

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par l'Institut National des Sciences Appliquées de Toulouse

Présentée et soutenue par

Malcolm BOURDON

Le 29 juin 2021

Détection d'intrusion basée sur l'analyse de compteurs matériels pour des objets connectés

Ecole doctorale : **SYSTEMES**

Spécialité : **Informatique**

Unité de recherche :
LAAS - Laboratoire d'Analyse et d'Architecture des Systèmes

Thèse dirigée par
Mohamed KAANICHE et Eric ALATA

Jury

Mme Maryline LAURENT, Rapporteure
Mme Isabelle CHRISMENT, Rapporteure
M. Gilles GRIMAUD, Examinateur
M. Youssef LAAROUCHI, Examinateur
M. Pierre-François GIMENEZ, Examinateur
M. Eric TOTEL, Examinateur
M. Mohamed KAANICHE, Directeur de thèse
M. Eric ALATA, Co-directeur de thèse

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ FÉDÉRALE TOULOUSE MIDI-PYRÉNÉES

Délivré par :
l'Institut National des Sciences Appliquées de Toulouse (INSA de Toulouse)

Présentée et soutenue le *29/06/2021* **par :**

MALCOLM BOURDON

Détection d'intrusion basée sur l'analyse de compteurs matériels pour des objets connectés

---

JURY

MARYLINE LAURENT	Professeure des Universités	Rapporteuse
ISABELLE CHRISMENT	Professeure des Universités	Rapporteuse
ERIC TOTEL	Professeur des Universités	Examinateur
GILLES GRIMAUD	Professeur des Universités	Examinateur
PIERRE-FRANÇOIS GIMENEZ	Maître de Conférences	Examinateur
YOUSSEF LAAROUCHI	Ingénieur de recherche	Encadrant industriel
ERIC ALATA	Maître de Conférences	Co-directeur de thèse
MOHAMED KAÂNICHE	Directeur de recherche	Directeur de thèse

École doctorale et spécialité :
EDSYS : Informatique 4200018
Unité de Recherche :
Laboratoire d'analyse et d'architecture des systèmes
Directeur(s) de Thèse :
Mohamed Kaâniche et *Eric Alata*
Rapporteurs :
Marilyne Laurent et *Isabelle Chrisment*

Remerciements

J'aimerais remercier beaucoup de personnes qui m'ont aidé à aller au bout de cette thèse.

Tout d'abord je tiens à remercier mon encadrant EDF R&D dans le cadre de cette thèse CIFRE, Youssef Laarouchi, qui m'a accompagné, soutenu et a cru en moi tout au long de cette thèse et du stage qui a précédé.

Je tiens également à remercier Mohamed Kaâniche, mon directeur de thèse, qui malgré ses nombreuses responsabilités au sein du LAAS a su m'aider et m'accompagner pour rester pertinent dans ma thèse.

Je remercie aussi Eric Alata, mon co-directeur de thèse, qui m'a guidé et grandement aidé techniquement. Et je remercie énormément Vincent Nicomette qui m'a suivi, accompagné et soutenu pendant toute la thèse et m'as aidé à persévérer jusqu'au bout. Je remercie également Vincent Migliore qui m'a également suivi pendant cette thèse.

Je remercie plus largement toute l'équipe d'EDF, qui m'ont tous très bien accueilli et suivi pendant toute la thèse. Ils ont montré de l'intérêt dans mes travaux en plus de partager leurs connaissances et leur bonne humeur.

Je remercie de la même manière l'équipe et surtout tous les doctorants du LAAS avec lesquels j'ai pu discuter et partager durant la thèse. Je n'oublierai jamais toutes les rencontres, les moments partagés et les différentes discussions que j'ai pu y avoir. Et je remercie particulièrement tous ceux qui m'ont aidé techniquement et moralement.

Je remercie les rapporteuses de ma thèse ainsi que les examinateurs, qui ont su poser des questions pertinentes sur les tenants et les aboutissants de mes travaux.

Pour finir je remercie toute ma famille et mes amis, qui m'ont permis d'avoir des moments de détente pour pouvoir ensuite me focaliser sur mes travaux sereinement.

Globalement cette expérience fut rude mais grâce à toutes les personnes qui m'ont entouré j'ai trouvé la motivation et le courage d'aller jusqu'au bout, merci à vous.

Table des matières

Introduction **1**

1 Internet des Objets : contexte et contraintes **5**
1.1 Internet des Objets et sécurité 5
1.1.1 Concept de l'internet des objets 5
1.1.2 Terminologie de la sécurité 9
1.1.3 Historique d'attaques 13
1.2 Internet des Objets pour l'industrie et cas d'usage 15
1.2.1 Description 15
1.2.2 Attaques sur les systèmes industriels 17
1.2.3 Contexte applicatif et hypothèses 20
1.3 Conclusion 23

2 De l'analyse de risque au modèle de menace **25**
2.1 Introduction 25
2.1.1 Méthode EBIOS 26
2.1.2 Cas d'étude 28
2.2 Analyse de risques 30
2.2.1 Étude du contexte 30
2.2.2 Étude des évènements redoutés 33
2.2.3 Étude des scénarios de menaces 34
2.2.4 Étude des risques 36
2.3 Modèle de menace 37
2.4 Conclusion 40

3 Etat de l'art **41**
3.1 Approches de protection 41
3.1.1 Fonction non clonable physiquement 42
3.1.2 Attestation à distance 44
3.1.3 Diversification Logicielle 47
3.1.4 Détection d'intrusion 50
3.1.5 Conclusion des approches relativement aux hypothèses 52
3.2 Taxonomie de la détection d'anomalie 53
3.2.1 Spécification 55
3.2.2 Détection statistique 56
3.2.3 Détection basée sur de l'apprentissage automatique 57
3.3 Conclusion 60

4 Détection d'anomalies du comportement du processeur 63
4.1 Les compteurs matériels de performance 63
4.1.1 Description 64
4.1.2 Les compteurs matériels pour la sécurité 65
4.1.3 Les critiques sur l'utilisation des compteurs 68
4.2 Une approche mêlant HPC et détection d'anomalies 70
4.2.1 Vue d'ensemble 70
4.2.2 Lecture 71
4.2.3 Collecte 73
4.2.4 Extraction des caractéristiques 73
4.2.5 Standardisation 74
4.2.6 Détection d'anomalies 75
4.3 Choix des algorithmes de détection de valeurs aberrantes 76
4.3.1 Généralités 76
4.3.2 Algorithmes basés sur les plus proches voisins 77
4.3.3 Algorithmes basés sur la classification 79
4.3.4 Algorithmes basés sur du regroupement 80
4.4 Conclusion 81

5 Expérimentations 83
5.1 Mise en place et validation de la plateforme 83
5.1.1 Environnement matériel et logiciel 84
5.1.2 Tests préliminaires 85
5.1.3 Preuve de concept 87
5.1.4 Amélioration de l'approche et des expérimentations 92
5.2 Mise à l'échelle et améliorations des expérimentations 95
5.2.1 Nouvelle plateforme d'expérimentations 95
5.2.2 Analyses et résultats 98
5.2.3 Conclusion et discussions 106
5.3 Conclusion 108

6 Conclusion 111
6.1 Bilan 111
6.2 Perspectives 112

Bibliographie 115

A Analyse EBIOS 125
A.1 Métriques utilisées 125
A.2 Biens essentiels 126

Liste des figures

1.1 Une représentation de l'IdO d'après l'ENISA de [ENISA 2017] 8
1.2 L'arbre de la sécurité de fonctionnement 10
1.3 Sondage sur les challenges de l'IoT de 2015 18

2.1 La représentation des différents modules EBIOS 27
2.2 Schéma résumant les différentes communications avec l'objet étudié . 29

3.1 Schéma montrant une PUF utilisant le principe de "course" d'après les travaux de [Maes 2010] 44
3.2 Schéma montrant les différentes étapes de l'attestation à distance d'après les travaux de [Steiner 2016] 45
3.3 Les catégories de différentes recherches en diversification logiciel des travaux selon [Larsen 2014] 49

4.1 Une vue d'ensemble de notre approche 72
4.2 Un exemple de mesures avec des valeurs légitimes (en orange, au centre) et des valeurs aberrantes (en bleu, tout autour) 77

5.1 Évolution de l'évènement INST_RETIRED en fonction de l'évènement BUS_ACCESS pour différents programmes communs 86
5.2 Évolution de l'évènement L1D_CACHE_REFILL en fonction de l'évènement BR_MIS_PRED pour différents programmes communs 86
5.3 Évolution de l'évènement INST_RETIRED en fonction de l'évènement BUS_ACCESS pour différents paramétrage d'un logiciel simple et répétitif 88
5.4 Évolution de l'évènement L1D_CACHE_REFILL en fonction de l'évènement BR_MIS_PRED pour différents paramétrage d'un logiciel simple et répétitif 89
5.5 Moyennes de l'évènement L1D_Refill en fonction du numéro de la trace. Les attaques sont aux indices 71 à 80 et les points rouges ont été identifiés comme anomalies par LOF. 90
5.6 Médianes de l'évènement L1D_Refill en fonction du numéro de la trace. Les attaques sont aux indices 71 à 80 et les points rouges ont été identifiés comme anomalies par LOF. 91
5.7 Addition des scores LOF pour les caractéristiques seules. Les attaques sont aux indices 71 à 80. 92
5.8 Maximums de l'évènement EXC_TAKEN en fonction des maximums de l'évènement BR_MIS_PRED 93
5.9 Score de l'algorithme LOF exécuté sur toutes les caractéristiques des traces 93

5.10 Photographie de la baie composée de 100 Raspberry Pi 94
5.11 Graphique ROC pour le logiciel légitime S1 99
5.12 Graphique ROC pour le logiciel légitime S2 100
5.13 Graphique ROC pour le logiciel légitime S3 100
5.14 Graphique ROC pour le logiciel légitime S4 101

Liste des tableaux

2.1 Relation entre biens supports et biens essentiels : fonctionnalités . . 32
2.2 Relation entre biens supports et biens essentiels : données 32
2.3 Gravité de l'impact en cas de manquement au critère de sécurité . . 34
2.4 Niveau de vraisemblance de mise à exécution des menaces 37
2.5 Matrice des risques 38

3.1 Résumé des solutions étudiées 53

5.1 Sensibilité de détection en fonction du taux de compromission des objets, en utilisant le calibrage fait avec un logiciel différent, et pour 0%, 0,1% et 0,5% d'attaques. Rouge : TPR ≤50%. Orange : TPR ≤80% or FPR ≥1%. 103
5.2 Sensibilité de détection face au taux de compromission des objets, en utilisant le calibrage fait avec un logiciel différent, et pour 1%, 2%, et 3% d'attaques. Rouge : TPR ≤50%. Orange : TPR ≤80% or FPR ≥1%. 104
5.3 efficacité de détection avec 1% d'objets compromis par charge malveillante. Rouge : TPR ≤50%. Orange : TPR ≤80%. 105
5.4 efficacité de détection avec 1% d'objets compromis par charge malveillante. Rouge : TPR ≤50%. Orange : TPR ≤80%. 105
5.5 Complexités des différents algorithmes. n le nombre de traces, k le nombre de plus proche voisins. 106

A.1 Métrique du critère de disponibilité 125
A.2 Métrique du critère d'intégrité 125
A.3 Métrique du critère de confidentialité 125
A.4 Métrique du critère de gravité 126
A.5 Métrique du critère de vraisemblance 126

Introduction

Le déploiement massif des objets connectés, formant l'Internet des Objets ou IoT (pour *Internet of Things*) est aujourd'hui en plein essor. Ces objets envahissent notre quotidien professionnel mais aussi familial. Si on peut trouver un intérêt ludique à leur utilisation, leur véritable valeur ajoutée se trouve ailleurs. En effet, la mise en commun des informations recueillies par d'innombrables objets disséminés dans le monde, ainsi que l'attrait économique de leur exploitation sont les principaux arguments mis en avant par les constructeurs d'objets et les défenseurs de ce concept. Cependant, les technologies liées à l'IoT et l'intégration massive des objets connectés soulèvent plusieurs problématiques notamment vis-à-vis de la sécurité informatique. Tout d'abord, ces objets sont souvent conçus et développés très rapidement sans réel soucis de la sécurité, et constituent donc une cible de choix pour les attaquants visant à s'introduire dans le réseau interne d'un environnement protégé ou à récupérer des informations personnelles. Par ailleurs, ces objets sont en général mobiles et peuvent donc grandement faciliter la propagation d'un logiciel malveillant au sein d'un environnement connecté. Ceci est d'autant plus vrai que les mécanismes de protection de nos réseaux classiques ne sont pas bien adaptés à la gestion de ces objets, potentiellement corrompus, qui peuvent se déplacer d'un environnement à un autre. Il est donc difficile à la fois d'empêcher la corruption de ces objets (qui communiquent massivement avec des protocoles sans fil dont certains sont très peu sécurisés) et de détecter leur compromission. En effet les mécanismes de surveillance sont compliqués à mettre en œuvre dans ces environnements où les objets n'ont que peu de ressources disponibles à dédier pour la sécurité.

Les environnements industriels n'échappent pas à cet engouement de l'Internet des objets. De plus en plus d'environnements s'équipent de capteurs variés, dont les informations peuvent être collectées pour améliorer la gestion des bâtiments par exemple, ou pour optimiser l'efficacité d'une ferme photo-voltaïque. Tout comme les environnements personnels, les environnements industriels connectés ont déjà été la cible de multiples attaques et certains programmes malveillants tels que Stuxnet sont aujourd'hui célèbres. Les travaux menés dans cette thèse se situent justement dans ce contexte, et plus spécifiquement le contexte de la gestion de l'énergie, ces travaux de thèse étant financés par EDF. Le contexte considéré est celui du déploiement massif d'objets connectés, qui ont pour objectif d'aider à la gestion de l'énergie (que ce soit dans les domiciles ou dans les nouveaux systèmes impliqués dans une gestion intelligente de l'énergie, communément appelés les smart grids). Ces objets peuvent être victimes d'attaques et il est crucial de mettre en œuvre des moyens permettant d'empêcher ces intrusions mais aussi de les détecter. Il est en effet illusoire de penser que toute attaque peut être empêchée, compte tenu des spécificités des objets connectés que nous avons décrites ci-dessus. Les travaux menés dans cette thèse visent à répondre à cette problématique de la détection d'intrusions pour ce type d'objets, et propose une solution originale basée sur l'utilisation de

compteurs matériels de performance des processeurs. Cette approche est originale dans le sens où elle est "légère", ne nécessitant pas de modification des logiciels s'exécutant sur l'objet, et dans le sens où elle ne nécessite pas de modélisation complexe du comportement des logiciels mis en œuvre sur des objets légitimes, ce qui pourrait s'avérer trop coûteux lors des mises à jour des objets notamment.

Le présent manuscrit s'articule autour de cinq chapitres principaux. Le premier chapitre présente le contexte de nos travaux, en introduisant tout d'abord les grands principes de l'Internet des objets puis en décrivant quelques attaques qui ciblent spécifiquement ces objets. Nous poursuivons ensuite ce chapitre en nous attardant sur les spécificités de l'Internet des Objets dans un contexte industriel et en décrivant quelques attaques célèbres qui ont spécifiquement ciblé ces environnements. Nous terminons ce chapitre en présentant le cas d'étude précis de nos travaux, en lien avec la gestion de l'énergie et en partenariat avec EDF et nous dressons une liste de nos hypothèses de travail ainsi que des principaux objectifs de la solution de sécurité que nous souhaitons déployer.

Le second chapitre présente une première analyse de risques concernant un exemple d'objet connecté représentatif des objets qui sont déployés dans notre cas d'étude. Cette analyse a été menée dans un but de mieux mettre en évidence le modèle de menace associé à ce type d'objets et à ces déploiements spécifiques. La méthode utilisée dans ce chapitre est EBIOS. Nous la déroulons pour arriver à une étude de risques qui nous permet ensuite de mieux cerner notre modèle de menaces.

Le troisième chapitre présente un état de l'art des différents travaux de recherche qui nous semblent de bons candidats pour améliorer la sécurité des objets connectés envisagés dans notre contexte. La diversification logicielle, les fonctions inclonables physiquement, l'attestation à distance et la détection d'intrusion sont ainsi présentées. Ce chapitre, compte tenu de nos hypothèses et de nos objectifs, conclut que la détection d'intrusion est la méthode qui nous semble la plus adaptée à nos travaux et nous en présentons ainsi les grands principes ainsi que les principaux travaux associés.

Le chapitre quatre présente une nouvelle approche permettant de réaliser la détection d'anomalies pour des objets connectés dans un environnement industriel. Cette approche est basée sur la collecte à distance et l'analyse des compteurs matériels de performance des processeurs qui équipent les différents objets. Cette approche, qui ne nécessite ni de modifier les objets, ni de réaliser au préalable un modèle de comportement des objets légitimes, permet de détecter au plus tôt des objets corrompus, dans une flotte d'objets identiques massivement déployés. Les grandes étapes de cette approche sont détaillés dans ce chapitre. Elles consistent en 1) la collecte de compteurs ; 2) l'extraction de caractéristiques à partir de ces valeurs : 3) une standardisation de ces caractéristiques ; et 4) l'utilisation d'algorithmes de détection de valeurs aberrantes afin de détecter les objets corrompus, c'est-à-dire- dont le comportement dévie significativement de celui des autres objets de la flotte.

Le cinquième chapitre présente deux expérimentations qui ont été mises en

œuvre pour valider notre approche. La première expérimentation a été menée à une échelle réduite et avait pour objectif de vérifier la pertinence de notre approche. La seconde expérimentation, à une plus grande échelle, a été menée de façon à valider que notre approche pouvait réaliser efficacement la détection d'intrusions sur un grand nombre d'objets (10 000 environ). Cette expérimentation nous a également permis de tester différents algorithmes de détection de valeurs aberrantes et de pouvoir ainsi déterminer ceux qui donnent les meilleurs résultats dans notre contexte.

Le dernier chapitre de ce manuscrit conclut nos travaux et propose quelques perspectives à plus ou moins long terme. Ces travaux de thèse ont donné lieu à la publication de deux articles dans des conférences internationales ([Bourdon] et [Bourdon 2020]) et un dépôt de brevet.

