Wireless Communications and Networking Conference Workshops*, pages 173–178. IEEE, 2018.

[30] Paulo C. Bartolomeu, Emanuel Vieira, Seyed M. Hosseini, and Joaquim Ferreira. Self-Sovereign Identity : Use-cases, Technologies, and Challenges for Industrial IoT. In *Proceedings of the International Conference on Emerging Technologies and Factory Automation*, pages 1173–1180, 2019.

[31] Elisabeth Bauer, Oliver Schluga, Silia Maksuti, Ani Bicaku, David Hofbauer, Igor Ivkic, Markus G. Tauber, and Alexander Wöhrer. Towards a Security Baseline for IaaS-Cloud Back-Ends in Industry 4.0. In *Proceedings of the International Conference for Internet Technology and Secured Transactions*, pages 427–432, 2017.

[32] Sunny Behal, Amanpreet Singh Brar, and Krishan Kumar. Signature-Based Botnet Detection and Prevention. In *Proceedings of the International Symposium on Computer Engineering and Technology*, pages 127–132, 2010.

[33] Marta Beltrán, Miguel Calvo, and Sergio González. Federated System-to-Service Authentication and Authorization Combining PUFs and Tokens. In *Proceedings of the International Symposium on Reconfigurable Communication-centric Systems-on-Chip*, pages 1–8, 2017.

[34] E. Bertino and N. Islam. Botnets and Internet of Things Security. *Computer*, 50(02) :76–79, Feb 2017.

[35] Fábio Bezerra, Jacques Wainer, and Wil M. P. Aalst. Anomaly Detection Using Process Mining. volume 29, pages 149–161, 01 2009.

[36] Ani Bicaku, Silia Maksuti, Silke Palkovits-Rauter, Markus Tauber, Rainer Matischek, Christoph Schmittner, Georgios Mantas, Mario Thron, and Jerker Delsing. Towards Trustworthy End-to-End Communication in Industry 4.0. In *Proceedings of the International Conference on Industrial Informatics*, pages 889–896, 2017.

[37] Ani Bicaku, Christoph Schmittner, Markus Tauber, and Jerker Delsing. Monitoring Industry 4.0 applications for security and safety standard compliance. In *Proceedings of the Industrial Cyber-Physical Systems*, pages 749–754, 2018.

[38] Sergi Blanch-Torné, Fernando Cores, and Ramiro Moreno Chiral. Agent-based PKI for Distributed Control System. In *Proceedings of the World Congress on Industrial Control Systems Security*, pages 28–35, 2015.

[39] Gedare Bloom, Bassma Alsulami, Ebelechukwu Nwafor, and Ivan Cibrario Bertolotti. Design patterns for the industrial Internet of Things. In *Proceedings of the International Workshop on Factory Communication Systems*, pages 1–10, 2018.

[40] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A Training Algorithm for Optimal Margin Classifiers. In *Proceedings of the Annual Workshop on Computational Learning Theory*, pages 144–152, 1992.

[41] Leo Breiman. Random Forests. *Machine Learning*, 45(1) :5–32, 2001.

[42] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. LOF : Identifying Density-Based Local Outliers. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, page 93–104. Association for Computing Machinery, 2000.

[43] Kalle Burbeck and Simin Nadjm-Tehrani. Adwice–Anomaly Detection with Real-Time Incremental Clustering. In *Proceedings of the International Conference on Information Security and Cryptology*, pages 407–424. Springer, 2004.

[44] Adrien Bécue, Yannick Fourastier, Isabel Praça, Alexandre Savarit, Claude Baron, Baptiste Gradussofs, Etienne Pouille, and Carsten Thomas. CyberFactory#1 — Securing the

Industry 4.0 with Cyber-Ranges and Digital Twins. In *Proceedings of the International Workshop on Factory Communication Systems*, pages 1–4, 2018.

[45] C. Caroni. Outlier Detection by Robust Principal Components Analysis. *Communications in Statistics - Simulation and Computation*, 29(1) :139–151, 2000.

[46] Mete Çelik, Filiz Dadaşer-Çelik, and Ahmet Şakir Dokuz. Anomaly Detection in Temperature Data using DBSCAN Algorithm. In *Proccedings of the International Symposium on Innovations in Intelligent Systems and Applications*, pages 91–95. IEEE, 2011.

[47] S. Chauhan and L. Vig. Anomaly Detection in ECG Time Signals via Deep Long Short-term Memory Networks. In *Proceedings of the International Conference on Data Science and Advanced Analytics*, pages 1–7, Oct 2015.

[48] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, and Tadayoshi Kohno. Comprehensive Experimental Analyses of Automotive Attack Surfaces. In *Proceedings of the USENIX Security Symposium*, 2011.