Chapitre 1

Internet des Objets : contexte et contraintes

Sommaire

1.1 Internet des Objets et sécurité 5
1.1.1 Concept de l'internet des objets 5
1.1.2 Terminologie de la sécurité 9
1.1.3 Historique d'attaques 13
1.2 Internet des Objets pour l'industrie et cas d'usage 15
1.2.1 Description 15
1.2.2 Attaques sur les systèmes industriels 17
1.2.3 Contexte applicatif et hypothèses 20
1.3 Conclusion 23

L'Internet des Objets (IdO), aussi connu sous le nom anglais *Internet of Things* (IoT), a pour origine l'ajout d'une fonctionnalité de communication à un objet, qui au départ n'a pas nécessairement vocation à communiquer. Grâce à cette nouvelle fonctionnalité, l'objet devient capable d'interagir avec l'utilisateur, avec d'autres objets ou avec des serveurs distants. Cela permet de faciliter l'utilisation et/ou d'étendre les possibilités d'utilisation de ces objets. Mais l'ajout de cette fonctionnalité n'est que le point de départ car l'IdO désigne aujourd'hui un concept plus complet et complexe. Dans ce chapitre, nous présentons en premier l'évolution de l'Internet des Objets et introduisons certaines problématiques de sécurité de cette technologie. Nous spécifions ensuite le contexte industriel et des cas d'usages que nous étudierons dans cette thèse, en soulignant l'importance de la sécurité au travers notamment de la présentation de certaines attaques.

1.1 Internet des Objets et sécurité

Nous commençons donc dans cette partie à introduire succinctement le concept d'Internet des objets, avant d'adresser le domaine étudié dans cette thèse : la sécurité informatique, parfois aussi appelée "cybersécurité".

1.1.1 Concept de l'internet des objets

Le terme d'Internet des Objets, bien qu'apparu initialement au début des années 2000, commence par un principe simple que l'on peut observer dès les années 80 :

l'ajout d'une fonctionnalité de communication à un objet. En effet, dès 1982, il est fait mention d'un distributeur de boisson dont il était possible de récupérer l'état (vide ou non) à distance. Cela montre comment la fonctionnalité de communications a été utilisée au départ : la collecte d'informations à distance. Cette notion de capteur communicant a ensuite été reprise dans de nombreux travaux. On retrouve dans la littérature par exemple le terme de *Réseau de capteur sans-fil*, ou *Wireless sensor Network* (WSN) en anglais. Il s'agit en effet de gestion d'un réseau de capteurs communiquant à distance, comme expliqué et montré dans le travail d'Al-Karaki et al. [Al-Karaki 2004]. Les capteurs ayant peu de capacités, il est principalement question d'optimisation de communications et d'utilisation des ressources à travers une adaptation logicielle et matérielle. Il est ainsi possible, grâce à la connexion des capteurs, de centraliser les informations simplement et rapidement. Cette idée est déjà un sujet discuté dans les années 90. En effet, on peut retrouver dans les travaux d'Al-Karaki et al. mentions de tels réseaux dans le domaine médical, militaire, ou encore la détection d'intrusion. Ces capteurs sans fils coopèrent donc afin de faire transiter les informations sur le réseau jusqu'à atteindre une passerelle permettant de centraliser les informations recueillies.

Cependant, les capteurs ne représentent en fait qu'une petite partie de ce que l'on peut appeler aujourd'hui IdO. L'IdO correspond à un écosystème complet, comprenant une myriade d'objets et appareils connectés et/ou de serveurs distants, coopérant tous pour fournir des services nouveaux, automatiques ou optimisés. De tels environnements connectés sont en général appelés intelligents, ou pour utiliser le terme anglais répandu : "*smart*". On retrouve donc les termes de maison intelligente, bâtiment intelligent, quartier ou ville intelligents, industrie intelligente, etc. Les utilisateurs de tels environnements peuvent également avoir un rôle central, en interagissant avec différents objets et en choisissant des préférences d'utilisation. Par exemple un bâtiment intelligent peut optimiser sa consommation en énergie, grâce à la récupération de certaines informations de capteurs. Ces informations peuvent être la présence de personnes dans le bâtiment, l'ensoleillement ou les habitudes des occupants. En couplant ces informations avec les différents actionneurs permettant de contrôler lumières, aérations et chauffage, il est ainsi possible d'optimiser l'utilisation d'énergie de tout un bâtiment. On peut par exemple citer le *Mansion ZCB*, un des premiers bâtiments écologiques sans émission de carbone situé à Honk-Kong, et équipé de près de 2800 capteurs. Pour donner un autre exemple, une ville intelligente comme Singapour ou Zurich [1] optimise la circulation en gérant intelligemment les feux rouges, les transports, ou en partageant des informations quant à la circulation dans la ville pour proposer par la suite des services d'itinéraires.

Pour que l'on puisse qualifier un "objet" d'intelligent, il faut que celui-ci s'adapte à son environnement et possède une certaine autonomie. Cela se traduit en général par l'utilisation des informations issues de capteurs pour par exemple s'adapter à des changements. Un certain nombre de fonctionnalités sont nécessaires afin de

1. `https://easyelectriclife.groupe.renault.com/fr/tendances/territoires/smart-cities-top-5-des-villes-les-plus-intelligentes/`

développer un écosystème tel qu'envisage l'IdO. Il est ainsi possible de définir une vue d'ensemble simplifiée via différentes fonctionnalités :

— Les interactions directes avec l'environnement, qui se fait via des capteurs et des actionneurs.

— La gestion des communications entre les différents objets, pour stocker des informations ou relayer les ordres par exemple, qui se fait généralement via une passerelle.

— La prise de décision en fonction des différentes informations disponibles ainsi que des potentielles préférences des utilisateurs.

— Les interactions entre les différents objets et avec les utilisateurs.

Certaines de ces fonctionnalités peuvent être rassemblées dans un seul objet. Cette représentation simplifiée permet d'avoir une vue d'ensemble, mais il est aussi possible de trouver des représentations plus complexes et détaillées. Par exemple, le schéma 1.1 de l'ENISA (*European Union Agency for Cybersecurity* ou Agence européenne chargée de la sécurité des réseaux et de l'information) montre une classification des différentes fonctionnalités, objets, services et technologies qui peuvent composer un environnement IdO. Ce schéma a été fait dans le cadre d'un document [ENISA 2017] publié en 2017 et ayant pour but de poser les fondations d'une vision unifiée de l'IdO. Pour cela, ce document propose de faire une analyse de risque pour évaluer les besoins en sécurité et guider les acteurs de l'IoT. Ce schéma décrit ainsi une représentation plus complexe des objets connectés à considérer pour pouvoir aborder le thème de la sécurité et de la gestion des risques de manière globale.

Plusieurs facteurs facilitent le développement et la réalisation du concept global de l'IdO :

— Le développement et la baisse des prix des microprocesseurs. Ceux-ci deviennent en effet de plus en plus performants pour un prix attractif, permettant de développer et de faciliter la collecte et la transmission d'informations.

— Le développement de protocoles de communication sans-fil moins exigeants en ressources de la part de l'objet que le Wi-Fi classique. Ils peuvent ainsi s'adapter à des situations plus variées, courte ou longue distance, fréquences de communications, débit, etc. On peut par exemple citer *Zigbee*[2], *Sigfox*[3], *LoRaWAN*[4] ou *Bluetooth*.

— Le développement de logiciels spécifiques plus adaptés et optimisés, comme les systèmes d'exploitation qui deviennent de plus en plus légers, conçus spécifiquement pour de tels microprocesseurs et de tels usages ; on peut citer par exemple TinyOS[5], Zephyr[6] ou ChibiOS[7].

2. https://zigbeealliance.org
3. https://www.sigfox.com
4. https://lora-alliance.org/about-lorawan
5. http://www.tinyos.net
6. https://www.zephyrproject.org
7. https://www.chibios.org

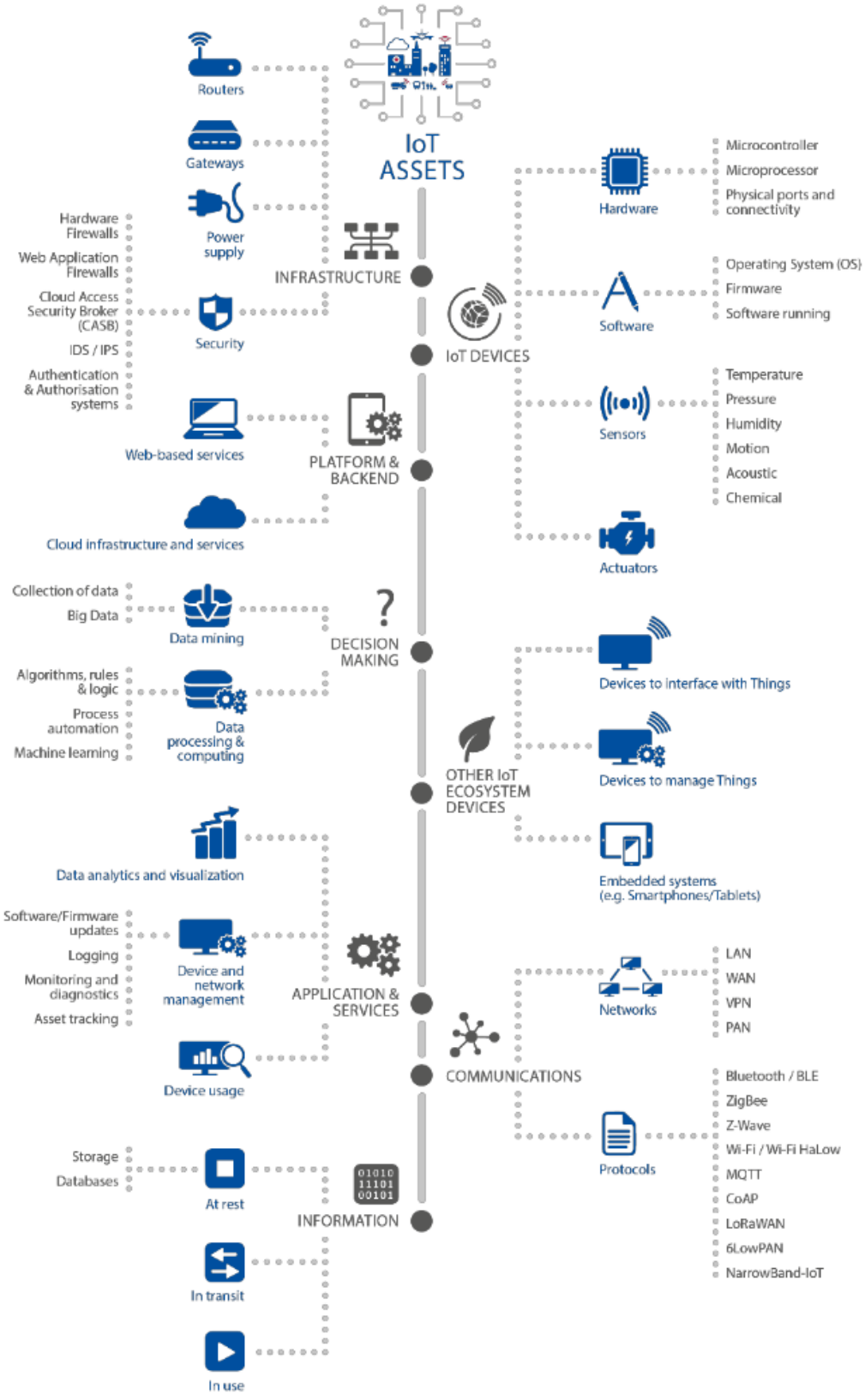


FIGURE 1.1 – Une représentation de l'IdO d'après l'ENISA de [ENISA 2017]

De plus en plus d'entreprises voient ainsi un intérêt à connecter divers objets, afin de diversifier leur gamme de produits en proposant des fonctionnalités nouvelles ou différentes. Gartner prévoyait pour 2020 près de 5,8 milliards d'objets connectés. Cependant l'IdO a souffert de sa grande diversité. En effet, un principe très important de l'IdO est la coopération. Mais l'hétérogénéité des services, objets, appareils, architectures ou technologies utilisés rend la tâche complexe. Et certaines entreprises créant des objets connectés ne possèdent pas forcément les compétences requises en informatique afin d'ajouter des fonctionnalités fiables bénéficiant de la connectivité ajoutée aux objets initialement construits. [8] Ainsi, un certain nombre de standards, normes et groupements d'acteurs sont donc apparus dans le but de créer une cohérence dans la conception des environnements connectés [9]. Néanmoins les environnements intelligents sont encore en phase de développement. De plus, certains constructeurs n'utilisent pas les standards consciemment afin d'avoir le contrôle sur leur environnement. On peut notamment souligner un certain manque de maturité en ce qui concerne le sujet que nous allons développer dans la suite : la sécurité.

1.1.2 Terminologie de la sécurité

La sûreté de fonctionnement définit les termes de sécurité-innocuité (safety) et sécurité-immunité (security), dans laquelle nos travaux s'inscrivent. Nous présentons ici le vocabulaire qui sera utilisé par la suite, en commençant par définir les principaux concepts de la sûreté de fonctionnement, issus de [Laprie 1996] et mis à jour dans [Avizienis 2004], puis en se focalisant sur les concepts liés à la sécurité-immunité.

1.1.2.1 Sûreté de fonctionnement

La sûreté de fonctionnement d'un système informatique est définie comme « la propriété qui permet aux utilisateurs d'un système de placer une confiance justifiée dans le service qu'il leur délivre ». Le service délivré correspond au comportement du système perçu par ses utilisateurs. La sûreté de fonctionnement comporte trois axes principaux : les attributs qui la décrivent, les entraves qui empêchent sa réalisation et les moyens d'atteindre celle-ci (figure 1.2).

La sûreté de fonctionnement associée à la sécurité-immunité définissent les attributs et les propriétés complémentaires suivantes :

— La capacité d'un système à être prêt à l'utilisation se définit comme la **Disponibilité**.

— La continuité du service se définit comme la **Fiabilité**.

— La non-occurrence de conséquences catastrophiques pour l'environnement se définit comme la **Sécurité-innocuité**.

8. https://www.gartner.com/en/newsroom/press-releases/2019-08-29-gartner-says-5-8-billion-enterprise-and-automotive-io

9. https://www.postscapes.com/internet-of-things-protocols

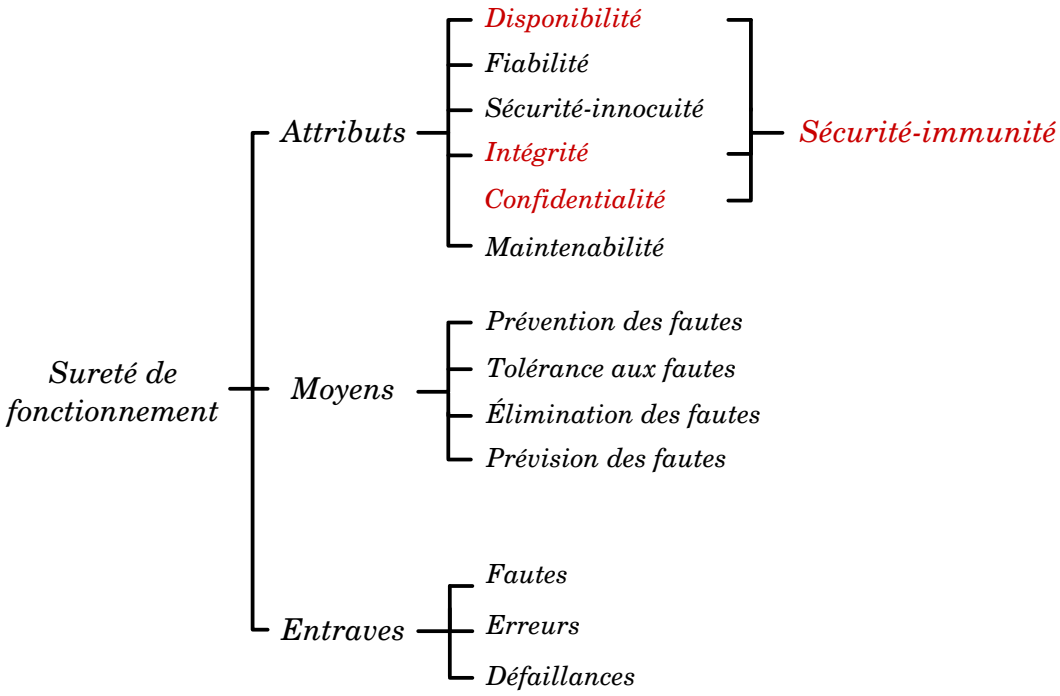


FIGURE 1.2 – L'arbre de la sécurité de fonctionnement

— La non-occurrence de divulgations non autorisées de l'information se définit comme la **Confidentialité**.

— La non-occurrence d'altérations inappropriées de l'information se définit comme l'**Intégrité**.

— L'aptitude d'un système à être réparé ou à subir des évolutions se définit comme la **Maintenabilité**.

La non-sûreté de fonctionnement correspond quant à elle à une perte de confiance, qui ne peut plus ou ne pourra plus être placée dans le service délivré. Les causes ou résultats de celle-ci sont les circonstances indésirables définies comme étant les *entraves* :

— Une **défaillance** survient lorsque le service délivré dévie de l'accomplissement de la fonction du système.

— Une **erreur** est la partie de l'état du système qui est susceptible d'entraîner une défaillance.

— Une **faute** est la cause adjugée ou supposée d'une erreur.

Pour mettre en place la sûreté de fonctionnement dans un système, nous disposons de plusieurs *moyens*. Ces moyens sont des méthodes et techniques permettant de fournir au système l'aptitude à délivrer un service conforme à l'accomplissement de sa fonction, et de donner confiance dans cette aptitude. Le développement d'un système sûr passe par l'utilisation combinée d'un ensemble de méthodes :

— **La prévention de fautes** empêche l'occurrence ou l'introduction de fautes.

— **La tolérance aux fautes** permet de fournir un service à même de remplir la fonction du système en dépit des fautes.
— **L'élimination des fautes** réduit la présence (nombre, sévérité) des fautes.
— **La prévision des fautes** estime la présence, la création et les conséquences des fautes.

Les travaux décrits et présentés dans ce manuscrit s'inscrivent dans le cadre de la sécurité-immunité, qui se définit comme l'association des attributs de disponibilité, de confidentialité et d'intégrité. Dans la suite, nous détaillons les concepts liés à la sécurité-immunité.

1.1.2.2 Sécurité-immunité

La sécurité-immunité a pour but de protéger un système contre les fautes définies comme intentionnellement nuisibles, c'est-à-dire créées ou commises délibérément pour nuire, appelées aussi *malveillances*. Dans cette section, nous allons appliquer les attributs génériques définis précédemment au contexte de la sécurité-immunité. Dans le reste de ce manuscrit, les termes de sécurité ou de sécurité informatique seront utilisés comme alias à la sécurité-immunité, sauf exception explicitement précisée.

Attributs Dans le contexte de la sécurité-immunité, les attributs de la sûreté de fonctionnement peuvent être spécialisés de la manière suivante :

— **Disponibilité** : prévention de rétentions d'information non autorisées.
— **Confidentialité** : prévention de divulgations d'information non autorisées.
— **Intégrité** : prévention de modifications d'information non autorisées.

Malveillances Le projet MAFTIA [Powell 2001] s'intéresse au concept de faute due à l'homme pour la sécurité-immunité. Dans nos travaux, nous nous concentrons principalement sur ces fautes, et notamment à deux classes de fautes intentionnellement nuisibles, appelées également *malveillances* : les logiques malignes et les intrusions. Les logiques malignes sont des fautes internes intentionnelles, qui sont conçues pour provoquer des dégâts ou pour faciliter les futures intrusions par l'ajout de vulnérabilités. Celles-ci peuvent être présentes dès la première utilisation du système, donc ajoutées par le concepteur du système, ou durant son exploitation. Les intrusions quant à elles peuvent être associées à deux causes :

— Un acte malveillant ou une *attaque* essayant d'exploiter une faiblesse du système. Une attaque étant une faute d'interaction, dont le but est de violer un ou plusieurs des attributs de sécurité. Elle peut être aussi définie comme une tentative d'intrusion.
— Une faiblesse ou une *vulnérabilité* placée dans les exigences, la spécification, la conception ou la configuration du système, ou dans la manière dont il est utilisé. Une vulnérabilité étant une faute accidentelle, ou une faute intentionnellement malveillante ou non malveillante.

L'intrusion se définit donc comme étant une faute externe nuisible qui résulte d'une attaque ayant réussi à exploiter une vulnérabilité.

Fautes dues à l'homme sans volonté de nuisance Le projet MAFTIA décrit également deux autres types de fautes dues à l'homme n'étant pas des malveillances, car sans volonté de nuisance, qui peuvent cependant être génératrices de vulnérabilités ou d'intrusions dans les systèmes :

— Les fautes de conception sont ajoutées à la conception du système, ce sont des fautes de développement accidentelles ou intentionnelles sans volonté de nuire.

— Les fautes d'interaction sont des fautes externes, dues à la mauvaise utilisation du système.

1.1.2.3 Surfaces d'attaques et modèle de menaces

Pour permettre de mettre en place les moyens associés à un système sûr de fonctionnement, il est nécessaire de comprendre deux éléments constitutifs du système que nous cherchons à protéger :

— **Modèle de menaces** : définit les menaces potentielles, de telle manière que les vulnérabilités d'un système soient identifiées pour proposer des moyens permettant de prévenir, tolérer, éliminer ou prévoir des malveillances.

— **Surface(s) d'attaques** : définit les vecteurs d'attaques, c'est-à-dire les points d'accès extérieurs au système pouvant contenir des vulnérabilités qu'un attaquant pourrait exploiter pour réaliser une intrusion.

Tous les attributs définis dans cette partie décrivent ainsi les objectifs et caractéristiques en général recherchés dans la sécurité informatique. Lorsque l'on étudie un système d'information que l'on veut sécuriser ou dont on veut évaluer la sécurité, il faut avoir une bonne connaissance du système afin de pouvoir choisir et adapter les solutions de sécurité en fonction des risques et des besoins. Pour cela il est possible de mener une analyse de risque, c'est-à-dire dans notre cas une étude du systèmes vis-à-vis des malveillances qui permet de décrire méthodiquement un système et les risques qu'il peut encourir.

Les risques dépendent ainsi des capacités des attaquants envisagés, des potentielles vulnérabilités du système ainsi que de la surface d'attaque, c'est-à-dire des différents points d'entrée possibles pour l'attaquant dans le système. Nos travaux portent sur des solutions de sécurité, nous nous focaliserons ainsi progressivement sur le type de système que nous allons étudier. Nous mènerons alors une analyse de risque qui nous aidera à comprendre le type de système étudié et les risques potentiels, ainsi que les différentes hypothèses et modèle de menaces associés à nos travaux. Cependant nous allons dans un premier temps montrer l'importance de la sécurité dans le domaine des objets connectés à travers la description des principales attaques et de leurs impacts.

1.1.3 Historique d'attaques

Les objets connectés, quoique facilitant la vie des utilisateurs et la gestion des environnements dits "intelligents", ont subi un certains nombre d'attaques, dont certaines connues de part leur virulence ou leur impact, que ce soit sur le réseau ou dans le monde physique.

On peut par exemple citer le virus *Mirai* [Kolias 2017]. Cette célèbre attaque a profité de l'accès distant (notamment par telnet) et de configurations simples par défaut afin d'avoir accès à de nombreux objets pour les infecter. Pour être plus précis, l'attaque possède une structure classique de Botnet avec un maître et des esclaves. Les esclaves sont les objets infectés et le maître est l'orchestrateur de l'attaque. Le virus essaie d'abord d'infecter d'autres objets en scannant le réseau qui lui est accessible. Pour cela il essaie d'utiliser des identifiants et configurations par défaut pour se connecter aux dits objets. S'en suit, en cas de réussite, une phase de reconnaissance pour essayer d'identifier l'objet ciblé et remonter les informations au maître. Le maître choisit, avec les informations collectées, comment infecter l'objet. Une fois les nouveaux objets infectés et contrôlés par l'attaquant, celui-ci peut télécharger sur ces objets les données nécessaires à une attaque de type DDoS, c'est-à-dire une attaque inondant une cible spécifique de messages et empêchant donc les communications légitimes avec la cible. Les caractéristiques notables de cette attaque sont une simplicité d'exécution et une grande vitesse de propagation, entraînant une redoutable efficacité de l'attaque DDoS. On peut par exemple citer des sites internet célèbres tels que Github, Twitter, Reddit, Netflix ou encore Airbnb qui ont subi des attaques DDoS par de tels botnets.

La simplicité et l'efficacité de l'attaque par force brute utilisant identifiants et configurations par défaut souligne malheureusement, à ce moment-là, un sérieux manque de sensibilisation à la cybersécurité dans le domaine des objets connectés. En effet les constructeurs, concepteurs et utilisateurs de ces objets ne sont pas suffisamment sensibilisés aux problématiques de sécurité ou n'ont pas les compétences requises pour implémenter des mécanismes de défense adéquats pour leurs objets. De même les utilisateurs de ces objets ne sont pas suffisamment sensibilisés aux problèmes de sécurité et ne modifient pas les configurations par défaut dans la plupart des cas, car ils ne réalisent pas les risques encourus. Par la suite, de nombreux virus se sont inspirés de cette méthode d'exécution, et essaient notamment de profiter des configurations par défaut des objets connectés. De très nombreuses caméras connectées ont été touchées par ce type d'attaque, ce qui peut également représenter une menace pour la vie privée des utilisateurs. Dans le même ordre d'idée, certaines attaques avaient pu avoir accès à des jouets pour enfants [10] ou à des baby-phones [11].

Une autre attaque ayant eu un impact significatif concerne la prise de contrôle d'un véhicule connecté [Miller 2015]. Cette attaque souligne les problèmes que l'on peut rencontrer, liés au contrôle à distance et à la connexion des appareils. En effet, Charlie Miller et Chris Valasek se sont intéressés à la sécurité de certains

10. https://www.pandasecurity.com/en/mediacenter/tips/hacked-toy/
11. https://www.kaspersky.fr/blog/babyphone-hack/5140/

véhicules connectés. Ils ont tout d'abord cherché à comprendre le fonctionnement de la voiture en identifiant les différents composants. Ils ont pu ainsi trouver un moyen de changer le programme de la voiture grâce à une clé USB. Le véhicule proposant une fonctionnalité permettant de créer un hotspot, c'est-à-dire une passerelle pour un réseau privé connecté à Internet, ils ont également pu avoir accès au programme de la voiture grâce à cette fonctionnalité. Enfin ils ont étudié la connexion cellulaire et constaté qu'il était possible de mettre à jour à distance les véhicules de cette manière, de telle façon que n'importe quel véhicule pouvait communiquer avec tous les véhicules connectés via le réseau cellulaire. Il est donc possible de créer un ver, c'est-à-dire un programme malveillant qui peut se répandre de façon autonome de véhicule en véhicule. L'impact d'une telle attaque aurait pu être désastreux. Un attaquant pouvait avoir le contrôle de la voiture et ainsi provoquer des accidents. Le constructeur a donc rappelé tous les véhicules concernés, près de 1,4 millions de véhicules. Cet évènement eut donc de sérieuses conséquences pour le constructeur du point de vue de l'image de marque mais également financièrement.

La dernière attaque que nous présentons succinctement ici est *Stuxnet* [Langner 2011] en 2010. Cette attaque visait des systèmes industriels iraniens afin de ralentir leur programme nucléaire. Bien qu'effectuée dans un système fermé de centrale nucléaire, l'attaque a pu corrompre le système informatique afin de modifier le comportement de centrifugeuses qui ont fini par exploser. Pour être plus précis, plusieurs failles informatiques alors inconnues (dites *zero day*), ont été exploitées, menant à la corruption de systèmes d'exploitation afin de prendre le contrôle de la partie contrôle et commande du système. Enfin les centrifugeuses ont été modifiées et le système a été corrompu afin de désactiver la levée d'alertes. Comme le système d'information est fermé, c'est-à-dire non connecté à Internet, les attaquants ont pénétré le système via une clé USB infectée qui a été introduite dans le système d'information. Cela montre bien que les systèmes industriels fermés peuvent aussi être victimes d'attaques ciblées, qui sont organisées et peuvent être dévastatrices.

Ces différentes attaques montrent bien que les objets connectés peuvent être attaqués sur plusieurs fronts, et pour diverses raisons. Ces attaques peuvent avoir un impact négatif important sur les concepteurs des objets vulnérables, en terme financier et d'image de marque, ou avoir de lourdes conséquences sur l'environnement direct auquel les objets sont rattachés, jusqu'à mettre en danger des vies humaines.

Des trois attaques présentées précédemment, Mirai est celle qui montre le plus un manque de sensibilisation de la part des constructeurs et des utilisateurs. En effet, cette attaque a pris avantage des configurations (identifiant et mot de passe) par défaut des objets connectés pour pouvoir s'y connecter et en prendre le contrôle. Or, il est de bonnes pratiques pour les concepteurs d'éviter ce genre de situation, en mettant par exemple un mot de passe aléatoire à changer dès la première connexion ou utilisation de l'objet. Il est également de bonne pratique pour l'utilisateur de changer le mot de passe de son objet connecté dès la première connexion, ce qui aurait rendu l'attaque inopérante.

Ce manque de sensibilisation, particulièrement évident aux débuts de l'IdO, est

dû au développement rapide et au concept même de l'IdO. En effet, dans ces environnements connectés, les objets sont particulièrement hétérogènes et sont développés ou conçus par de nombreuses entités différentes. Étant un concept toujours en développement, les standards et autres documentations ou bonnes pratiques n'était encore il y a quelques années que peu ou pas formés, rendant plus complexes les interactions entre les différents acteurs. Bien que la situation s'améliore aujourd'hui et que l'on peut maintenant remarquer le développement de standards et de technologies prenant plus en compte la sécurité, certains des concepteurs ne possèdent pas encore les compétences nécessaires pour créer un objet répondant aux problématiques de sécurité. De plus, le marché des objets connectés étant en pleine expansion, de nombreuses entreprises se sont essayées hâtivement à cette idée, en déployant rapidement des objets à bas coût pour être parmi les premiers sur le marché. L'IdO a ainsi encore une certaine maturité à acquérir [12], que ce soit au niveau des projets d'un ordre plus général ou de la sécurité [Bauer 2017]. Par exemple, les approches standards ou classiques de la sécurité ne sont pas directement applicable aux environnements IdO, le réseau et les capacités de calculs et de communications étant différents. Par exemple un thermostat connecté ne supportera pas un antivirus classique à cause du stockage des signatures des attaques et des scans de fichiers que cela peut demander, sans parler du fait que ce thermostat peut ne pas avoir de système d'exploitation mais plutôt être basé sur un *firmware*. Il faut donc proposer ou adapter des solutions de sécurité pour de tels environnements.

Dans cette partie, nous avons ainsi présenté l'Internet des objets et la sécurité, séparément puis conjointement. Maintenant que nous avons introduit ces deux composantes principales de la thèse, nous allons préciser le contexte et les cas d'usage qui vont nous intéresser plus particulièrement dans cette thèse. Cela nous permettra d'étudier par la suite des solutions de sécurité adaptées à notre contexte.

1.2 Internet des Objets pour l'industrie et cas d'usage

Maintenant que nous avons dépeint succinctement le concept de l'IdO, sa diversité et ses problématiques en ce qui concerne la sécurité, nous précisons dans la suite le contexte industriel dans lequel nos travaux se sont inscrits. Après avoir défini ce contexte, nous détaillerons certaines attaques qui ont affecté le domaine industriel.

1.2.1 Description

L'Internet industriel des objets est un terme qui désigne l'utilisation du concept de l'IdO pour différents cas d'usage industriels comme la logistique, le transport, la fabrication automatisée, l'énergie, la sidérurgie et la métallurgie, l'aviation, les systèmes pétroliers. Grâce à la coopération entre différents objets via les communications machine à machine (M2M), il est possible d'avoir une production connectée

12. `https://iotbusinessnews.com/2020/04/16/04740-lessons-from-the-front-line-of-iot-real-world-deployments-show-a-frustrating-lack-of-ambition/`

et intelligente. De plus, avec le développement du *Cloud Computing*, c'est-à-dire le stockage, l'utilisation et l'accès aux données par l'intermédiaire d'Internet, il est possible de délocaliser certaines fonctionnalités comme l'analyse des données et de faciliter le contrôle et la surveillance à distance.

Dans le monde industriel et l'automatisme, on utilise souvent des automates programmables industriels (ou Programmable Logic Controler, PLC en Anglais), similaires à des ordinateurs, disposant d'entrées/sorties, et permettant d'automatiser et de contrôler des machines industrielles, pompe, éclairage, etc. Ces PLC étaient, à leur création en 1968, destinées à des automaticiens et non à des informaticiens. Cependant, l'informatique au service de l'industrie se développe et depuis 1986, ces appareils sont de plus en plus connectés à des ordinateurs plus classiques avec lesquels ils peuvent interagir avec des protocoles de communication standards. S'ensuit le développement de la coopération des objets grâce à l'évolution des techniques de communications de machine à machine (M2M) à partir de 1997. Cependant, un peu de la même manière que pour l'IdO classique, il faut attendre un certain nombre d'avancées technologiques supplémentaires avant de voir émerger le concept actuel de l'Internet industriel des objets :

— Le développement de standards, notamment pour les communications, afin d'avoir des protocoles et des communications sécurisés et optimisés, utilisant moins de ressources pour les appareils connectés.

— Le développement des batteries, les rendant plus économiques et efficaces

— Le développement de l'énergie solaire, rendant certains appareils isolés autonomes

— La miniaturisation, l'amélioration et la baisse de coûts matériels ; les appareils ont ainsi plus de capacités à moindre coûts, rendant leur utilisation plus rentable pour les industries.

C'est ainsi qu'en 2016, on évoque et envisage la vision actuelle de l'industrie connectée, dont le principe est donc de tirer partie d'un grand nombre de données des objets connectés peu coûteux. Grâce aux avancées dans le domaine du *big data* et grâce à l'utilisation d'algorithmes d'intelligence artificielle, ces données peuvent être analysées afin d'optimiser et améliorer les différents processus industriels.

Les systèmes cyber-physiques (*cyber physical systems*, CPS en anglais), désignent les systèmes agissant sur l'environnement physique, tout en intégrant des composants logiciels et de communication. Ils permettent ainsi d'agir sur l'environnement physique en s'adaptant à la situation en fonction des données captées. Il devient donc possible de surveiller et de contrôler les systèmes et processus industriels de manière plus optimisée et efficace grâce à leur coopération. Mais il est aussi possible d'améliorer les automatismes et de faciliter la maintenance. En effet, les industries peuvent être moins sujets à des erreurs humaines et réduire le travail manuel, permettant un gain de temps, d'argent, et une réduction des risques physiques que peuvent encourir les employés dans certains cas. Par exemple, certains automatismes d'interruption de systèmes en cas de danger remonté par les capteurs peuvent améliorer la sécurité physique du système et des employés. De plus, grâce

au partage rapide des informations et des alertes, il est possible de réagir promptement aux anomalies, (à défaut d'interrompre le système entier) et de réduire le temps d'interruption du système.

Pour donner un exemple, dans le cas du transport de marchandises, le référencement des différentes marchandises ainsi que le suivi des différents modes de transport peuvent se révéler fastidieux et complexes. Mais un système connecté et automatique, permettant de suivre le cheminement des marchandises et des transports plus facilement, est une composante essentielle des entreprises de transport. De plus, grâce à des algorithmes d'intelligence artificielle qui pourraient être reliés à tout l'environnement connecté du transport, il est possible de construire un système dynamique, mêlant automatismes et décisions humaines, pouvant s'adapter rapidement et efficacement à des imprévus.

Cependant, la mise en place d'environnements connectés et intelligents doit toujours faire face à un certain nombre de défis, dont certains ont déjà été évoqués. La figure 1.3 représentant un sondage fait en 2015 auprès de 200 dirigeants d'entreprises nous montrent les inquiétudes qui émergent concernant l'IdO pour l'industrie. On retrouve ainsi un certain nombre de critères évoqués précédemment, dans les problématiques auxquelles l'IdO d'une manière plus générale fait face. Nous pouvons également souligner que la cybersécurité est en première ligne des préoccupations. Ces inquiétudes concernant l'importance de la cybersécurité dans les systèmes industriels seront en effet confirmées après 2015 à travers différentes cyber-attaques que nous allons décrire succinctement dans la prochaine section.

1.2.2 Attaques sur les systèmes industriels

Nous avons présenté précédemment, dans un contexte plus général, de célèbres attaques dans le cadre des objets connectés. Nous avons ainsi introduit l'attaque *Stuxnet*, ciblant des systèmes industriels nucléaires iraniens avec un réseau informatique fermé. Cette attaque est considérée comme précurseur d'autres attaques telles que *Black energy* (2015) [13], *Industroyer* (2016) [Cherepanov 2017], *Triton* (2017) [14] [Di Pinto 2018] ou *notPetya* (2017) [McQuade 2018] qui ont toutes ciblé des systèmes industriels et que nous présentons dans la suite. Nous allons aussi présenter le cas de Norsk Hydro (2019) [15] attaque plus récente qui, contrairement à d'autres, a été assumée par l'entreprise victime qui a partagé son expérience.