[49] Min Chen, Yiming Miao, Yixue Hao, and Kai Hwang. Narrow Band Internet of Things. *IEEE Access*, 5 :20557–20577, 2017.

[50] Shanzhi Chen, Hui Xu, Dake Liu, Bo Hu, and Hucheng Wang. A Vision of IoT : Applications, Challenges, and Opportunities with China Perspective. *IEEE Internet of Things journal*, 1(4) :349–359, 2014.

[51] Michael W. Condry and Catherine Blackadar Nelson. Using Smart Edge IoT Devices for Safer, Rapid Response With Industry IoT Control Operations. *Proceedings of the IEEE*, 104(5) :938–946, 2016.

[52] Davide Conzon, Mohammad Rifat Ahmmad Rashid, Xu Tao, Angel Soriano, Richard Nicholson, and Enrico Ferrera. BRAIN-IoT : Model-Based Framework for Dependable Sensing and Actuation in Intelligent Decentralized IoT Systems. In *Proceedings of the International Conference on Computing, Communications and Security*, pages 1–8, 2019.

[53] Hui Cui, Robert H. Deng, Joseph K. Liu, Xun Yi, and Yingjiu Li. Server-Aided Attribute-Based Signature With Revocation for Resource-Constrained Industrial-Internet-of-Things Devices. *IEEE Transactions on Industrial Informatics*, 14(8) :3724–3732, 2018.

[54] R. Currie. Hacking the CAN Bus : Basic Manipulation of a Modern Automobile Through CAN Bus Reverse Engineering. Technical report, The SANS Institute, 2017.

[55] Ashok Kumar Das, Mohammad Wazid, Neeraj Kumar, Athanasios V. Vasilakos, and Joel J. P. C. Rodrigues. Biometrics-Based Privacy-Preserving User Authentication Scheme for Cloud-Based Industrial Internet of Things Deployment. *IEEE Internet of Things Journal*, 5(6) :4900–4913, 2018.

[56] Anh Dau, Vic Ciesielski, and Andy Song. Anomaly Detection Using Replicator Neural Networks Trained on Examples of One Class. pages 311–322, 12 2014.

[57] B. D. Deebak, Fadi Al-Turjman, Moayad Aloqaily, and Omar Alfandi. An Authentic-Based Privacy Preservation Protocol for Smart e-Healthcare Systems in IoT. *IEEE Access*, 7 :135632–135649, 2019.

[58] Ozgur Depren, Murat Topallar, Emin Anarim, and M Kemal Ciliz. An Intelligent Intrusion Detection System (IDS) for Anomaly and Misuse Detection in Computer Networks. *Expert systems with Applications*, 29(4) :713–722, 2005.

[59] Haimonti Dutta, Chris Giannella, Kirk Borne, and Hillol Kargupta. Distributed Top-K Outlier Detection from Astronomy Catalogs using the DEMAC System. 04 2007.

[60] Mohamed H. Eldefrawy, Nuno Pereira, and Mikael Gidlund. Key Distribution Protocol for Industrial Internet of Things Without Implicit Certificates. *IEEE Internet of Things Journal*, 6(1) :906–917, 2019.

[61] Widad Es-Soufi, Esma Yahia, and Lionel Roucoules. On the use of Process Mining and Machine Learning to Support Decision Making in Systems Design. In *Proceedings of the International Conference on Product Lifecycle Management*, volume AICT-492, pages 56–66. Springer, July 2016.

[62] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A Density-based Algorithm for Discovering Clusters a Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 226–231. AAAI Press, 1996.

[63] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *kdd*, volume 96, pages 226–231, 1996.

[64] European Commission. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance), 2016.

[65] C. Ferri, J. Hernández-Orallo, and R. Modroiu. An Experimental Comparison of Performance Measures for Classification. *Pattern Recognition Letters*, 30(1) :27–38, 2009.

[66] Evelyn Fix and J. L. Hodges. Discriminatory Analysis. Nonparametric Discrimination : Consistency Properties. *International Statistical Review / Revue Internationale de Statistique*, 57(3) :238–247, 1989.

[67] Francisco Fraile, Takuya Tagawa, Raul Poler, and Angel Ortiz. Trustworthy Industrial IoT Gateways for Interoperability Platforms and Ecosystems. *IEEE Internet of Things Journal*, 5(6) :4506–4514, 2018.

[68] Mabrouka Gmiden, Mohamed Hedi Gmiden, and Hafedh Trabelsi. Cryptographic and Intrusion Detection System for automotive CAN bus : Survey and contributions. In *Proceedings of the International Multi-Conference on Systems, Signals & Devices*, pages 158–163. IEEE, 2019.

[69] Nicolas Goix, Nicolas Drougard, Romain Brault, and Mael Chiapino. One Class Splitting Criteria for Random Forests. In Min-Ling Zhang and Yung-Kyun Noh, editors, *Proceedings of the Asian Conference on Machine Learning*, volume 77, pages 343–358. PMLR, Nov 2017.

[70] Bogdan Groza, Stefan Murvay, Anthony van Herrewege, and Ingrid Verbauwhede. LiBrA-CAN : A Lightweight Broadcast Authentication Protocol for Controller Area Networks. In *Proceedings of the International Conference on Cryptology and Network Security*, pages 185–200. Springer, 2012.

[71] Frank E Grubbs. Procedures for Detecting Outlying Observations in Samples. *Technometrics*, 11(1) :1–21, 1969.

[72] Guangjie Han, Hao Wang, Xu Miao, Li Liu, Jinfang Jiang, and Yan Peng. A Dynamic Multipath Scheme for Protecting Source-Location Privacy Using Multiple Sinks in WSNs Intended for IIoT. *IEEE Transactions on Industrial Informatics*, 16(8) :5527–5538, 2020.

[73] Md. Mahmud Hasan and Hussein T. Mouftah. Cloud-Centric Collaborative Security Service Placement for Advanced Metering Infrastructures. *IEEE Transactions on Smart Grid*, 10(2) :1339–1348, 2019.

[74] Mohammad Hassan. State of IoT 2022 : Number of Connected IoT Devices growing 18% to 14.4 Billion globally. `https://iot-analytics.com/number-connected-iot-devices/`. Accessed : 2022-07-5.

[75] Adrien Hemmer, Mohamed Abderrahim, Remi Badonnel, and Isabelle Chrisment. An Ensemble Learning-Based Architecture for Security Detection in IoT Infrastructures. In *Proceedings of the International Conference on Network and Service Management*, pages 180–186, 2021.

[76] Adrien Hemmer, Mohamed Abderrahim, Rémi Badonnel, Jérôme François, and Isabelle Chrisment. Comparative Assessment of Process Mining for Supporting IoT Predictive Security. *IEEE Transactions on Network and Service Management*, 18(1) :1092–1103, 2021.

[77] Adrien Hemmer, Remi Badonnel, and Isabelle Chrisment. A Process Mining Approach for Supporting IoT Predictive Security. In *Proceedings of the Network Operations and Management Symposium*, April 2020.

[78] Adrien Hemmer, Rémi Badonnel, Jérôme François, and Isabelle Chrisment. A Process Mining Tool for Supporting IoT Security. In *Proceedings of the Network Operations and Management Symposium*, pages 1–2, 2020.

[79] Andrea Hoeller and Ronald Toegl. Trusted Platform Modules in Cyber-Physical Systems : On the Interference Between Security and Dependability. In *Proceedings of the European Symposium on Security and Privacy Workshops*, pages 136–144, 2018.

[80] P. Holgado, V. A. Villagrá, and L. Vázquez. Real-Time Multistep Attack Prediction Based on Hidden Markov Models. *IEEE Transactions on Dependable and Secure Computing*, 17(1) :134–147, Jan 2020.

[81] Harold Hotelling. Analysis of a Complex of Statistical Variables into Principal Components. *Journal of Educational Psychology*, 24 :498–520, 1933.

[82] Peng Hu. A System Architecture for Software-Defined Industrial Internet of Things. In *Proceedings of the International Conference on Ubiquitous Wireless Broadband*, pages 1–5, 2015.

[83] Mia Hubert and Michiel Debruyne. Minimum Covariance Determinant. *WIREs Computational Statistics*, 2(1) :36–43, 2010.

[84] Philokypros Ioulianou, Vasileios Vasilakis, Ioannis Moscholios, and Michael Logothetis. A Signature-Based Intrusion Detection System for the Internet of Things. *Information and Communication Technology Form*, 2018.

[85] Xiaolin Jiang, Zhibo Pang, Michele Luvisotto, Fei Pan, Richard Candell, and Carlo Fischione. Using a Large Data Set to Improve Industrial Wireless Communications : Latency, Reliability, and Security. *IEEE Industrial Electronics Magazine*, 13(1) :6–12, 2019.

[86] Eszter Kail, Anna Banati, Erdödi Lászlo, and Miklós Kozlovszky. Security Survey of Dedicated IoT Networks in the Unlicensed ISM Bands. In *Proceedings of the International Symposium on Applied Computational Intelligence and Informatics*, pages 000449–000454, 2018.