1.2.2.1 Black energy et KillDisk (2015)

Plusieurs attaques ont eu lieu en Ukraine sur des systèmes informatiques en 2015 et 2016. Ces attaques semblent être ciblées et ont plus particulièrement visé le

13. https://www.welivesecurity.com/2016/01/03/blackenergy-sshbeardoor-details-2015-attacks-ukrainian-news-media-electric-industry/

14. https://www.industrie-techno.com/article/le-recit-par-schneider-electric-de-triton-l-attaque-qui-a-fait-trembler-l-industrie.57306

15. https://radiflow.com/wp-content/uploads/2019/03/RF-Security-brief-Norsk-Hydro-V4-090619.pdf

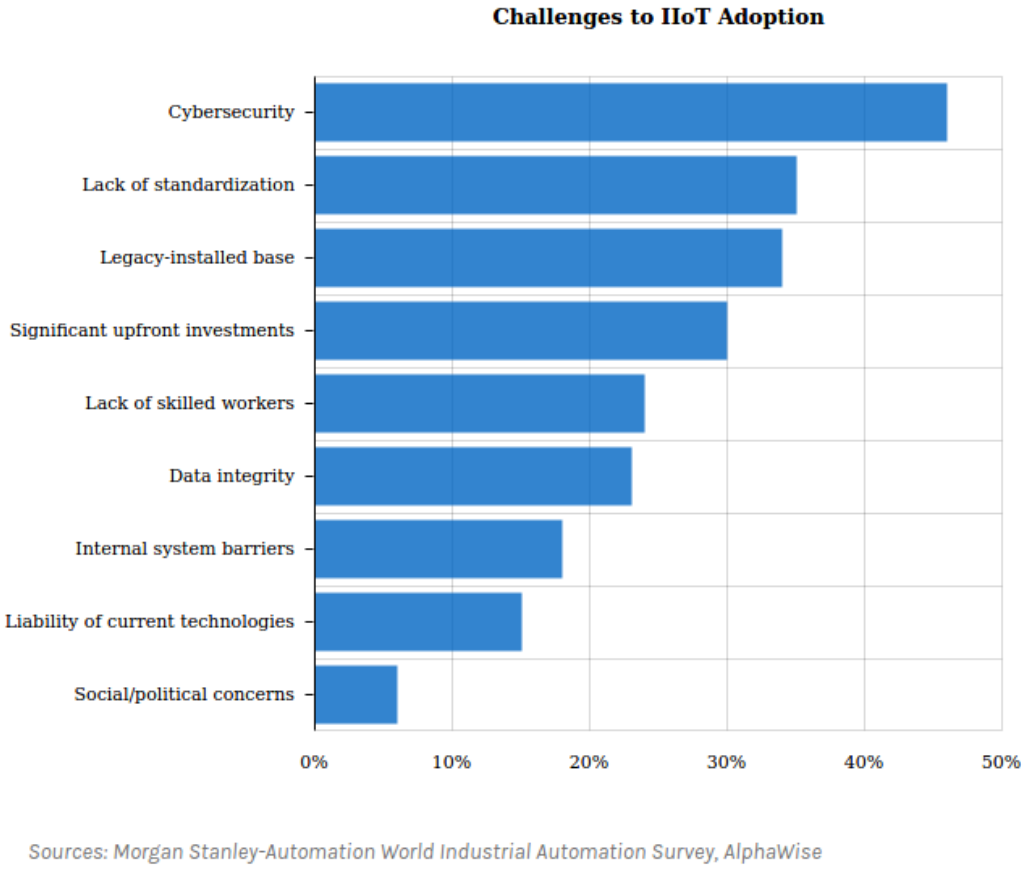


FIGURE 1.3 – Sondage sur les challenges de l'IoT de 2015

réseau énergétique de l'Ukraine. Les deux attaques ne semblent en revanche pas directement reliées d'après les experts. La première attaque utilise une association de deux maliciels dénommés *BlackEnergy* et *Killdisk*. *BlackEnergy*, logiciel malveillant détecté la première fois en 2007, est un "cheval de Troie" dans le vocabulaire des attaques informatiques. Un cheval de Troie est un logiciel en apparence légitime qui contient en fait une fonctionnalité cachée et en général malveillante. Ainsi ce programme malveillant a notamment été utilisé dans le cadre d'espionnage industriel, d'attaques DDoS, ou de corruptions de systèmes contrôle/commande. L'infection initiale se fait souvent à travers des liens ou pièces-jointes malveillantes. Ce maliciel a été adapté pour l'attaque puis utilisé comme point d'entrée aux systèmes ciblés. Une fois les cibles infectées, le maliciel utilise un plugin dénommé *KillDisk* (qui fait partie de la modification de *BlackEnergy* évoqué précédemment). Ce plugin a pour objectif de remplacer les données existantes par des données aléatoires ou tout simplement d'effacer les données, ce qui peut rendre les appareils impossibles à démarrer et à utiliser. Cette attaque a de cette manière interrompu les systèmes informatiques des entreprises responsables du réseau électrique en Ukraine, ce qui a provoqué des pannes d'électricité majeures sur plusieurs régions.

1.2.2.2 Industroyer (2016)

Industroyer est le nom d'une autre attaque informatique ayant eu lieu en Ukraine en 2016. Cette attaque visait directement les disjoncteurs, switchs et autres actionneurs du réseau électrique contrôlables à distance. Les appareils infectés communiquaient avec un serveur malveillant afin de faire remonter des informations ou recevoir des ordres. Une des particularités de cette attaque est que les messages échangés afin de perturber le réseau électrique s'inscrivaient totalement dans les charges légitimes possibles, rendant la détection des perturbations de l'attaque plus difficile. Après analyse de l'attaque, il a été clair que l'attaque ciblait particulièrement le réseau électrique ukrainien et que celui-ci a été étudié préalablement par les attaquants. L'attaque, déclenchée en décembre 2016, a provoqué une panne d'électricité sans précédent du réseau électrique ukrainien.

1.2.2.3 Triton (2017)

L'attaque dénommée *TRITON* et parfois aussi connue sous les noms de *TRISIS* ou de *Hatman* fut dévoilée en décembre 2017 dans un complexe pétrochimique d'Arabie Saoudite. Cette attaque ciblait plus précisément un équipement du système de sécurité-innocuité (la non-occurrence de conséquences catastrophiques pour l'environnement). L'appareil ciblé, qui est un type particulier de*PLC* (*Programmable Logic Controller*) utilisé dans les systèmes industriels, permet par exemple la gestion de brûleurs et de chaudières ou le contrôle de turbomachines. Les attaquants ont donc étudié cet objet en particulier avant de lancer l'attaque. Ils ont ainsi pénétré le réseau informatique du complexe industriel, qui est également relié au réseau opérationnel. Une fois l'appareil infecté, l'attaquant a eu la possibilité de lire, d'écrire et d'exécuter du code directement dans la mémoire. Heureusement, une vérification redondante a déclenché une alarme et permis de détecter l'attaque avant qu'il y ait de réels dommages.

Cette attaque, avec *Stuxnet*, fait partie des attaques qui ciblent directement des systèmes de sécurité-innocuité (*safety*) du monde industriel, et qui peuvent donc créer des dégâts physiques potentiellement catastrophiques.

1.2.2.4 NotPetya (2017)

NotPetya tire son nom de sa ressemblance avec un logiciel de rançon, ou rançongiciel (*ransomware*) connu sous le nom de Petya. Cependant, après analyse, il semblerait que les deux logiciels ne soient pas reliés. Cette attaque, bien qu'affichant un message de rançon où il est proposé de payer pour récupérer le contrôle et les données de son ordinateur, a un but totalement destructeur. NotPetya couple l'utilisation de deux outils : *EternalBlue* et *Mimikatz*. *EternalBLue* est un outil créé par la NSA (National Security Agency) des États-unis qui avait fuité plus tôt la même année. Cet outil exploitait une vulnérabilité d'un protocole de *Windows* vulnérable afin d'exécuter du code à distance sur la machine. En effet, lorsque la mise à jour contre cette attaque était disponible, la validation et le déploiement dans les entreprises ou industries furent plus lents, ce qui a rendu l'attaque possible. Mimikatz

quant à elle, était une preuve de concept développée par un chercheur Français du nom de Benjamin Delpy, permettant de récupérer un mot de passe d'un utilisateur sur le système d'exploitation Windows. Cette attaque fut d'une virulence sans précédent, touchant de nombreuses entreprises et industries à travers le monde. Les dommages sont estimés à plus de 10 milliards de dollars.

1.2.2.5 LockerGoga (2019)

Norsk Hydro est une des plus grandes entreprises de production d'aluminium au monde. Cette entreprise a été la cible en mars 2019 d'une attaque de type rançongiciel d'origine inconnue. Le rançongiciel utilisé, nommée LockerGoga, n'avait été que rarement rencontré, mais rapporté par l'entreprise française Altran plus tôt en Février. Cette attaque a interrompu une partie de la production. L'incident a un coût estimé à au moins 40 millions de dollars, à cause de l'interruption de la production, des arrêts de certaines ventes, et des moyens mis en œuvre pour remettre en fonctionnement le système informatique. L'attaque se serait propagée via l'*active directory*, un système de gestion des identifiants centralisé permettant d'authentifier et de surveiller les accès réseaux. Cette méthode de propagation explique pourquoi seule l'entreprise a été touchée par cette attaque. L'attaque semble donc venir du réseau interne mais l'intrusion initiale dans le réseau de l'entreprise n'est pas encore claire. L'entreprise a pu remédier à ce problème en rechargeant localement et manuellement les sauvegardes des systèmes.

Ces différentes attaques nous montrent bien que les systèmes industriels, qui ont parfois des problématiques et environnements particuliers dans le monde des objets connectés, sont fortement concernés par la sécurité de leurs objets ou appareils connectés. Dans la suite, nous allons préciser les différents scénarios et hypothèses auxquels nous nous intéressons, pour finir par présenter le contexte de nos travaux.

1.2.3 Contexte applicatif et hypothèses

Après avoir précisé le contexte industriel, nous allons spécifier les cas d'usage que nous allons considérer puis les hypothèses, objectifs et modèles de menace qui en découlent.

1.2.3.1 Domaine de l'énergie

Nos travaux concernent plus particulièrement le domaine de l'énergie. Nous nous mettons donc à la place d'un fournisseur d'énergie qui est intéressé par l'IdO pour certains cas d'usage. Nous allons ainsi dans un premier temps expliciter différents cas d'usage dans lesquels nous pouvons nous inscrire dans le cadre de l'Internet industriel des objets.

Le premier cas d'usage est le bâtiment intelligent. L'objet auquel on s'intéresse dans cette thèse communique avec des capteurs et des actionneurs à l'aide de communications bien définies. Cet objet n'essaie pas de s'adapter à tout l'environnement du bâtiment, mais uniquement à la partie spécifique à l'énergie, son utilisation est

peu impactée par son environnement. Un tel objet a ainsi pour objectif de gérer l'autoconsommation du bâtiment, c'est-à-dire de choisir quand et où stocker l'énergie produite par le bâtiment et quand la dépenser. Les objets auxquels nous nous intéressons sont plus en marges de l'environnement du bâtiment connecté dans le sens où leurs interactions et les objets avec lesquels ils peuvent communiquer sont définis à l'avance.

Le deuxième cas est un complexe de génération d'électricité, tel qu'une ferme de panneaux photovoltaïques ou d'éoliennes. Dans ce genre de systèmes, la connexion des objets permet une surveillance de l'état des équipements (panneaux ou éoliennes) ou permet d'améliorer leur rendement grâce à l'orientation des panneaux en fonction du soleil ou des éoliennes en fonction du vent. Chacun de ces dispositifs doit donc être muni d'un objet connecté permettant la remontée d'information et le contrôle automatique de l'orientation grâce à des capteurs possiblement désactivables à distance. Pour cela, il est possible d'utiliser une connexion filaire ou sans-fil, mais l'éloignement géographique des différents appareils oriente vers une solution sans-fil. La centralisation des informations et leur envoi aux serveurs du fournisseur de services se fait via une passerelle, qui permet de traduire et filtrer les informations des différents objets pour les serveurs. En effet, le protocole de communication permettant de communiquer entre le serveur et les objets n'est pas forcément le même que celui utilisé entre le serveur et la passerelle. Nous avons ainsi ici un système permettant une surveillance de l'état des fermes et un contrôle à distance relativement simple.

Enfin pour le dernier cas, nous considérons un poste de gestion du réseau électrique. Les objets et objectifs sont similaires au scénario précédent, sans le côté optimisation de la génération d'énergie, mais avec une surveillance plus importante. De plus, les communications se feront plutôt en filaire, car ces postes peuvent avoir une criticité très forte, et nécessitent donc un contrôle strict des différentes communications et des commandes envoyées. Cela nous amène donc à nous questionner sur l'impact des attaques que l'on pourrait envisager ou sur le type d'attaquant que l'on pourrait rencontrer dans ce contexte.

1.2.3.2 Réseau électrique intelligent

Ces différents cas d'usage sont tous relatifs au réseau électrique intelligent, ou *smart grid* en anglais. Avec le développement des énergies renouvelables, le nombre de nœuds du réseau pouvant fournir de l'énergie augmente considérablement. Cette augmentation est également due au fait qu'il est possible pour les particuliers de posséder eux-mêmes des équipements générant de l'électricité pour la revendre. Cela complexifie grandement le réseau électrique ainsi que sa gestion pour deux raisons. La première est que les énergies renouvelables sont très dépendantes d'évènements non maîtrisés comme la météo. La production est ainsi beaucoup plus complexe à prévoir, et équilibrer l'offre et la demande, la production et la consommation devient bien plus complexe. L'avènement des objets connectés et des environnements intelligents contribue ainsi au développement de ces réseaux électriques intelligents, dans lesquels les particuliers et producteurs d'énergie peuvent communiquer de

manière automatisée afin de remplir cet objectif d'équilibrage entre l'offre et la demande en énergie électrique.

Ainsi, grâce à ces smart grids, l'objectif est d'obtenir des informations sur la production et la consommation pour organiser le réseau. Par exemple dans le cas où l'on considère uniquement l'énergie solaire, une zone sans soleil pourrait être couverte en électricité par une zone adjacente qui en bénéficierait. Dans le cas de la consommation, certains automatismes peuvent être créés pour que la consommation d'électricité s'adapte aux changements de la production. Pour donner un exemple concret dans le cas des domiciles connectés (smart homes), on pourrait imaginer que ceux-ci puissent attendre un message du réseau électrique avant de lancer le fonctionnement d'une machine à laver par exemple, ou recharger des batteries personnelles ou professionnelles (comme pour les voitures électriques). En contrepartie, l'électricité pourrait être moins chère car le client s'adapte à la situation de surproduction et en fonction du coût et de la période de la journée. Il est également possible de penser au cas contraire où le client pourrait être récompensé lorsqu'il s'adapte à une situation de sous-production.

La gestion du réseau électrique se fait donc au plus proche des clients finaux, avec une vision distribuée du réseau. Cela engendre la nécessité d'avoir un niveau de sécurité élevé pour tous les nœuds permettant la gestion du réseau. Cependant cette idée est encore en cours de développement. En effet, le déploiement de compteurs communicants, qui se déroule depuis quelques années en France comme dans le reste de l'Europe ou encore aux États-Unis, est une étape nécessaire mais non suffisante à la création d'un réseau électrique intelligent. Notre étude ne se concentre pas sur le cœur des appareils pouvant constituer une smart grid, mais se situe au plus proche des objets déployés en masse qui pourraient se retrouver impliqués dans un tel dispositif.

1.2.3.3 Hypothèses et objectifs

Dans cette partie, nous allons expliciter les différentes hypothèses et objectifs que nous tirons du contexte, des cas d'usage et des attaques décrits dans ce chapitre.

Hypothèse 1 : Nous nous inscrivons dans le contexte d'un déploiement massif d'un grand nombre d'objets répartis sur une large zone géographique. Afin de réduire les coûts de production et faciliter le déploiement, tous les objets sont composés des mêmes éléments matériels et logiciels. Les objets s'inscrivent dans un système global connecté et intelligent, mais notre étude ne se référera qu'aux objets eux-mêmes. Les objets considérés sont bien connus et les différentes interactions se font dans un cadre bien délimité. Nous restons donc ici dans un contexte plutôt maîtrisé relativement à d'autres environnements d'objets connectés ayant des objectifs purement liés au confort par exemple.

Hypothèse 2 : S'il y a plusieurs versions du logiciel déployées en même temps, nous considérons que le fournisseur de service a la traçabilité nécessaire afin de connaître les versions des objets. De plus, les différentes versions sont toutes exécutées par un grand nombre d'objets.

Hypothèse 3 : Les objets ont des capacités limitées en terme de puissance de

traitement et d'énergie. Ils exécutent des tâches relativement simples et répétitives, avec un système d'exploitation léger, afin de satisfaire l'optimisation des ressources de l'objet et la réduction du son coût global.

Objectif 1 : Les solutions de sécurité envisagées ne doivent pas induire de modifications majeures au niveau du matériel, afin de réduire les coûts de production et faciliter le déploiement

Objectif 2 : Les solutions de sécurité doivent donc utiliser peu de ressources supplémentaires de l'objet. Les communications et les ressources nécessaires au déploiement du dispositif de sécurité doivent également être dimensionnées pour éviter des problématiques d'engorgement des communications au niveau du réseau, en conformité avec l'hypothèse 2.

Objectif 3 : Les solutions de sécurité que nous étudions doivent être adaptées à un environnement industriel, être relativement simples à déployer et à maintenir. La solution doit s'adapter aux mises à jours des objets sans interventions.

Nous utiliserons ces différentes hypothèses et objectifs dans la suite de ce document. Ils nous permettront d'expliquer notre approche.

1.3 Conclusion

Dans ce chapitre, nous avons tout d'abord présenté de manière générale l'Internet des objets, en soulignant certaines de ses particularités et problématiques. Nous avons ensuite décrit le domaine qui nous intéresse dans la suite du document : la sécurité informatique, dont nous avons souligné l'importance dans le contexte général de l'Internet des objets. Nous avons dans un second temps présenté un contexte plus spécifique de l'Internet des objets que nous abordons dans nos travaux, celui des systèmes industriels. Nous avons encore une fois souligné l'importance de la sécurité informatique dans ce contexte particulier et présenté un certain nombre d'attaques connues sur les systèmes industriels. Finalement, nous avons terminé par introduire plus concrètement les cas d'usage que nous envisageons dans le cadre de nos travaux.

Le chapitre suivant présente une analyse de risques que nous avons menée dans la première partie de cette thèse, au travers d'une étude EBIOS [16] concernant un objet connecté typique qui est aujourd'hui à l'étude pour être déployé dans le contexte de l'énergie. Cette étude nous a été utile pour mieux comprendre et mettre en évidence les risques encourus lors du déploiement d'objets connectés dans notre contexte spécifique, et surtout afin de mieux définir les modèles de menaces que nous devons considérer.