[87] Arijit Karati, SK Hafizul Islam, and Marimuthu Karuppiah. Provably Secure and Lightweight Certificateless Signature Scheme for IIoT Environments. *IEEE Transactions on Industrial Informatics*, 14(8) :3701–3711, 2018.

[88] Sotirios Katsikeas, Konstantinos Fysarakis, Andreas Miaoudakis, Amaury Van Bemten, Ioannis Askoxylakis, Ioannis Papaefstathiou, and Anargyros Plemenos. Lightweight & Secure Industrial IoT Communications via the MQ Telemetry Transport Protocol. In *Proceedings of the Symposium on Computers and Communications*, pages 1193–1200. IEEE, 2017.

[89] Harald Klaus, Felicitas Hetzelt, Peter Hofmann, Andreas Blecker, and Daniela Schwaiger. Challenges and Solutions for Industry-Grade Secure Connectivity. In *Proceedings of the International Conference on Networked Systems*, pages 1–5. IEEE, 2019.

[90] Thomas Kobzan, Sebastian Schriegel, Simon Althoff, Alexander Boschmann, Jens Otto, and Jürgen Jasperneite. Secure and Time-sensitive Communication for Remote Process Control and Monitoring. In *Proceedings of the International Conference on Emerging Technologies and Factory Automation*, volume 1, pages 1105–1108, 2018.

[91] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas. DDoS in the IoT : Mirai and Other Botnets. *Computer*, 50(7) :80–84, 2017.

[92] Flip Korn, Alexandros Labrinidis, Yannis Kotidis, Christos Faloutsos, Alex Kaplunovich, and Dejan Perkovic. Quantifiable Data Mining using Principal Component Analysis. Technical report, 1998.

[93] Tanesh Kumar, An Braeken, Vidhya Ramani, Ijaz Ahmad, Erkki Harjula, and Mika Yliant-tila. SEC-BlockEdge : Security Threats in Blockchain-Edge based Industrial IoT Networks. In *Proccedings of the International Workshop on Resilient Networks Design and Modeling*, pages 1–7. IEEE, 2019.

[94] Ryo Kurachi, Yutaka Matsubara, Hiroaki Takada, Naoki Adachi, Yukihiro Miyashita, and Satoshi Horihata. CaCAN-centralized authentication system in CAN (controller area network). In *Proceedings of the International Conference on Embedded Security in Cars*, 2014.

[95] Fabian Kurtz, Caner Bektas, Nils Dorsch, and Christian Wietfeld. Network Slicing for Critical Communications in Shared 5G Infrastructures - An Empirical Evaluation. In *Proceedings of the Conference on Network Softwarization and Workshops*, pages 393–399, 2018.

[96] Heikki Laaki, Yoan Miche, and Kari Tammi. Prototyping a Digital Twin for Real Time Remote Control Over Mobile Networks : Application of Remote Surgery. *IEEE Access*, 7 :20325–20336, 2019.

[97] Tim Lackorzynski, Stefan Köpsell, and Thorsten Strufe. A Comparative Study on Virtual Private Networks for Future Industrial Communication Systems. In *Proceedings of the International Workshop on Factory Communication Systems*, pages 1–8, 2019.

[98] Jorma Laurikkala, Martti Juhola, Erna Kentala, N Lavrac, S Miksch, and B Kavsek. Informal Identification of Outliers in Medical Data. In *Proceedings of the International Workshop on Intelligent Data Analysis in Medicine and Pharmacology*, volume 1, pages 20–24. Citeseer, 2000.

[99] C. Lesjak, T. Ruprechter, J. Haid, H. Bock, and Eugen Brenner. A Secure Hardware Module and System Concept for Local and Remote Industrial Embedded System Identification. In *Proceedings of the Emerging Technology and Factory Automation*, pages 1–7, 2014.

[100] Christian Lesjak, Holger Bock, Daniel Hein, and Martin Maritsch. Hardware-Secured and Transparent Multi-Stakeholder Data Exchange for Industrial IoT. In *Proceedings of the International Conference on Industrial Informatics*, pages 706–713. IEEE, 2016.

[101] Christian Lesjak, Daniel Hein, Michael Hofmann, Martin Maritsch, Andreas Aldrian, Peter Priller, Thomas Ebner, Thomas Ruprechter, and Günther Pregartner. Securing Smart Maintenance Services : Hardware-Security and TLS for MQTT. In *Proceedings of the International Conference on Industrial Informatics*, pages 1243–1250. IEEE, 2015.

[102] Christian Lesjak, Thomas Ruprechter, Holger Bock, Josef Haid, and Eugen Brenner. ESTADO—Enabling Smart Services for Industrial Equipment through a Secured, Transparent and Ad-Hoc Data Transmission Online. In *Proceedings of the International Conference for Internet Technology and Secured Transactions*, pages 171–177. IEEE, 2014.

[103] Fagen Li, Jiaojiao Hong, and Anyembe Andrew Omala. Efficient Certificateless Access Control for Industrial Internet of Things. *Future Generation Computer Systems*, 76 :285–292, 2017.

[104] Giampaolo L Libralon, André C Carvalho, and Ana C Lorena. Ensembles of Pre-Processing Techniques for Noise Detection in Gene Expression Data. In *Proceedings of the International Conference on Neural Information Processing*, pages 486–493. Springer, 2008.

[105] Giampaolo Luiz Libralon, André Carlos Ponce de Leon Ferreira de Carvalho, and Ana Carolina Lorena. Pre-Processing for Noise Detection in Gene Expression Classification Data. *Journal of the Brazilian Computer Society*, 15(1) :3–11, 2009.

[106] Chao Lin, Debiao He, Xinyi Huang, Kim-Kwang Raymond Choo, and Athanasios V. Vasilakos. BSeIn : A Blockchain-Based Secure Mutual Authentication with Fine-Grained Access Control System for Industry 4.0. *Journal of Network and Computer Applications*, 116 :42–52, 2018.

[107] F. T. Liu, K. M. Ting, and Z. Zhou. Isolation Forest. In *Proceedings of the IEEE International Conference on Data Mining*, pages 413–422, Dec 2008.

[108] Yingbo Liu, Jiujun Cheng, Chendan Yan, Xiao Wu, and Fuzhen Chen. Research on the Matthews Correlation Coefficients Metrics of Personalized Recommendation Algorithm Evaluation. *International Journal of Hybrid Information Technology*, 8(1) :163–172, 2015.

[109] S. Lloyd. Least Squares Quantization in PCM. *IEEE Transactions on Information Theory*, 28(2) :129–137, March 1982.

[110] Zhendong Ma, Aleksandar Hudic, Abdelkader Shaaban, and Sandor Plosz. Security Viewpoint in a Reference Architecture Model for Cyber-Physical Production Systems. In *Proceedings of the European Symposium on Security and Privacy Workshops*, pages 153–159, 2017.

[111] Silia Maksuti, Ani Bicaku, Markus Tauber, Silke Palkovits-Rauter, Sarah Haas, and Jerker Delsing. Towards Flexible and Secure End-to-End Communication in Industry 4.0. In *Proceedings of the International Conference on Industrial Informatics*, pages 883–888, 2017.

[112] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, and Puneet Agarwal. Long Short Term Memory Networks for Anomaly Detection in Time Series. 04 2015.

[113] Daniel J. Mankowitz, Nir Levine, Rae Jeong, Yuanyuan Shi, Jackie Kay, Abbas Abdolmaleki, Jost Tobias Springenberg, Timothy Mann, Todd Hester, and Martin Riedmiller. Robust Reinforcement Learning for Continuous Control with Model Misspecification, 2019.

[114] Letizia Marchegiani and Ingmar Posner. Long-Term Driving Behaviour Modelling for Driver Identification. In *Proceedings of the International Conference on Intelligent Transportation Systems*, pages 913–919, 2018.

[115] Guido Marchetto, Riccardo Sisto, Jalolliddin Yusupov, and Adlen Ksentinit. Formally Verified Latency-Aware VNF Placement in Industrial Internet of Things. In *Proceedings of the International Workshop on Factory Communication Systems*, pages 1–9, 2018.

[116] Stefan Marksteiner. Reasoning on Adopting OPC UA for an IoT-Enhanced Smart Energy System from a Security Perspective. In *Proceedings of the Conference on Business Informatics*, volume 2, pages 140–143. IEEE, 2018.

[117] Bridget A Martin, Frank Michaud, Don Banks, Arsalan Mosenia, Riaz Zolfonoon, Susanto Irwan, Sven Schrecker, and John K Zao. OpenFog Security Requirements and Approaches. In *Proceedings of the Fog World Congress*, pages 1–6. IEEE, 2017.

[118] Adrian McEwen and Hakim Cassimally. *Designing the Internet of Things*. John Wiley & Sons, 2013.

[119] David W McKee, Stephen J Clement, Jaber Almutairi, and Jie Xu. Survey of Advances and Challenges in Intelligent Autonomy for Distributed Cyber-Physical Systems. *CAAI Transactions on Intelligence Technology*, 3(2) :75–82, 2018.

[120] David Wesley McKee, Stephen J Clement, Jaber Almutairi, and Jie Xu. Massive-Scale Automation in Cyber-Physical Systems : Vision & Challenges. In *Proceedings of the International Symposium on Autonomous Decentralized System*, pages 5–11. IEEE, 2017.