16. https://www.ssi.gouv.fr/guide/ebios-2010-expression-des-besoins-et-identification-des-objectifs-de-securite/

Chapitre 2

De l'analyse de risque au modèle de menace

Sommaire

2.1 Introduction 25
2.1.1 Méthode EBIOS 26
2.1.2 Cas d'étude 28
2.2 Analyse de risques 30
2.2.1 Étude du contexte 30
2.2.2 Étude des évènements redoutés 33
2.2.3 Étude des scénarios de menaces 34
2.2.4 Étude des risques 36
2.3 Modèle de menace 37
2.4 Conclusion 40

Dans cette partie nous utilisons la méthode EBIOS afin de mener une analyse de risque sur un objet typique d'un environnement connecté qui peut s'inscrire dans les scénarios sur lesquels nous travaillons dans ce document. Pour cela nous allons tout d'abord introduire la méthode d'analyse et le type d'objet considéré, avant de mener l'analyse de risque. Cela nous permet ainsi d'avoir une idée et un exemple des enjeux et risques de sécurité dans les environnements intelligents étudiés dans ce document. Une analyse de risque ainsi qu'un test de pénétration ont été effectués sur un prototype d'objet du domaine de l'énergie. Nous tenons à souligner que, pour des raisons de confidentialité, nous ne pouvons donner précisément le modèle de l'objet que nous étudions dans l'analyse de risques. Cependant, les fonctionnalités principales et les caractéristiques que nous présentons correspondent bien aux capacités de l'objet considéré.

2.1 Introduction

Afin de faire une analyse de risque, nous avons choisi d'utiliser la méthode développée par l'Agence Nationale de la Sécurité des Systèmes d'Informations (ANSSI) en France appelée EBIOS (Expression des besoins et Identification des Objectifs de sécurité). Cette méthode permet ainsi d'apprécier et de traiter les risques relatifs à la sécurité des systèmes d'informations (SSI).

La méthode EBIOS peut être employée autant pour étudier des systèmes à concevoir que des systèmes existants. Il est également possible de ne déployer qu'une partie de la démarche pour réaliser par exemple une analyse de vulnérabilités (études des menaces uniquement), ou un recueil d'éléments stratégiques (étude du contexte, expression des besoins non détaillée, étude de menace non détaillée).

Les bases de connaissances d'EBIOS présentent et décrivent des types d'entités, des méthodes d'attaques, des vulnérabilités, des objectifs de sécurité et des exigences de sécurité. Elles sont directement applicables à la plupart des secteurs, et peuvent être facilement adaptées à la situation. Pour plus de clarté, nous allons commencer par définir les termes principaux utilisés dans ce type d'analyse.

2.1.1 Méthode EBIOS

Les définitions ci-dessous sont fournies par l'ANSSI [1], elles seront employées en ce sens dans le reste de ce document.

— **Bien :** Toute ressource qui a de la valeur pour l'organisme/entreprise et qui est nécessaire à la réalisation de ses objectifs. On distingue notamment les biens essentiels et les biens supports.

— **Bien essentiel :** Information ou processus jugé comme important pour l'organisme/entreprise. On appréciera ses besoins de sécurité mais pas ses vulnérabilités.

— **Bien support :** Bien sur lequel reposent des biens essentiels. On distingue notamment les systèmes informatiques, les organisations et les locaux. On appréciera ses vulnérabilités mais pas ses besoins de sécurité.

— **Besoin de sécurité :** Définition précise et non ambiguë du niveau d'exigences opérationnelles relatives à un bien essentiel pour un critère de sécurité donné (disponibilité, confidentialité, intégrité, ...).

— **Critère de sécurité :** Caractéristique d'un bien essentiel permettant d'apprécier ses différents besoins de sécurité.

— **Évènement redouté :** Scénario générique représentant une situation crainte par l'organisme. Il s'exprime par la combinaison des sources de menaces susceptibles d'en être à l'origine, d'un bien essentiel, d'un critère de sécurité du besoin de sécurité concerné et des impacts potentiels.

— **Gravité :** Estimation de la hauteur des effets d'un évènement redouté ou d'un risque. Elle représente ses conséquences.

— **Impact :** Conséquence directe ou indirecte de l'insatisfaction des besoins de sécurité sur l'organisme et/ou sur son environnement.

— **Menace :** Moyen type utilisé par une source de menace.

1. `https://www.ssi.gouv.fr/uploads/2011/10/EBIOS-1-GuideMethodologique-2010-01-25.pdf`

— **Mesure de sécurité :** Moyen de traiter un risque de sécurité de l'information. La nature et le niveau de détail de la description d'une mesure de sécurité peuvent être très variables.

— **Objectif de sécurité :** Expression de la décision de traiter un risque selon des modalités prescrites.

— **Scénario de menace :** Scénario, avec un niveau donné, décrivant des modes opératoires. il combine les sources de menaces susceptibles d'en être à l'origine, un bien support, un critère de sécurité, des menaces et des vulnérabilités exploitables pour qu'elles se réalisent. Son niveau correspond à l'estimation de sa vraisemblance.

— **Source de menace :** Chose ou personne à l'origine de menaces. Elle peut être caractérisée par son type (humain ou environnemental), par sa cause (accidentelle ou délibérée et selon le cas par ses ressources disponibles, son expertise, sa motivation, etc.

— **Vraisemblance :** Estimation de la possibilité qu'un scénario de menace ou un risque, se produise. Elle représente sa force d'occurrence.

La méthode EBIOS est divisée en 5 modules différents, représentés dans le schéma 2.1

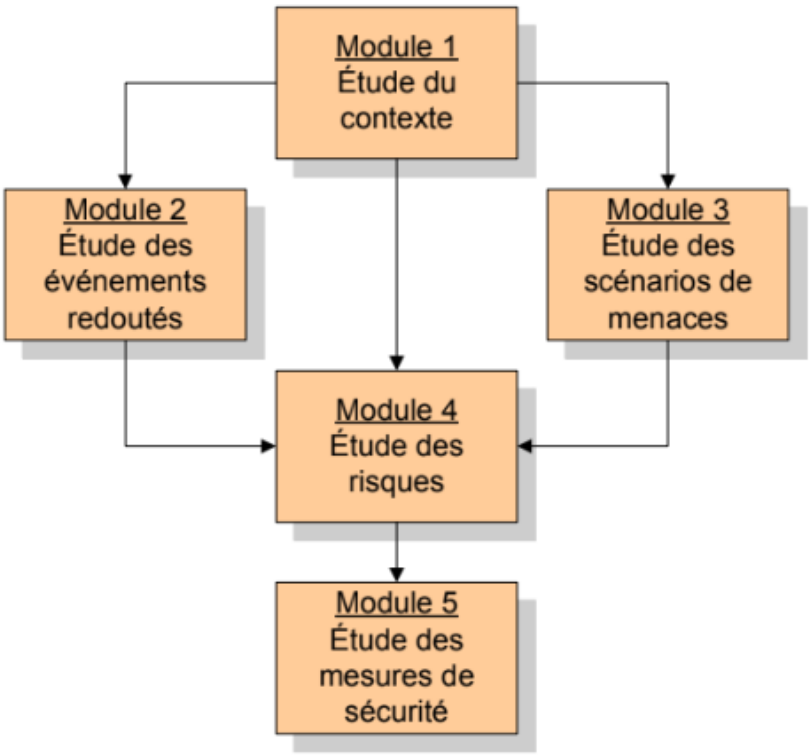


FIGURE 2.1 – La représentation des différents modules EBIOS

L'**étude de contexte** permet de définir un cadre et des hypothèses de départ pour comprendre le sujet et les enjeux. On définit notamment les biens essentiels, les biens supports, les liens entre les deux et les métriques utilisées.

L'**étude des évènements redoutés** est une étude plus fonctionnelle que technique. Un évènement redouté porte sur un critère de sécurité d'un bien essentiel. Nous précisons pour chacun de ces évènements le besoin de sécurité, la source de menace, ainsi que l'impact et la gravité en cas de manquement au besoin de sécurité.

L'**étude des scénarios de menaces** est une étude plus technique que fonctionnelle. Un scénario de menace porte sur un critère de sécurité d'un bien support. Chaque scénario de menace est composé des sources de menaces, des critères de sécurité des biens supports concernés ainsi que de la vraisemblance du scénario.

L'**étude des risques** confronte les évènements redoutés aux scénarios de menaces. Pour cela, nous utilisons les liens effectués entre les biens supports et les biens essentiels établis dans le premier module. Nous créerons alors une matrice des risques, qui croise **vraisemblance** et **gravité** de chacun des évènements pour nous permettre d'évaluer chaque risque

L'**étude des mesures de sécurité** concerne le traitement des risques. Cette étude formalise les mesures de sécurité à mettre en place afin de réduire les risques de l'étude précédente. Elle explique également comment mettre en oeuvre ces mesures de sécurité et en quoi celles-ci réduisent la vraisemblance des risques afin de les rendre acceptables. Finalement elle explique les risques résiduels ainsi que la position par rapport à l'analyse de risque les concernant.

Il est possible après la dernière étape de reprendre l'analyse depuis l'étude des scénarios de menaces et de créer ainsi une boucle. Il n'est normalement pas nécessaire de refaire l'étude des évènements redoutés car les mesures de sécurité n'influent en théorie que sur le critère de vraisemblance des scénarios. Dans le cas où le système subis un changement plus impactant pour le système étudié, il faut refaire une analyse complète.

2.1.2 Cas d'étude

Dans cette partie, nous allons décrire l'objet qui sera le sujet de notre analyse de risques. L'objet étudié est destiné à être utilisé dans le cadre d'un bâtiment intelligent. Il sera connecté à différents capteurs et actionneurs lui permettant d'interagir avec l'environnement. L'objet utilisera les informations à disposition via les capteurs, mais aussi certaines préférences et configurations entrées via une interface homme-machine, qui correspond à une interface web simple sauvegardée sur l'objet.

L'objet peut se connecter à un réseau Wi-fi ou en filaire via un port RJ45, ce qui permet par la suite de le configurer et d'avoir une liaison avec des serveurs distants.

Nous faisons l'hypothèse que le bâtiment dispose d'une passerelle permettant des échanges sur Internet. Un fournisseur d'accès à internet (FAI) permet d'assurer la connexion avec les serveurs distants. Les serveurs distants serviront pour la maintenance de l'objet ainsi que pour effectuer des interactions à distance (en dehors du réseau local), via un site web mis à disposition. Les objets essaieront de se connecter régulièrement aux serveurs en quête de mise à jour, mais également pour permettre la remontée d'informations utiles au bon fonctionnement du service. La figure 2.2 montrent les différentes communications et liens que peut avoir l'objet connecté étudié.

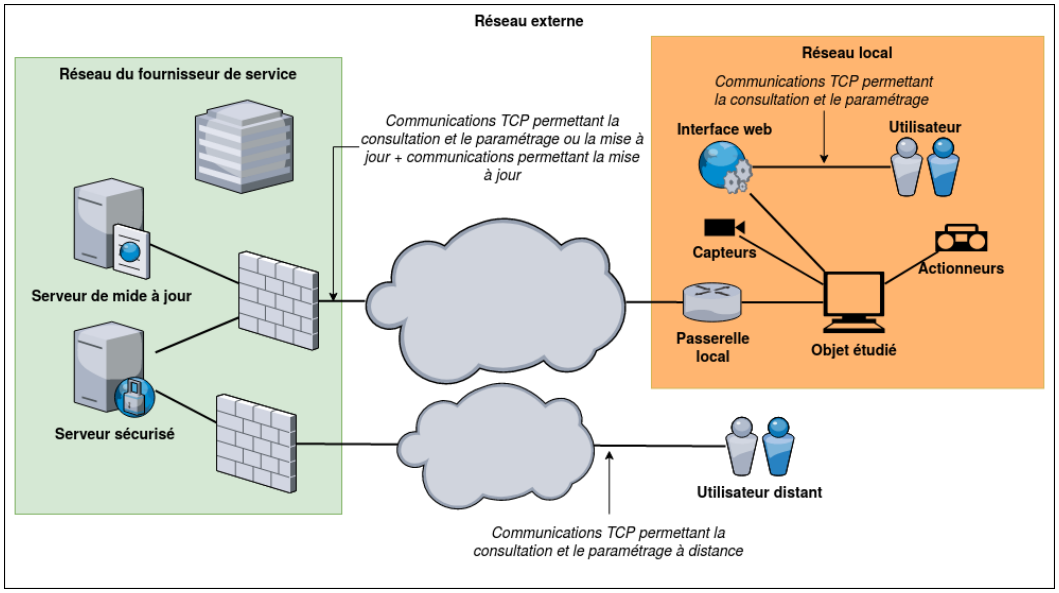


FIGURE 2.2 – Schéma résumant les différentes communications avec l'objet étudié

L'objet possède également un port USB permettant une mise à jour locale, ainsi que certains boutons de contrôles manuels : on/off, activation/désactivation du contrôle automatique des actionneurs, activation/désactivation des communications avec les serveurs, avec des indicateurs lumineux donnant l'état de ces fonctionnalités. L'objet contient également des ports lui permettant de se connecter en filaire aux capteurs ou actionneurs. En ouvrant le boîtier de l'objet, il est possible d'accéder à un port JTAG permettant de récupérer ou de changer la mémoire de l'objet.

Concernant la configuration via l'interface web, il est possible d'accéder à trois menus différents : un pour la configuration, un pour la visualisation d'informations relatives au service rendu, et un pour afficher d'éventuels erreurs et messages d'archives. Dans la partie configuration, il est possible de configurer un SSID et un mot de passe pour se connecter au réseau local et entrer des préférences d'utilisation. Il est toujours possible de se connecter sur l'interface de configuration de l'objet depuis le réseau local grâce à son adresse IP, sans mot de passe nécessaire.

Pour les communications avec les serveurs distants, seul l'identifiant de l'objet

et le mot de passe associé sont à entrer. Ce mot de passe est commun à tous les objets. Les serveurs appartiennent au fournisseur de service lié à l'objet. Le fournisseur de service est également le vendeur de l'objet connecté. Notons que toutes ces fonctionnalités sont assez classiques dans le cadre de la gestion de bâtiments intelligents.

Nous allons maintenant, grâce à ces différentes informations, mener une analyse de risque permettant d'éclaircir certaines problématiques que l'on peut rencontrer avec les objets connectés.

2.2 Analyse de risques

Maintenant que nous avons introduit la méthode EBIOS et le contexte dans lequel nous voulons l'utiliser, nous allons suivre la méthodologie afin de mener une analyse de risques. Nous aborderons ainsi les 4 premiers modules décrits précédemment. Le dernier module sur l'étude des mesures de sécurité ne sera pas abordé dans le cadre de cette analyse de risque. Une discussion plus générale sur les solutions de sécurité fera l'objet des chapitres suivants.

2.2.1 Étude du contexte

Nous commençons l'analyse par l'étude de contexte, le premier des modules présentés précédemment. Pour cela nous présentons tout d'abord les hypothèses que nous allons poser, avant d'identifier des sources de menace et de définir des mesures de sécurité existantes. Ces dernières seront inclues plus tard dans le calcul de vraisemblance ou d'impact des évènements redoutés. Nous rappelons également que le contexte est inspiré d'objets et d'analyses existants.

Hypothèses :

- Nous effectuons une analyse de risque uniquement sur l'objet, et ne considérons pas directement les objets ou serveurs avec lesquels l'objet interagit.
- Nous considérons les serveurs distants comme sécurisés et par conséquent ne participant pas à une attaque.
- Nous prenons en compte uniquement la sécurité vis-à-vis des malveillances et ne traitons pas de problématiques liées à des fautes accidentelles.
- Nous distinguons deux environnements distincts, l'environnement local permettant un accès direct à l'objet et un environnement externe sans accès direct.
- Nous considérons comme difficile pour une personne malveillante d'avoir un accès physique à l'objet.

Sources de menaces :

- Les personnes malveillantes ayant un accès au réseau interne.

- Les personnes malveillantes ayant un accès physique à l'objet.
- Les personnes extérieures malveillantes.
- Les objets connectés malveillants (de conception ou infectés) sur le réseau local.
- Le fournisseur d'accès internet.

Mesures de sécurité existantes :

- Authentification auprès du serveur : utilisation d'un couple identifiant/mot de passe et chiffrement asymétrique de type RSA. Taille de clés très faible.
- Vérification des champs de configuration (valeur, lettres, ...).
- Fichiers de mise à jour couplés avec un CRC.
- Nous considérons ici qu'il est difficile pour une personne malveillante d'avoir un accès physique à l'objet du fait de la vigilance et de la sécurité physique.

Dans la suite de cette analyse, nous définissons les différentes métriques liées à la sécurité de la méthodologie EBIOS. Les différents critères pris en compte ici sont donc : disponibilité, intégrité et confidentialité. Les différents niveaux des besoins des critères cités précédemment sont décrits dans les tableaux en annexe A.3, A.1, A.4, A.2, A.5.

Nous allons maintenant identifier les différents biens liés au système et sur lesquels l'analyse portera. La méthode EBIOS distingue les **biens essentiels** et les **biens supports**. Les **biens essentiels** étant les biens immatériels à protéger. Les **biens supports** sont quant à eux les biens "tangibles" qui manipulent ou gèrent les **biens essentiels**. Chaque **bien essentiel** est ainsi matérialisé par sa présence sur différents **biens supports** qui eux-même sont regroupés dans plusieurs catégories.

Besoins essentiels :

- Consultation et paramétrage de la configuration de l'objet depuis le serveur distant.
- Service rendu par l'objet, que nous confondons ici intentionnellement avec le logiciel.
- Contrôle par l'utilisateur des différents modes de l'objet (communications et automatismes).
- Mise à jour du logiciel.
- Données liées au service rendu, et données de configuration de l'objet.
- Données d'authentification de l'objet auprès des serveurs.
- Données d'identification sur le réseau local (Wi-Fi par exemple).

Besoins supports :

- Matériels : objet, JTAG.
- Organisations : fournisseur de service, fournisseur d'accès à internet.
- Réseaux : Réseau local, réseau extérieur.
- Personnels : Technicien.

Les tableaux 2.1 2.2 montrent les liens que l'on peut établir entre biens supports et biens essentiels, ces liens nous serviront pour définir les différents risques de l'étape d'analyse des risques :

TABLE 2.1 – Relation entre biens supports et biens essentiels : fonctionnalités

		Biens essentiels : fonctionnalités			
		Consultation et paramétrage	Service rendu	Contrôle par l'utilisateur	Mise à jour du logiciel
Biens supports	Objet	X	X	X	X
	Fournisseur de service	X			X
	FAI	X			X
	Réseau local	X		X	X
	Réseau externe	X			X
	JTAG	X	X	X	X
	Technicien				

TABLE 2.2 – Relation entre biens supports et biens essentiels : données

		Biens essentiels : données		
		Données personnelles du client	Données d'authentification Wi-Fi	Données d'authentification de l'objet auprès du serveur
Biens supports	Objet	X	X	X
	Fournisseur de service			X
	FAI			X
	Réseau local	X	X	X
	Réseau externe			X
	JTAG	X	X	X
	Technicien		X	X

2.2.2 Étude des évènements redoutés

L'étude des évènements redoutés consiste à déterminer, pour chaque critère de sécurité de chaque bien essentiel, la gravité des impacts, lorsqu'il y a manquement au besoin de sécurité. Différents impacts ont été retenus suite à des discussions avec des experts : financier, image de marque, perte de données, juridique, perte de propriété des données. Le besoin de sécurité correspond au niveau attendu de sécurité (besoins définis précédemment). Ainsi, pour chaque bien essentiel et pour chaque critère de sécurité, nous allons associer :

- Le besoin de sécurité attendu.
- Les sources de menaces possibles.
- Les impacts possibles en cas de manquement au besoin de sécurité.
- La gravité de l'impact.