[121] Nay Myat Min, Vasaka Visoottiviseth, Songpon Teerakanok, and Nariyoshi Yamai. OWASP IoT Top 10 based Attack Dataset for Machine Learning. In *Proceedings of the International Conference on Advanced Communication Technology*, pages 317–322, 2022.

[122] Haralambos Mouratidis and Vasiliki Diamantopoulou. A Security Analysis Method for Industrial Internet of Things. *IEEE Transactions on Industrial Informatics*, 14(9) :4093–4100, 2018.

[123] Imanol Mugarza, Andoni Amurrio, Ekain Azketa, and Eduardo Jacob. Dynamic Software Updates to Enhance Security and Privacy in High Availability Energy Management Applications in Smart Cities. *IEEE Access*, 7 :42269–42279, 2019.

[124] Robert Müller, Steffen Illium, Thomy Phan, Tom Haider, and Claudia Linnhoff-Popien. Towards Anomaly Detection in Reinforcement Learning. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, pages 1799–1803, 2022.

[125] Emmanuel Müller, Matthias Schiffer, and Thomas Seidl. Statistical Selection of Relevant Subspace Projections for Outlier Ranking. In *Proceedings of the International Conference on Data Engineering*, pages 434–445, 2011.

[126] Alexandre Nairac, Neil Townsend, Roy Carr, Steve King, Peter Cowley, and Lionel Tarassenko. A System for the Analysis of Jet Engine Vibration Data. *Integrated Computer-Aided Engineering*, 6(1) :53–66, 1999.

[127] Jürgen Neises, George Moldovan, Thomas Walloschke, and Bianca Popovici. Trustworthiness in Supply Chains : A Modular Extensible Approach Applied to Industrial IoT. In *Proceedings of the Global Internet of Things Summit*, pages 1–6. IEEE, 2020.

[128] Matthias Niedermaier, Florian Fischer, and Alexander von Bodisco. PropFuzz — An IT-Security Fuzzing Framework for Proprietary ICS Protocols. In *Proceedings of the International Conference on Applied Electronics*, pages 1–4, 2017.

[129] Stefan Nürnberger and Christian Rossow. Vatican–Vetted, Authenticated CAN Bus. In *Proceedings of the International Conference on Cryptographic Hardware and Embedded Systems*, pages 106–124. Springer, 2016.

[130] M. Pahl and L. Donini. Securing IoT Microservices with Certificates. In *Proceedings of the IEEE/IFIP Network Operations and Management Symposium*, pages 1–5, April 2018.

[131] Swapnil Paliwal. Hash-Based Conditional Privacy Preserving Authentication and Key Exchange Protocol Suitable for Industrial Internet of Things. *IEEE Access*, 7 :136073–136093, 2019.

[132] Lucas Parra, Gustavo Deco, and Stefan Miesbach. Statistical Independence and Novelty Detection with Information Preserving Nonlinear Maps. *Neural Computation*, 8(2) :260–269, 1996.

[133] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, et al. Scikit-learn : Machine learning in python. *Journal of Machine Learning Research*, 12(85) :2825–2830, 2011.

[134] Davy Preuveneers, Wouter Joosen, and Elisabeth Ilie-Zudor. Data Protection Compliance Regulations and Implications for Smart Factories of the Future. In *Proceedings of the International Conference on Intelligent Environments*, pages 40–47. IEEE, 2016.

[135] Peter Priller, Andreas Aldrian, and Thomas Ebner. Case Study : From Legacy to Connectivity Migrating Industrial Devices into the World of Smart Services. In *Proceedings of the Emerging Technology and Factory Automation*, pages 1–8, 2014.

[136] L. Rabiner and B. Juang. An Introduction to Hidden Markov Models. *IEEE ASSP Magazine*, 3(1) :4–16, 1986.

[137] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. Efficient Algorithms for Mining Outliers from Large Data Sets. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 427–438, 2000.

[138] Martin Roesch. Snort - Lightweight Intrusion Detection for Networks. In *Proceedings of the USENIX Conference on System Administration*, LISA '99, page 229–238, USA, 1999. USENIX Association.

[139] Loïc Rouch, Jérôme François, Frédéric Beck, and Abdelkader Lahmadi. A Universal Controller to Take Over a Z-Wave Network. In *Proceedings of Black Hat Europe*, pages 1–9, 2017.

[140] Ahmad-Reza Sadeghi, Christian Wachsmann, and Michael Waidner. Security and Privacy Challenges in Industrial Internet of Things. In *Proceedings of the Design Automation Conference*, pages 1–6. IEEE, 2015.