Par soucis de lisibilité du document, nous décrivons uniquement l'évènement redouté lié au premier besoin essentiel. L'intégralité de ces évènements sont décrits en annexe. En effet, la description de ces évènements n'est pas essentielle pour la compréhension, et les différents niveaux de gravité sont résumés dans le tableau 2.3.

Bien essentiel 01 : Consultation et paramétrage de l'objet depuis les serveurs

Critère de confidentialité

Cet évènement redouté correspond à la divulgation des ordres envoyés du serveur à l'objet.

- Besoin de sécurité : Privé, ces informations peuvent révéler un certain comportement de l'utilisateur et doivent être chiffrées (niveau 2).
- Sources de menace : FAI, personne extérieure ou intérieure malveillante ou objet connecté malveillant procédant à une attaque réseau.
- Impacts : Image de marque et possiblement révélation du comportement de l'utilisateur.
- Gravité : Importante.

Critère d'intégrité

Cet évènement redouté correspond au changement des ordres du serveur.

- Besoin de sécurité : Détectable, les ordres ne doivent pas pouvoir être changés par une tiers personnes mais n'ont pas de contraintes temporelles fortes.
- Sources de menace : FAI, personne extérieure ou intérieure malveillante procédant à une attaque réseau, objet connecté malveillant.
- Impacts : Image de marque.
- Gravité : Limitée.

Critère de disponibilité

Cet évènement redouté correspond à l'impossibilité pour le serveur d'envoyer des données à l'objet.

- Besoin de sécurité : Doit être disponible dans les 48h, les ordres du serveur sont importants mais pas primordiaux au bon fonctionnement de l'objet.
- Sources de menace : FAI, personne extérieure ou intérieure malveillante procédant à une attaque réseau, objet connecté malveillant.
- Impacts : Image de marque
- Gravité : Limitée

Le tableau 2.3 résume la gravité de l'impact en fonction du manquement au critère de sécurité correspondant, pour tous les besoins essentiels identifiés :

Table 2.3 – Gravité de l'impact en cas de manquement au critère de sécurité

		Critères de sécurité		
		Disponibilité	Intégrité	Confidentialité
Biens essentiels	01. Consultation et paramétrage de l'objet depuis les serveurs	Limitée	Limitée	Importante
	02. Contrôle des modes par l'utilisateur	Limitée	Importante	Importante
	03. Données du client	Limitée	Importante	Critique
	04. Données d'authentification auprès des serveurs	Importante	Importante	Importante
	05. Données d'authentification au réseau local	Critique	Limitée	Importante
	06. Logiciel	Importante	Critique	Importante
	07. Mise à jour du logiciel	Critique	Critique	Importante

2.2.3 Étude des scénarios de menaces

L'étude des scénarios de menaces permet de mettre en évidence les attaques potentielles sur le système étudié. Les scénarios de menaces portent sur les biens supports en fonction du critère de sécurité correspondant, en y ajoutant les source(s) de menace et les vulnérabilité(s) exploité(s). Tous ces critères couplés aux discussions avec un groupe d'experts nous ont permis de définir la vraisemblance de chaque scénario.

Pour résumer, chaque scénario de menace est composé de :

- Bien support.

- Critère(s) de sécurité.
- Source(s) de menace(s).
- Vulnérabilité(s) exploité(s).
- Vraisemblance (le niveau du scénario de menace).

Nous n'allons pas ici détailler et expliquer tous les niveaux de vraisemblance par soucis de simplicité et de lisibilité. La vraisemblance prend en compte la facilité d'exploitation de la vulnérabilité, l'exposition aux menaces, et les ressources nécessaires à la menace.

Nous allons expliciter maintenant différents scénarios de menaces que l'on peut envisager ici et qui sont pertinents à citer.

Scénario 01 : Une personne malveillante écoute les communications entre le serveur et l'objet à des fins d'atteinte à la confidentialité en exploitant le fait que ces communications ne soient pas chiffrées. Ce scénario concerne donc le réseau local, le réseau externe ou le FAI. Vraisemblance : Forte.

Scénario 02 : Une personne malveillante brouille le Wi-Fi du client et émet un signal plus fort avec le même SSID afin que l'objet se connecte sur le réseau malveillant. L'attaquant peut ainsi atteindre l'interface web de configuration. Ce scénario concerne le réseau local. Vraisemblance : Significative.

Scénario 03 : Une personne malveillante retrouve le mot de passe de l'objet, lui permettant ainsi de s'authentifier auprès du serveur en écoutant les échanges à des fins d'atteinte à la confidentialité en exploitant le fait que les clés de chiffrement sont de petites tailles. Ce scénario concerne le réseau local, le réseau externe ou le FAI. Vraisemblance : Significative.

Scénario 04 : Une personne malveillante ou un objet connecté malveillant se fait passer pour le serveur auprès de l'objet afin d'envoyer des ordres et porter atteinte à l'intégrité de l'objet en exploitant le fait que l'objet n'authentifie pas le serveur. Ce scénario concerne donc le réseau local, le réseau externe ou le FAI. Vraisemblance : Significative.

Scénario 05 : Une personne malveillante ou un objet connecté malveillant accède à l'interface de configuration afin de recueillir des informations confidentielles en exploitant le fait que l'objet ne demande aucune authentification afin d'accéder à son interface de configuration. Ce scénario concerne donc le réseau local, le réseau externe et le FAI. Vraisemblance : Forte, les deux scénarios précédents pouvant mener à celui-ci.

Scénario 06 : Une personne malveillante ou un objet connecté malveillant simule le serveur permettant la mise à jour, et ainsi prendre le contrôle de l'objet. Ce scénario concerne donc le réseau local, le réseau externe ou le FAI. Vraisemblance : Importante.

Scénario 07 : Une personne malveillante se fait passer pour l'objet auprès du serveur afin d'envoyer de fausses informations au serveur, en exploitant les vulnérabilités liées au stockage des identifiants et mots de passe de l'objet. Ce scénario concerne le réseau local, l'objet ou le port JTAG. Vraisemblance : Significative, les motivations d'une telle attaque seraient faibles car les informations remontées n'influent pas sur le fonctionnement de l'objet, et on ne considère pas les attaques sur les serveurs.

Scénario 08 : Une personne malveillante interne ou un objet connecté malveillant change la configuration de l'objet (via l'interface web de configuration) afin d'arrêter son fonctionnement ou de faire dysfonctionner les différents appareils auxquels l'objet envoie des ordres en exploitant le fait que l'objet ne demande aucune authentification particulière lors de la configuration.

Scénario 09 : Une personne malveillante ou un objet connecté malveillant se fait passer pour le serveur NTP et falsifie l'heure de l'objet pour dérégler son fonctionnement et ainsi porter atteinte à l'intégrité de l'objet en exploitant le fait que celui-ci ne chiffre ni n'authentifie ses communications. Ce scénario concerne donc le réseau local, le réseau externe ou le FAI. Vraisemblance : Significative.

Scénario 10 : Une personne malveillante ou un objet connecté malveillant simule les serveurs distants afin de fausser les communications avec l'objet. Ce scénario concerne le réseau local, le réseau externe ou le FAI.

On peut remarquer qu'en regroupant les scénarios 07 et 10, il est possible d'effectuer une attaque de type Man-In-The-Middle sans que les serveurs ou les objets ne puissent la détecter.

Le tableau suivant montre le niveau de vraisemblance des scénarios par bien support et critère de sécurité, pour cela nous avons recherché les scénarios pouvant affecter les biens supports en fonction des critères de sécurité puis gardé les vraisemblances les plus grandes pour former le tableau.

2.2.4 Étude des risques

Dans les parties précédentes, nous avons analysé les évènements redoutés et les scénarios de menaces. Nous allons maintenant dans cette partie relier les deux afin de former des "risques". Pour identifier un risque, pour chaque évènement redouté, nous allons repérer les scénarios de menaces qui :

— Portent sur les biens supports liés au bien essentiel de l'évènement redouté.

TABLE 2.4 – Niveau de vraisemblance de mise à exécution des menaces

		Critères de sécurité		
		Disponibilité	Intégrité	Confidentialité
Biens supports	Objet	Significative	Forte	Forte
	Fournisseur de service	Significative	Significative	Significative
	FAI	Significative	Minime	Significative
	Réseau local	Significative	Minime	Significative
	Réseau externe	Minime	Maximale	Maximale
	JTAG	Minime	Significative	Minime
	Technicien	Minime	Minime	Significative

— Portent sur le même critère de sécurité.

— Possèdent les mêmes sources de menaces.

Chaque risque est donc composé de la combinaison d'un évènement redouté avec un scénario de menace. Cependant pour plus de simplicité, il est courant de rassembler tous les scénarios de menace pour un seul évènement redouté puis de prendre le critère de *vraisemblance* le plus élevé. Le tableau 2.5 montre la matrice des risques résultant de notre analyse. Les cases rouges correspondent aux cases où le risque n'est pas acceptable, les cases oranges acceptables sous certaines conditions, et les cases vertes acceptables.

On remarque que dans le cas de l'IdO, les vraisemblances sont souvent élevées, du fait de la grande surface d'attaque de ces objets. En effet, même le réseau local peut être malveillant, en plus du fait que les données transitent sur des réseaux publics (internet). De plus, les attaques sur les systèmes industriels décrits dans le chapitre précédent nous montrent que les attaquants sont organisés et qu'ils ont une bonne connaissance du système qu'ils attaquent.

Quant à la gravité, elle est dans notre cas élevée car nous avons considéré un objet ayant des fonctionnalités pouvant impacter le monde physique. Ainsi la compromission d'un tel objet peut avoir des impacts importants sur son environnement ainsi que sur l'image de marque du fournisseur de service lié à la vente de l'objet.

Dans cette partie, nous avons effectué une analyse de sécurité sur un objet de bâtiment intelligent, en s'inspirant d'objets connectés existants. Cela nous permet d'avoir un exemple et une vision d'ensemble plus concrète sur le fonctionnement des objets connectés et de la sécurité. Nous allons ainsi nous appuyer sur cette analyse, sur le contexte et les cas d'usage, afin de définir le modèle de menace que nous allons utiliser dans la suite de ce document.

2.3 Modèle de menace

La matrice de risque du tableau 2.5, résultant de la section précédente, met en évidence un certain nombre de risques dont certains sont associés à une gravité

TABLE 2.5 – Matrice des risques

		Vraisemblance			
		Minime	Significative	Forte	Maximale
Gravité	Négligeable				
	Limitée		Disponibilité de consultation et paramétrage depuis le serveur Disponibilité du contrôle des modes Disponibilité d'authentification local	Disponibilité des données clients	Confidentialité mise à jour du logiciel
	Importante		Confidentialité de consultation et paramétrage depuis le serveur Intégrité du contrôle des modes Intégrité authentification local Confidentialité du contrôle des modes	Confidentialité du logiciel Disponibilité du logiciel Disponibilité authentification de l'objet	Intégrité de consultation et paramétrage depuis le serveur Disponibilité de la mise à jour
	Critique		Confidentialité authentification local Intégrité du logiciel	Intégrité des données clients Intégrité authentification de l'objet Intégrité de la mise à jour	Confidentialité des données clients Confidentialité authentification de l'objet

critique et à une forte vraisemblance. Mais si la matrice de risques est bien sûr spécifique à l'objet étudié, il n'en reste pas moins qu'elle met en lumière un certain nombre de risques très importants et montre bien que la sécurité des objets connectés dans un contexte industriel doit être sérieusement considérée. Au vu des scénarios de menaces et de nos objectifs, nous allons écarter de notre étude les solutions portant sur le réseau lui-même, on en déduit ainsi l'hypothèse de menace suivante.

Hypothèse de menace 1 : Les serveurs ainsi que les communications entre objets et serveurs sont sécurisées. Les serveurs ne participent donc pas à une attaque. Nous supposons que des solutions classiques de sécurité sont déjà mises en place sur le réseau, permettant notamment de repérer les appareils ne répondant pas aux messages du serveur.

Ainsi en reprenant l'analyse de la section précédente, les scénarios de menaces reposant sur les communications entre les objets connectés et le serveur ont leur vraisemblance réduite grâce à cette hypothèse. Pour nos travaux et donc la suite de ce document, nous nous concentrerons sur les biens essentiels liés aux fonctionnalités de l'objet et à leur intégrité.

Dans le chapitre précédent, nous avons également présenté des attaques célèbres sur des objets connectés. On peut ainsi constater différents types d'attaques avec des objectifs différents. Il y a ainsi des attaquants dont le but est juste de prendre le contrôle des objets connectés pour pouvoir par la suite les utiliser à leur fin comme Mirai, et des attaques plus destructrices comme NotPetya et en général plus ciblées qui peuvent avoir de plus graves conséquences sur les environnements et réseaux des appareils infectés.

Hypothèse de menace 2 : Dans notre cas, nous allons considérer que l'objectif de l'attaquant est de prendre le contrôle des objets déployés afin de les utiliser à ses propres fins. Relativement à l'analyse de menaces, cela revient à considérer les attaques sur l'intégrité des fonctionnalités de l'objet. L'attaquant pourrait ainsi utiliser les objets afin d'effectuer des attaques sur d'autres systèmes tiers ou pour servir ses propres intérêts. Par exemple l'attaquant pourrait créer un botnet afin d'effectuer une attaque de type DDoS grâce à tous ses objets infectés, ou alors pour miner de la crypto-monnaie.

Hypothèse de menace 3 : Nous considérons également que l'attaquant essaie de cacher un minimum son activité malveillante sur les objets, afin d'éviter une détection trop évidente de la corruption. L'attaquant ne désactive donc pas le logiciel légitime sur les objets (action qui pourrait déjà être détectée d'après nos hypothèses précédentes de surveillance du réseau), mais installe un logiciel malveillant qui s'exécute en parallèle de l'application légitime.

Hypothèse de menace 4 : Finalement, nous considérons que l'attaquant n'a pas les capacités nécessaires afin de mener une attaque sur tous les objets exactement en même temps. Pour effectuer une telle attaque, l'attaquant devrait posséder non seulement une capacité de communication très élevée afin de mener son attaque, mais aussi avoir une très bonne connaissance du réseau des objets

déployés pour pouvoir tous les cibler.

2.4 Conclusion

Dans ce chapitre, grâce à une analyse de risques, nous avons approfondi notre compréhension des enjeux et des différentes possibilités d'attaques d'un objet connecté. Nous avons pu définir grâce à ces informations un modèle de menaces et des hypothèses correspondant à nos objectifs et à notre recherche. Dans le chapitre suivant, nous présentons une étude de certaines solutions de sécurité qui semblent pertinentes dans le contexte de nos travaux avant de nous intéresser à l'approche de détection d'intrusion que nous avons au final développée dans nos travaux.

Chapitre 3

Etat de l'art

Sommaire

3.1 Approches de protection 41
3.1.1 Fonction non clonable physiquement 42
3.1.2 Attestation à distance 44
3.1.3 Diversification Logicielle 47
3.1.4 Détection d'intrusion 50
3.1.5 Conclusion des approches relativement aux hypothèses 52
3.2 Taxonomie de la détection d'anomalie 53
3.2.1 Spécification 55
3.2.2 Détection statistique 56
3.2.3 Détection basée sur de l'apprentissage automatique 57
3.3 Conclusion 60

Dans le domaine des technologies de l'information, de nombreuses techniques et solutions de sécurité existent et ont été développées pour des réseaux privés ou d'entreprises, des routeurs, des ordinateurs, des serveurs, des communications, etc. Toutes ces solutions ne sont pas forcément adaptées en l'état à un environnement intelligent de l'Internet des objets, que ce soit à cause de l'architecture qui est nouvelle ou à cause de l'hétérogénéité des objets et des environnements. De plus les faibles ressources et coûts des objets ne permettent pas l'utilisation de solutions complexes. Nous allons donc présenter de manière non-exhaustive différentes solutions qui ont motivé le monde de la recherche ces dernières années.

3.1 Approches de protection

Dans cette première section, nous présentons différents mécanismes de sécurité que nous avons étudiés dans le cadre de nos travaux et qui peuvent représenter des candidats intéressants pour la problématique de sécurité que nous traitons. Nous avons retenu notamment les PUF, les mécanismes d'attestation à distance et de diversification. Le domaine de la sécurité de l'IoT étant un domaine très actif actuellement, il existe bien entendu d'autres approches mais que nous n'avons pas abordées ici, soit parce-qu'elles ne correspondaient pas à nos hypothèses, soit parce-qu'elles n'étaient pas le coeur de notre contribution. A titre d'exemple, les travaux de recherche sur les *Trusted Execution Environment* ou TEE auraient été pertinents mais auraient nécessité la modification matérielle des objets, ce qui était exclu dans

nos hypothèses de travail. D'autres travaux qui concernent par exemple la cryptographie légère adaptée aux objets connectés disposant de peu de ressources en énergie, sont également des travaux pertinents sur lesquels nous pourrions nous baser par exemple pour les communications sécurisées entre les différents objets et les serveurs centraux, mais cette partie sécurisation des échanges n'est pas considérée dans nos contributions. Nous faisons simplement l'hypothèse que cette sécurisation existe. Notre état de l'art a donc été sélectif vis-a-vis des solutions qui nous semblaient les plus pertinentes compte tenu de notre contexte, de nos hypothèses et de nos objectifs.

Pour chacune de ces solutions, nous en expliquons également les limites, qui, compte tenu des hypothèses et contraintes spécifiques de nos travaux, peuvent les rendre au final mal adaptées et nous justifions donc les raisons qui nous ont amené à privilégier une autre solution. Nous présentons donc dans une seconde partie, un état de l'art plus approfondi sur les mécanismes de détection d'intrusion, sur lesquels nous allons nous appuyer dans la suite des travaux.

3.1.1 Fonction non clonable physiquement

Nous abordons uniquement dans cette partie les *Physically Unclonable Functions* (PUF) électroniques, c'est-à-dire les PUFs appliquées à des circuits électroniques. Ce mécanisme de défense a pour objectif initial d'identifier un objet grâce à certaines caractéristiques uniques de ses circuits électroniques. Il est souvent comparé à une empreinte digitale, mais appliqué à des circuits électroniques. Ces caractéristiques uniques sont en fait définies par des imperfections infimes de fabrication, uniques à chaque circuit, et qui sont impossibles à reproduire précisément.

Une des idées de base est la suivante : en fabriquant deux fois le même circuit électronique et en mettant en entrée le même signal, le signal va se propager de manière relativement fiable plus rapidement dans l'un que dans l'autre. Grâce à ce principe, des circuits complexes sont créés, pouvant prendre en entrée différents signaux plus ou moins complexes, et proposant ainsi des sorties qui se veulent "imprévisibles", "uniques" et non-reproductibles (ou inclonables). Les PUFs sont ainsi spécialement conçues pour obtenir de telles caractéristiques. Une autre caractéristique intéressante de ces PUFs est la sensibilité à leur environnement physique, ce qui peut présenter à la fois un avantage et un inconvénient. Cela peut permettre de détecter certaines attaques modifiant physiquement l'environnement physique de la PUF, mais cela peut aussi poser problème lors du vieillissement de la PUF ou losrque la PUF change significativement d'environnement (au niveau de la chaleur ou des interférences électromagnétiques par exemple). La fiabilité des réponses d'une PUF suite aux entrées ou à l'envoi d'un "challenge" particulier n'est pas parfaite, mais la réponse donnée est toujours suffisamment proche de celle qui a été testée. Les PUFs sont donc souvent couplées à des codes correcteurs d'erreurs afin de pallier aux imperfections de réponses.

L'utilisation des PUFs pour l'identification d'un circuit électronique, et par extension pour celle d'un objet connecté, est donc basée sur un principe de chal-

lenge/réponse. La PUF est sollicitée avant le déploiement de l'appareil afin de modéliser ou de tester les couples challenge/réponse. Les couples permettent ensuite d'authentifier l'objet. Cependant, dans le cas où les messages pourraient être interceptés ou lus par un attaquant, et donc pour éviter d'éventuelles attaques par rejeu, chaque couple ne peut être utilisé qu'une seule fois.

Il est également possible d'utiliser les PUFs pour faire de l'obfuscation de secret, le principe étant de sauvegarder le challenge sur l'appareil et d'utiliser la réponse comme secret. Il suffit ensuite de faire une demande à la PUF uniquement lorsqu'il y a réellement besoin du secret (une clé de chiffrement par exemple). De la même manière, il est possible d'utiliser les PUFs pour chiffrer la mémoire de l'appareil.

Il existe différents types de PUF en fonction des principes électroniques et des circuits utilisés. Certaines PUFs sont basées sur la mémoire vive statique (*Static Random Access Memory*, SRAM en Anglais). Le principe est alors de récupérer l'état de cette mémoire lors de son alimentation. Cet état était considéré comme aléatoire mais il s'avère que certaines parties de cette mémoire ont un état suffisamment stable lors de l'alimentation pour en faire une PUF. Cela a été introduit dans les travaux de [Holcomb 2007]. Les PUFs basées sur les bascules vont utiliser des bascules formées de deux portes logiques NON-OU entrecoupées. Puis avec un signal permettant un *reset* de la bascule, celle-ci va se mettre d'abord dans un état instable puis se stabiliser pour donner un 0 ou un 1. Le principe est donc similaire aux PUFs basé sur la mémoire SRAM mais ne dépend pas d'une mise en alimentation. D'autres PUFs sont basées sur un principe d'arbitre, et ont été présentées initialement dans les travaux de [Lim 2005]. Le principe de cette PUF correspond à ce qui a été décrit dans le paragraphe d'introduction des PUFs : le principe de la course. Les auteurs proposent ainsi de faire la course entre deux signaux, mais les circuits (ou les chemins) pris par les signaux sont variables en fonction d'un challenge d'entrée. Chaque bit de ce challenge correspond à un changement de chemin. Le résultat dépend à la fin du résultats de la course, décidé par "l'arbitre". Le schéma 3.1 des travaux de [Maes 2010] qui sonde différentes manières de faire des PUFs, décrit bien ce principe. Les PUFs basées sur un oscillateur en anneau ont été introduites par [Gassend 2002]. Le principe est cette fois-ci de faire une boucle et d'utiliser le temps que le signal met pour parcourir la boucle. Il est ainsi possible de comparer et de paramétrer les circuits avec un challenge d'entrée comme pour les PUFs basées sur un arbitre.

Il est ainsi possible de distinguer deux types de PUFs, les fonctions "faibles" (*weak PUF*) et les fonctions "fortes" (*strong PUFs*). Les fonctions faibles ont un faible nombre de couples challenge/réponse, comme les PUFs basées sur la SRAM ou sur les bascules. En général ce type de PUFs sont directement intégrées dans le circuits de bases et servent plutôt au chiffrement de la mémoire ou à l'obfuscation de secret. Les fonctions fortes ont un grand nombre de couples challenge/réponse et sont en fait des circuits dédiés séparés du circuit fonctionnel normal de l'appareil, comme les PUFs basées sur des arbitres ou des oscillateurs en anneau. Ils sont ainsi plutôt utilisés dans le cadre de l'authentification.

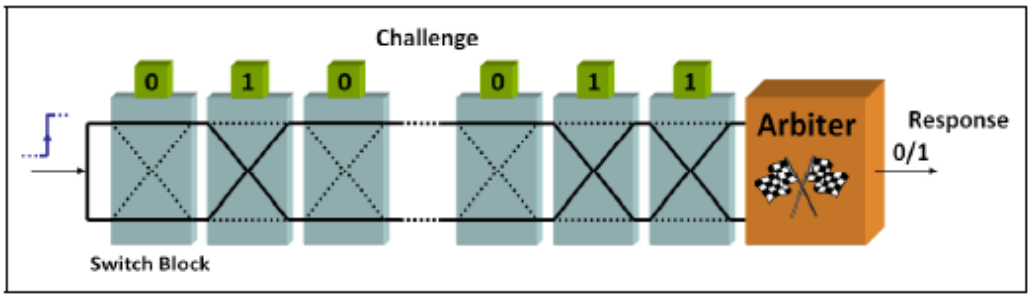


FIGURE 3.1 – Schéma montrant une PUF utilisant le principe de "course" d'après les travaux de [Maes 2010]

Enfin, les travaux [Halak 2018] présentent de manière plus complète les différents tenants et aboutissants des PUFs. Cependant comme nous ne retenons pas cette solution dans la suite, par soucis de lisibilité, nous ne détaillerons pas cette source d'informations. Dans le contexte de nos travaux, on peut souligner que les PUFs se révèlent plutôt compliquées à déployer et à utiliser dans le cas de déploiement de masse. En effet, il faut tester chacun des appareils déployés avant son déploiement afin de récupérer les informations sur la PUF nécessaire à son utilisation. Cela n'est donc pas en accord avec notre objectif **O3**. De plus les changements matériels que nécessitent les PUFs fortes ne correspondent pas à notre premier objectif. Enfin, les PUFs sont surtout utilisées dans le cadre de l'authentification de l'objet, ce qui n'est pas exactement ce que l'on a décrit dans nos hypothèses de menace.

3.1.2 Attestation à distance

Le mécanisme d'attestation à distance (*remote attestation* en anglais), consiste à s'assurer à distance que le programme exécuté sur l'appareil n'a pas été modifié. Pour cela, on identifie un vérifieur et un prouveur. Le but du vérifieur est de s'assurer que le programme du prouveur a bien été exécuté et qu'il n'a pas été modifié. Le vérifieur connaît ainsi l'état attendu du prouveur. En général, le vérifieur est considéré de confiance et l'étude n'est faite que pour vérifier la bonne exécution du programme sur le prouveur. Les attaquants considérés ont une bonne connaissance ou une connaissance complète du programme, et ont le contrôle sur la mémoire du prouveur. Cependant, l'attaquant ne peut modifier les capacités de calculs ou de mémoire du prouveur. En effet, les mécanismes d'attestation à distance utilisent ce type d'informations (taille de la mémoire ou temps de calcul) pour prouver la légitimité de l'exécution du logiciel.

Le principe d'attestation à distance consiste en trois grandes étapes : le challenge, l'attestation et la vérification. Le challenge, envoyé par le vérifieur, doit être authentique, unique et imprévisible. Le principe du challenge est que chaque demande d'attestation par le vérifieur soit unique et qu'un attaquant ne puisse préparer à l'avance une attestation qui pourrait être valide. De plus, le prouveur doit aussi être sûr que la demande d'attestation provient bien du vérifieur. La deuxième

étape correspond à l'étape d'attestation ou de création de l'attestation. Pour cette étape, le prouveur va utiliser le challenge envoyé par le vérifieur ainsi que son état, afin de créer une réponse. Il faut donc s'assurer que la réponse est authentique et qu'elle puisse être déterminée par le vérifieur pour qu'il puisse procéder à l'étape de vérification. Mais il faut également s'assurer que la fonction ne puisse pas être interrompue ou être recréée par un attaquant facilement. Enfin la réponse doit être dépendante de l'état actuel du programme exécuté, et non d'une partie statique de la mémoire. Enfin la dernière étape de vérification va utiliser l'état attendu du prouveur, le challenge, la réponse et parfois le temps de réponse, afin de déterminer la légitimité de la réponse et donc de l'état du prouveur. Les travaux [Steiner 2016], expliquent ces différents points en y ajoutant le schéma 3.2 décrivant succinctement le principe de ce mécanisme.

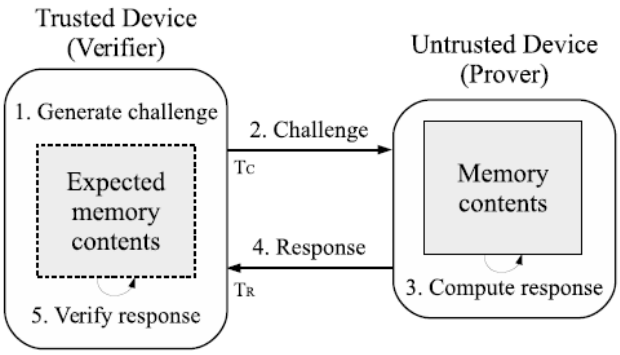


FIGURE 3.2 – Schéma montrant les différentes étapes de l'attestation à distance d'après les travaux de [Steiner 2016]

Les différentes techniques d'attestation à distance sont en général divisées en trois catégories :

— Attestation basée sur le matériel ou *Hardware-based remote attestation* : le principe général est d'utiliser certains composants matériels comme les fonctions inclonables physiquement (voir la section précédente) afin de s'assurer de l'intégrité et de l'exécution du logiciel légitime.

— Attestation basée sur le logiciel ou *Software-based remote attestation* : à l'inverse, cette catégorie essaie de n'utiliser aucun matériel supplémentaire. La vérification se fait uniquement via des échanges de messages entre le prouveur et le vérifieur et l'exécution du logiciel légitime.

— Attestation hybride ou *Hybrid remote attestation* : dans cette catégorie, l'idée est de tirer partie des deux mondes, et d'essayer de faire de la vérification à distance en minimisant les modifications ou ajouts matériels nécessaires.

Dans le cadre de nos travaux, nous avons déjà écarté des ajouts ou modifications du matériel dans la partie précédente. Nous pouvons donc écarter de notre étude l'attestation basée sur le matériel, comme les techniques de [Kong 2014] qui font

appel aux PUFs dont nous avons discuté dans la sous-section précédente. Concernant les approches basées sur le logiciel uniquement, les solutions demandent alors en général des communications fiables. En effet, une des composantes de ce type de mécanisme est le temps de réponse qui est précis. Or, dans un réseau qui n'est pas entièrement maîtrisé, il est complexe d'assurer cette caractéristique des communications. De plus, pour pallier au manque d'une racine de confiance prodiguée par le matériel, il faut parfois utiliser plus de ressources en terme de calculs ou de communications. SWATT [Seshadri 2004] est un exemple d'une des premières techniques d'attestation qui utilise un temps de réponse strict comme paramètre de réponse, et qui est basée uniquement sur le logiciel. Dans le cas de l'attestation hybride, les modifications ou utilisations matérielles nécessaires à cette catégorie d'attestation à distance doivent être minimes. Par exemple ces techniques peuvent reposer sur l'utilisation de la mémoire morte (ROM : *Read Only Memory* en anglais), qui permet de stocker le code de vérification de manière "sécurisée", c'est-à-dire sans que l'on puisse le réécrire. C'est ce que font notamment PoSE [Perito 2010], qui propose un mécanisme dont le code repose dans la ROM et qui permet de s'assurer que toute la mémoire hors ROM a bien été effacée avant une mise à jour. SMART [Eldefrawy 2012], est une autre solution qui va utiliser la mémoire ROM pour stocker le code de vérification du côté du prouveur.

Il est aussi possible de trouver des solutions plus distribuées et plus adaptées aux déploiements massifs d'objets. On peut retrouver ces techniques sous le nom de *swarm attestation* en anglais que l'on peut traduire littéralement en attestation par essaim ou nuée d'attestations. Le principe est donc de vérifier plusieurs exécutions en même temps, soit par la propagation de l'attestation dans le réseau, soit en proposant un système distribué ou les différents objets du réseau coopèrent pour s'attester les uns les autres. Par exemple, SEDA [Asokan 2015] propose une solution se propageant de proche en proche, avec un réseau organisé sous la forme d'un arbre, et où les attestations des prouveurs sont envoyées depuis les feuilles pour remonter jusqu'à la racine qui correspond au vérifieur final. Les appareils intermédiaires attendent donc les réponses de tous les appareils en dessous d'eux, et transmettent leur certification accompagnée de toutes celles reçues, après avoir reçu toutes les réponses ou après un certain temps. Cependant, ce type de solutions demandent plus de ressources aux objets en terme de communications et de calculs, car ils participent plus activement au processus d'attestation. Ainsi, [Ambrosin 2020] présente les différents travaux des dernières années concernant l'attestation dite collective, c'est-à-dire une attestation appliquée à un réseau plutôt qu'à un seul objet.

Pour résumer, les solutions basées sur le matériel ne sont pas adaptées à nos hypothèses à cause des changements matériels nécessaires et du coût ou de la complexité de déploiement que cela peut ajouter, et ne nous permettent pas de respecter les objectifs **O1** et **O3**. Les solutions basées sur le logiciel et les solutions hybrides quant à elles demandent un réseau totalement maîtrisé ou des communications et capacités de calcul trop élevées compte tenu de nos hypothèses et donc ne nous permettent pas de respecter l'objectif **O2**. Même si certaines solutions récentes es-

saient de pallier à ces problèmes, nous aborderons dans nos travaux une approche différente.

3.1.3 Diversification Logicielle

La diversification logicielle consiste à faire varier le logiciel d'un appareil à un autre ou d'une exécution à une autre afin de rendre le code plus complexe à analyser pour un attaquant, et aussi à rendre les attaques plus difficiles à répliquer ou à propager. En effet grâce à ce principe, une attaque peut fonctionner sur une version du logiciel mais par forcément sur les autres. Cette version du logiciel peut changer d'un objet à un autre, voire même d'une exécution à une autre. En d'autres mots les fonctionnalités proposées par le logiciel sont les mêmes mais implémentées différemment, d'un objet à un autre ou d'un moment à un autre. Les travaux de [Larsen 2014] expliquent et résument bien les principes et les différentes techniques de diversification logicielle. Nous présentons de manière succincte le principe général ainsi que quelques travaux de recherches associés dans la suite de cette section.

Il est possible d'effectuer des changements d'un logiciel pendant toutes les étapes de son cycle de vie. Nous allons donc présenter différents travaux de recherche relatifs à ces étapes : implémentation, compilation et création des liens, installation, chargement, exécution, mise à jour.

La phase d'**implémentation** dans le cycle de vie d'un logiciel correspond à la phase de création et de développement du code. Elle prend en compte par exemple le langage de programmation. Une diversification à ce niveau consiste ainsi à faire développer un même programme par des équipes différentes ou des langages différents [Avizienis 1977]. Ce principe, appelé en anglais le *N-versioning* est rarement utilisé dans le cas de la sécurité, mais a été utilisé dans le cadre de la sûreté de fonctionnement dans l'avionique, pour s'assurer de la résilience de certaines fonctionnalités critiques. Les fonctionnalités du logiciel ont donc été développées deux fois en parallèle pour éviter que le même problème logiciel puisse survenir en même temps sur les deux programmes.

La diversification au niveau de la phase de **compilation et de création des liens** est probablement plus intuitive et facilement automatisable. Le principe est de changer le programme lors de la compilation pour distribuer différentes variantes du même programme. Ainsi, une attaque sur l'une des variantes n'est pas forcément reproductible sur les autres versions. Appliquer les techniques à cette étape présente plusieurs avantages : il n'y a pas besoin de désassembler le code pour le changer et le code sera directement adapté à la plateforme matérielle correspondante. Par exemple dans les recherches de [Jackson 2013], les auteurs insèrent aléatoirement lors de la compilation des instructions NOP (*No Operation*), c'est-à-dire des instructions qui ne font rien. Cela a donc un impact sur l'optimisation du code réalisée par le compilateur, mais permet d'avoir une meilleure sécurité et plusieurs versions différentes du code que l'on peut déployer.

Concernant l'étape d'**installation** du logiciel, il est plus complexe de faire varier le programme. Souvent, un désassembleur est utilisé pour changer le comportement

du logiciel. En effet, à ce moment du cycle de vie du logiciel, nous avons uniquement accès au binaire. Ainsi la technique employée est de retraduire ce binaire en code à l'aide d'un désassembleur, de faire les changements sur le code désassemblé puis recompiler le code. Cependant les désassembleurs ne sont pas parfaits et certaines erreurs peuvent se produire, ce qui rend la diversification parfois incertaine mais aussi longue et complexe à cause du retour en arrière pour effectuer les changements. Dans [Pappas 2013], les auteurs discutent de la difficulté de désassembler le code et proposent une solution plus fiable, ne changeant que certaines parties du code plus simple à analyser et changer, mais réduisant ainsi le champ des possibilités de modifications.

Au **chargement** du logiciel, le changement se fait lorsque le programme est chargé en mémoire par l'OS. L'exemple le plus connu d'une diversification lors du chargement est l'ASLR (Address Space Layout Randomization), qui est utilisé par les OS depuis longtemps et qui charge le programme à des adresses mémoires aléatoires pour éviter qu'un attaquant puisse manipuler facilement ces adresses. Par exemple, les travaux de [Whitehouse 2007] montrent une analyse de l'utilisation d'un tel mécanisme dans le cadre de Windows Vista.

Pour la diversification lors de l'**exécution** du logiciel, l'idée est principalement de faire varier l'allocation des données dynamiques et des meta-données du "tas" (*heap* en anglais), ou de contrer des attaques prenant avantage de la génération de codes dynamiques des navigateurs par exemple. Ainsi [Jangda 2015] va insérer des instructions NOPs pour diversifier le programme créé dynamiquement dans le cas du langage Java et de sa mécanique de compilation à la volée (*Just in time*, JIT).