[141] Sven Schrecker, Hamed Soroush, J Molina, J LeBlanc, F Hirsch, M Buchheit, A Ginter, R Martin, H Banavara, S Eswarahally, et al. Industrial Internet of Things Volume G4 : Security Framework. *Industrial Internet Consortium*, pages 1–173, 2016.

[142] Julian Schuette and Gerd Stefan Brost. LUCON : Data Flow Control for Message-Based IoT Systems. In *Proceedings of the International Conference On Trust, Security And Privacy In Computing And Communications/ IEEE International Conference On Big Data Science And Engineering*, pages 289–299, 2018.

[143] Bernhard Schölkopf, John C. Platt, John Shawe-Taylor, Alex J. Smola, and Robert C. Williamson. Estimating the Support of a High-Dimensional Distribution. *Neural Computation*, 13(7) :1443–1471, 2001.

[144] Emiliano Sisinni, Abusayeed Saifullah, Song Han, Ulf Jennehag, and Mikael Gidlund. Industrial Internet of Things : Challenges, Opportunities, and Directions. *IEEE Transactions on Industrial Informatics*, 14(11) :4724–4734, 2018.

[145] Borut Sluban, Dragan Gamberger, and Nada Lavra. Advances in Class Noise Detection. In *Proceedings of the European Conference on Artificial Intelligence*, pages 1105–1106. IOS Press, 2010.

[146] Koen Tange, Michele De Donno, Xenofon Fafoutis, and Nicola Dragoni. A Systematic Survey of Industrial Internet of Things Security : Requirements and Fog Computing Opportunities. *IEEE Communications Surveys & Tutorials*, 22(4) :2489–2520, 2020.

[147] A. Ukil, S. Bandyoapdhyay, C. Puri, and A. Pal. IoT Healthcare Analytics : The Importance of Anomaly Detection. In *Proceedings of the 30th International Conference on Advanced Information Networking and Applications*, pages 994–997, March 2016.

[148] Thomas Ulz, Thomas Pieber, Christian Steger, Sarah Haas, and Rainer Matischek. Sneakernet on Wheels : Trustworthy NFC-Based Robot to Machine Communication. In *Proceedings of the International Conference on RFID Technology & Application*, pages 260–265, 2017.

[149] Jo Van Bulck, Jan Tobias Mühlberg, and Frank Piessens. VulCAN : Efficient Component Authentication and Software Isolation for Automotive Control Networks. In *Proceedings of the Annual Computer Security Applications Conference*, pages 225–237, 2017.

[150] W. M. P. VAN DER AALST. The Application of Petri Nets to Workflow Management. *Journal of Circuits, Systems and Computers*, 08(01) :21–66, 1998.

[151] Wil MP van der Aalst, Vladimir Rubin, Boudewijn F van Dongen, Ekkart Kindler, and Christian W Günther. Process mining : A two-step approach using transition systems and regions. *BPM Center Report BPM-06-30, BPMcenter. org*, 6, 2006.

[152] W.M.P. van der Aalst, A.J. Bolt Iriondo, and S.J. van Zelst. *RapidProM : Mine your Processes and not just your Data*. Chapman & Hall/CRC Press, 2018.

[153] Juan Vanerio and Pedro Casas. Ensemble-Learning Approaches for Network Security and Anomaly Detection. In *Proceedings of the Workshop on Big Data Analytics and Machine Learning for Data Communication Networks*, pages 1–6, 2017.

[154] Vitaly Simonovich. Imperva blocks our largest ddos l7/brute force attack ever (peaking at 292,000 rps). `https://www.imperva.com/blog/imperva-blocks-our-largest-ddos-l7-brute-force-attack-ever-peaking-at-292000-rps/`. Last visited on September 2020.

[155] Cynthia Wagner, Alexandre Dulaunoy, Gérard Wagener, and Andras Iklody. MISP : The Design and Implementation of a Collaborative Threat Intelligence Sharing Platform. In *Proceedings of the ACM on Workshop on Information Sharing and Collaborative Security*, pages 49–56. ACM, 2016.

[156] Kevin Wallis, Florian Kemmer, Eugen Jastremskoj, and Christoph Reich. Adaption of a Privilege Management Infrastructure (PMI) Approach to Industry 4.0. In *Proceedings of the International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, pages 101–107, 2017.

[157] Mark Walport. The Internet of Things :making the most of the Second Digital Revolution. `https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/409774/14-1230-internet-of-things-review.pdf`. Accessed : 2022-07-5.

[158] Qiyan Wang and Sanjay Sawhney. VeCure : A practical security framework to protect the CAN bus of vehicles. In *Proceddings of the 2014 International Conference on the Internet of Things*, pages 13–18. IEEE, 2014.

[159] C. Warrender, S. Forrest, and B. Pearlmutter. Detecting Intrusions using System Calls : Alternative Data Models. In *Proceedings of the Symposium on Security and Privacy*, pages 133–145, 1999.

[160] Edgar Weippl and Peter Kieseberg. Security in Cyber-Physical Production Systems : A Roadmap to Improving IT-security in the Production System Lifecycle. In *Proceedings of the International Annual Conference*, pages 1–6, 2017.

[161] Samuel Woo, Hyo Jin Jo, and Dong Hoon Lee. A Practical Wireless Attack on the Connected Car and Security Protocol for In-Vehicle CAN. *IEEE Transactions on Intelligent Transportation Systems*, 16(2) :993–1006, 2014.

[162] Qiao Yan, Wenyao Huang, Xupeng Luo, Qingxiang Gong, and F. Richard Yu. A Multi-Level DDoS Mitigation Framework for the Industrial Internet of Things. *IEEE Communications Magazine*, 56(2) :30–36, 2018.

[163] Xingjie Yu and Huaqun Guo. A Survey on IIoT Security. In *Proceedings of the Asia Pacific Wireless Communications Symposium*, pages 1–5. IEEE, 2019.

[164] Andrea Zanella, Nicola Bui, Angelo Castellani, Lorenzo Vangelista, and Michele Zorzi. Internet of Things for Smart Cities. *IEEE Internet of Things journal*, 1(1) :22–32, 2014.

[165] Hongming Zhang, Ke Sun, Bo Xu, Linglong Kong, and Martin Müller. A Simple Unified Framework for Anomaly Detection in Deep Reinforcement Learning. *arXiv preprint arXiv :2109.09889*, 2021.

[166] Ji Zhang, Qigang Gao, and Hai Wang. SPOT : A System for Detecting Projected Outliers From High-dimensional Data Streams. In *Proceedings of the International Conference on Data Engineering*, pages 1628–1631, 2008.

[167] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. BIRCH : An Efficient Data Clustering Method for Very Large Databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 103–114, New York, NY, USA, 1996. ACM.

[168] Lu Zhou, Kuo-Hui Yeh, Gerhard Hancke, Zhe Liu, and Chunhua Su. Security and Privacy for the Industrial Internet of Things : An Overview of Approaches to Safeguarding Endpoints. *IEEE Signal Processing Magazine*, 35(5) :76–87, 2018.

Résumé

Cette thèse porte sur de nouvelles méthodes de détection pour la sécurité des systèmes IoT hétérogènes, et s'inscrit dans le cadre du projet européen SecureIoT. Nous avons tout d'abord proposé une solution utilisant le *process mining* couplé à un pré-traitement des données, pour construire des modèles de comportement et identifier des anomalies à partir de données hétérogènes. Nous avons évalué cette solution à partir de jeux de données issus de plusieurs domaines d'applications différents : véhicules connectés, industrie 4.0, robots d'assistance. Cette solution permet de construire des modèles plus facilement compréhensibles. Elle obtient des meilleurs résultats de détection que d'autres méthodes usuelles, mais demande un temps de traitement plus long. Pour réduire ce dernier sans dégrader les performances de détection, nous avons ensuite étendu notre méthode à l'aide d'une approche ensembliste, qui permet de combiner les résultats de plusieurs méthodes de détection utilisées simultanément. En particulier, nous avons comparé différentes stratégies d'agrégation des scores. Nous avons aussi évalué un mécanisme permettant d'ajuster dynamiquement la sensibilité de la détection. Enfin, nous avons implanté la solution sous la forme d'un prototype, qui a été intégré à une plateforme de sécurité développée avec des partenaires européens.

Mots-clés: cybersécurité, gestion de la sécurité, intelligence artificielle, Internet des objets.

Abstract

This thesis concerns new detection methods for the security of heterogenous IoT systems, and fits within the framework of the SecureIoT european project. We have first proposed a solution exploiting the process mining together with pre-treatment techniques, in order to build behavioral models, and identifying anomalies from heterogenous systems. We have then evaluated this solution from datasets coming from different application domains : connected cars, industry 4.0, and assistance robots.. This solution enables to build models that are more easily understandable. It provides better detection results than other common methods, but may generate a longer detection time. In order to reduce this time without degrading detection performances, we have then extended our method with an ensemble approach, which combines the results from several detection methods that are used simultaneously. In particular, we have compared different score aggregation strategies, as well as evaluated a feedback mechanism for dynamically adjusting the sensitivity of the detection. Finally, we have implemented the solution as a prototype, that has been integrated into a security plateform developed in collaboration with other european industrial partners.

Keywords: cybersecurity, security management, artificial intelligence, Internet of Things.