Enfin en ce qui concerne la **mise à jour**, le principe est de changer le programme significativement lors de chaque mise à jour. Cela permet non seulement de réduire le temps que l'attaquant possède pour analyser et compromettre le programme, mais aussi de "cacher" les vrais changements fonctionnels au milieu de multiples changements "virtuels" (sans conséquences fonctionnelles). Les travaux de [Collberg 2012] vont plus loin et proposent un mécanisme où le serveur de mise à jour envoie à chaque exécution du logiciel une nouvelle mise à jour. Dans leur cas, ce serveur et l'appareil exécutant le logiciel sont couplés et le serveur de mise à jour n'accepte les remontées d'informations de l'appareil distant uniquement dans le cas où celui-ci possède la version à jour du logiciel. Cependant cette solution demande beaucoup de communications entre le serveur et l'objet ainsi que beaucoup de ressources de l'objet qui doit constamment se mettre à jour.

Ces différentes techniques de diversification peuvent être classées en deux sous-catégories : la diversification pré-déploiement et la diversification post-déploiement. Concernant la diversification pré-déploiement, celle-ci est constitué des étapes d'implémentation, de compilation et création des liens, et de mise à jour. L'avantage des diversifications pré-déploiement est qu'elles vont demander moins de ressources lors de l'exécution du programme diversifié. Cependant la diversification sera moins efficace car le programme, sur un même appareil, peut être exécuté plusieurs fois exactement de la même manière. Concernant les techniques post-déploiement, celles-ci

sont en général plus efficaces du point de vue de la sécurité, mais engendrent la plupart du temps des pertes de performances plus importantes lors de l'exécution du logiciel. Notons cependant que l'ASLR est un contre-exemple de diversification post-déploiement efficace car il demande peu de ressources de la part du dispositif exécutant le logiciel, pour un résultat intéressant. Le schéma 3.3 proposé par [Larsen 2014] résume ces différentes approches de diversification logicielle.

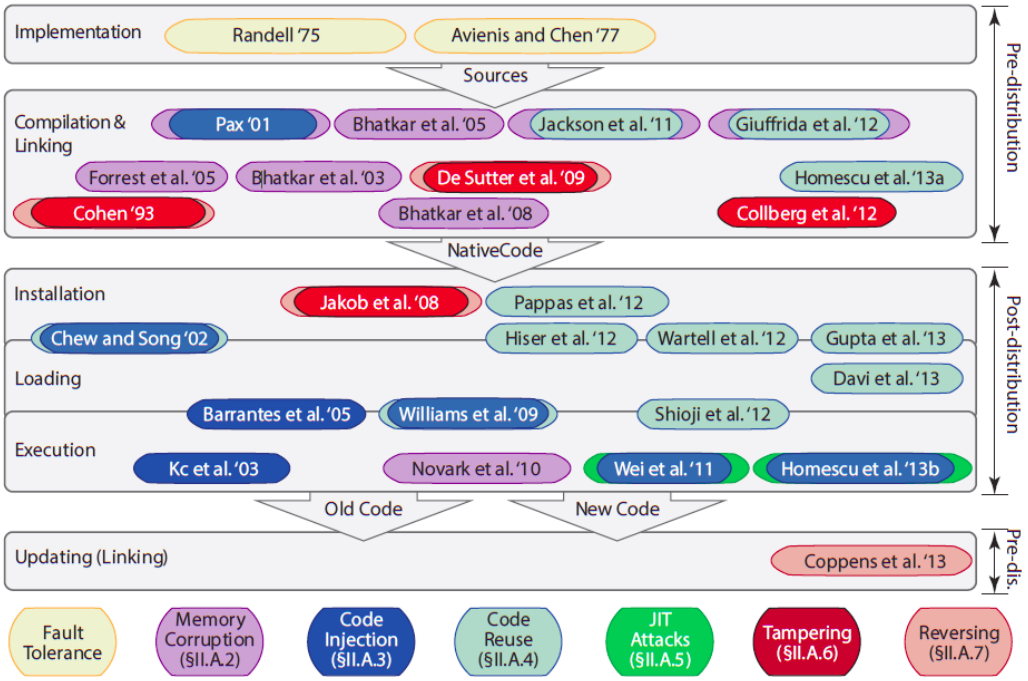


FIGURE 3.3 – Les catégories de différentes recherches en diversification logiciel des travaux selon [Larsen 2014]

La diversification peut également se faire à plusieurs niveaux du programme. Il a par exemple précédemment été mention du "tas" ou des instructions NOPs, mais voici une liste de ces niveaux :

— Instructions : il est possible en analysant le programme ou en utilisant un désassembleur d'échanger des instructions de places, ou d'ajouter des instructions "inutiles" comme des NOPs, ou enfin de remplacer une instruction par d'autres équivalentes.

— Bloc : il est possible d'interchanger certains blocs d'instructions, ou de changer la logique des différents sauts entre les blocs.

— Fonction : il est possible d'ajouter de l'aléa lors de l'appel de fonction et du comportement de la pile. Il est également possible de fusionner des fonctions ou de les intégrer directement dans le code si ces fonctions ne sont pas souvent appelées par exemple.

— Programme : il est possible avec une vision encore plus élargie de réordon-

ner les fonctions, de changer le chargement en mémoire (ASLR), ou encore d'encoder toutes les données par exemple.

Concernant l'applicabilité de cette technologie dans le cadre de nos travaux, on peut cependant remarquer certains défauts qui la rend inadaptée à nos cas d'usage. Les techniques de pré-déploiement ne sont pas facilement adaptables à toute plateforme matérielle et peuvent demander de grands efforts avant le déploiement du programme et rendre la maintenance des mises à jour plus complexe. Cela ne correspond donc pas à l'objectif **O3**. Les techniques post-déploiement, qui apportent en général un haut niveau de sécurité, demandent quant à elles trop de ressources aux objets surveillés, ce qui n'est pas compatible avec l'objectif **O2**.

3.1.4 Détection d'intrusion

La détection d'intrusion a pour objectif de détecter les attaques que des intrus peuvent mettre en œuvre sur des systèmes d'informations. Un système de détection d'intrusions est composé des trois composantes principales suivantes : des capteurs, un outil d'analyse et un système d'alerte [Zarpelão 2017]. En jouant sur ces différentes composantes, il est possible de définir différentes méthodes et manières d'opérer afin de détecter une intrusion. Nous allons ainsi discuter des caractéristiques suivantes dans la suite :

— le placement des capteurs permettant la détection d'intrusion

— la méthode de détection

Pour le placement des capteurs, on classifie en général les systèmes de détection d'intrusion basés sur le réseau (*Network based intrusion detection system* ou NIDS en anglais) et ceux basés sur les hôtes (*Host based intrusion detection system* ou HIDS en anglais). Ainsi les NIDS se concentrent sur l'analyse des communications lorsque les HIDS se concentrent davantage sur le comportement des hôtes auxquels ils sont liés. Les HIDS ont donc accès à des informations plus complètes sur l'hôte comme les programmes exécutés et leur temps d'exécution ou les communications du bus interne. Cette méthode est également facilement compatible avec le chiffrement des communications sur le réseau. En contrepartie, les HIDS vont consommer des ressources de l'appareil surveillé et ont besoin d'être installées sur tous les hôtes pour avoir une surveillance globale. Les NIDS quant à eux, ont la capacité de surveiller d'une manière plus globale le système, en surveillant uniquement les communications et donc sans avoir à intervenir sur les appareils surveillés. Cependant ils représentent ainsi des appareils supplémentaires à déployer pour surveiller le réseau et ont plus de difficultés à détecter les attaques venant du réseau interne. Il peut également être complexe d'analyser des liens ayant de très grands débits, ou de surveiller des échanges chiffrés, et seules les attaques se manifestant d'une façon ou d'une autre dans les communications réseaux peuvent être détectées.

Quant à la méthode de détection, on peut classifier les systèmes de détection en deux grandes catégories : les méthodes basées sur les connaissances et les méthodes basées sur le comportement.

La détection basée sur les connaissances désigne les systèmes utilisant des signatures d'attaques. Le principe est ainsi de comparer les informations captées (sur le réseau ou sur l'hôte) à une base de signature des attaques. Cette méthode est rapide et efficace pour détecter les attaques connues, et présente en général peu de faux positifs. Cependant elle nécessite alors une base de données de signature des attaques à jour et elle ne peut pas détecter des attaques qui n'ont pas une signature répertoriée. Ainsi elle ne permet pas de détecter de nouvelles attaques et nécessite une bonne gestion (mise à jour et accès) de la base de données des signatures constantes. Ce principe est souvent rencontré dans le cas de NIDS. les travaux de [Kumar 2007] par exemple montrent les différentes méthodes dans le cas de la détection réseau. La difficulté avec ce type de détection repose sur la détection de chaînes de caractères qui peuvent apparaître dans les messages échangés qui peuvent être considérés comme frauduleux.

La détection basée sur le comportement va interpréter les informations pour en déduire une caractérisation du comportement légitime. Le système va ainsi comparer le comportement courant au comportement légitime préalablement établi et lève une alerte lors de déviations de ce comportement légitime. La difficulté de cette méthode repose sur la définition du comportement légitime. Cela peut reposer sur des statistiques, des connaissances expertes ou des algorithmes d'apprentissage automatique (*machine learning* en anglais). Cette méthode permet ainsi de détecter plus de types d'attaques différents, mais est souvent sujette à des faux positifs, à cause de la définition du comportement légitime qui peut être complexe à aborder dans sa totalité. Cette méthode serait plus efficace pour détecter des attaques abusant des ressources ainsi que les nouvelles attaques, mais peut être plus lourde à supporter à cause des algorithmes utilisés pour les statistiques ou l'apprentissage automatique. On peut à titre d'exemple citer les travaux de [Smaha 1988], présentant ce principe d'utilisation de statistiques descriptives dans le cadre de comportement des utilisateurs. Mais on peut également citer plus récemment les travaux basés sur du machine learning comme ceux de [Chiba 2019]. Finalement, il est également possible d'utiliser simultanément différentes méthodes de détection afin de créer une détection plus complète, et permettant d'adapter les méthodes aux appareils surveillés dans le cas de réseaux ou de systèmes complexes. La gestion du système d'alarme peut dans ce cas devenir plus complexe. Par exemple il faut définir quand lever une alerte lorsque plusieurs méthodes ont un avis différent sur un même évènement. Les systèmes utilisés par les entreprises sont en général un mélange de différentes méthodes, en fonction d'une analyse de risque faite au préalable.

Outre la position des capteurs, il est aussi légitime de réfléchir au placement de l'outil d'analyse. Autrement dit, quel appareil ou partie du réseau est en charge d'exécuter les algorithmes d'analyse des données ? En effet, il est possible de placer cette fonctionnalité de plusieurs manières différentes :

— **Centralisée** La détection est réalisée sur un seul appareil qui doit donc avoir à sa disposition toutes les informations nécessaires. Cela permet d'alléger le travail des autres appareils surveillés qui ont juste besoin de communiquer les

informations nécessaires à la détection. En revanche cela augmente considérablement les capacités nécessaires pour effectuer la détection par l'appareil central. De plus, cette méthode est celle qui utilise le plus les canaux de communications afin de faire transiter les différentes informations.

— **Distribuée** La détection se fait "localement", et chaque partie du réseau participe à la détection d'intrusion autrement que par le simple envoi d'informations. Ainsi, les appareils peuvent par exemple s'entraider afin de vérifier qu'il n'y ait pas un intrus. Cette méthode permet de partager les coûts en ressources (temps de calcul, mémoire, communications) de la détection. Elle peut parfois se révéler complexe à déployer de par l'hétérogénéité des appareils du réseau.

— **Hybride** L'idée est cette fois-ci de choisir certains appareils en particulier qui ont les capacités et la possibilité d'effectuer le travail d'analyse. Nous sommes donc dans une situation mélangeant les deux dernières solutions, permettant de partager le travail et les communications nécessaires à la détection et facilitant la compatibilité des appareils déployés à la solution. Il est ainsi possible de créer des sous-groupes surveillés sous la responsabilité d'un seul appareil. Il est également possible de créer une hiérarchisation des appareils de surveillance, et de faire surveiller ces noeuds par un noeud central par exemple. Cela permet d'optimiser la détection en fonction du niveau de surveillance souhaité dans le cas de réseaux complexes comprenant différents sous-réseaux distincts par exemple.

Dans notre cas, au vue de l'hypothèse **H1** (stipulant que les objets seront déployés massivement et seront identiques) et de l'objectif **O2** (solution demandant peu de ressources aux objets déployés) il semble que la solution centralisée soit la plus adaptée. De plus, les solutions utilisant les signatures ne semblent pas adaptées aux types d'attaques que l'on voudrait repérer. En effet, les nouvelles attaques ne peuvent être détectées de cette manière. De ce côté là, les solutions basées sur le comportement semblent plus adaptées pour répondre à l'hypothèse de menace **HM2** (utilisation des objets par l'attaquant) car plus efficace d'un ordre général pour détecter des abus de ressources. Finalement, la gestion des signatures, qui doivent constamment être mises à jour, est un inconvénient qui est incompatible avec notre objectif **O3** (simplicité de déploiement, de maintenance, c'est-à-dire peu ou pas d'intervention à faire lors de mise à jour).

3.1.5 Conclusion des approches relativement aux hypothèses

Nous allons dans cette partie résumer en utilisant le tableau 3.1 les différentes approches et leurs inconvénients vis-à-vis de notre contexte. Les différentes solutions ont été présentées globalement et non exhaustivement au cas par cas.

Dans la suite nous allons nous intéresser plus en détails aux solutions de détection d'intrusion basées sur le comportement. En effet, les autres solutions semblent présenter des limites ne correspondant pas à nos hypothèses. Il y a de nombreuses

TABLE 3.1 – Résumé des solutions étudiées

			Hypothèses, objectifs ou hypothèses de menaces non respectés
Solutions étudiées	Diversification logicielle	Post-déploiement	**O2** ou **O3**
		Pré-déploiement	**O3**
	PUF	Weak	**O1** ou **HM2**
		Strong	**H2**, **O1**
	Attestation à distance	Matériel	**O1** ou **O2**
		Logiciel	**O2** ou **O3**
		Hybride	**O1** ou **O2** ou **O3**
	Détection d'intrusion	Signature	**HM2**
		Comportement	**O3**

possibilités de mettre en œuvre la détection d'intrusion, mais ses avantages semblent bien correspondre au type de détection que l'on aimerait obtenir. En outre, certaines solutions de détection d'intrusion ne demandent que très peu de ressources de l'objet, ce qui est un critère important dans notre contexte. Nous allons donc dans la suite détailler la taxonomie de ce mécanisme de défense.

3.2 Taxonomie de la détection d'intrusion basée sur le comportement

Nous allons dans cette partie présenter plus en détails les différentes solutions de détection d'intrusion basées sur le comportement que l'on peut trouver dans la littérature. La détection par le comportement vise à modéliser le comportement légitime et à considérer que toute déviation de ce modèle lors de l'observation correspond à une anomalie. Le modèle du comportement légitime peut être obtenu de deux manières différentes : soit à partir de la spécification, soit à partir de l'observation du comportement usuel, en faisant l'hypothèse que ce comportement usuel est un reflet du comportement légitime. L'analyse du comportement usuel peut se faire soit avec des méthodes statistiques, soit avec des algorithmes d'apprentissage automatique de modèles.

Dans un premier temps, nous allons définir des principes communs des systèmes de détection d'intrusion. Les systèmes de détection ou de classification peuvent commettre des erreurs. Il peut donc y avoir une différence entre ce que le système détecte et la réalité. De plus, ces erreurs ne sont pas comparables : dans certains cas, il peut être acceptable de ne pas détecter une attaque mais inacceptable que le système lève une alerte à tort, alors que dans d'autres applications cela pourrait être le contraire. Le diagnostic du système de détection d'intrusion est dit "positif" lorsqu'il lève une alerte et "négatif" lorsqu'il ne lève aucune alerte. Si ce diagnostic est juste, on le qualifie de "vrai" et s'il est erroné, on le qualifie de "faux". En

conséquence, on divise donc les différentes possibilités de succès et d'erreur en quatre catégories :

- Les vrais positifs (TP) : le système de détection détecte correctement un comportement anormal
- Les vrais négatifs (TN) : le système de détection considère avec raison qu'il n'y a aucun comportement anormal
- Les faux positifs (FP) : le système de détection détecte un comportement anormal et lève une alerte alors qu'aucune attaque n'est en cours
- Les faux négatifs (FN) : le système de détection ne détecte aucun comportement anormal alors qu'une attaque est en cours

Afin de tester l'efficacité d'un système de détection d'intrusions, il est possible d'utiliser plusieurs métriques. On peut simplement utiliser le nombre de faux positifs ou de faux négatifs par exemple, mais il est également courant d'utiliser d'autres métriques qui sont des combinaisons de ces différentes valeurs. Ces métriques sont très souvent utilisées par la communauté et elles sont au nombre de 5 : *precision, accuracy, recall, F1 score et taux de faux positifs*. Nous préférons utiliser les termes anglais pour certaines définitions car ces termes sont moins ambigus que les équivalent français.

- *accuracy* : Le taux de bonnes prédictions parmi l'ensemble de toutes les prédictions (bonnes ou mauvaises). Cela donne une idée de la confiance que l'on peut avoir aux résultats de la solution globalement, relativement au jeu de données utilisé.
$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.1}$$
- *recall* : Le taux d'attaques bien détectées parmi toutes les attaques du jeu de données. On peut également le retrouver sous le nom de taux de détection. Cela montre l'efficacité de la solution à détecter des attaques au sens large (sans distinguer les types d'attaques différents du jeu de données) relativement au jeu de données utilisé.
$$recall = \frac{TP}{TP + FN} \tag{3.2}$$
- *precision* : Le taux d'alertes levées par le détecteur qui correspondent effectivement à des attaques. Cela donne une idée de la confiance que l'on peut donner à une alarme relativement au jeu de données utilisé.
$$precision = \frac{TP}{TP + FP} \tag{3.3}$$
- *F1 score* : score qui est la moyenne harmonique de la *precision* et du *recall*. Plus le score est proche de 1 et plus la détection est efficace de manière globale,

sans privilégier certains types de résultats.

$$F1 = 2\frac{recall \times precision}{recall + precision} \tag{3.4}$$

- taux de faux positifs : ou taux de fausses alarmes qui est le taux de comportements légitimes considérés à tort comme des attaques. C'est la proportion d'alertes que l'on aurait en l'absence d'attaque.

$$FPR = \frac{FP}{FP + TN} \tag{3.5}$$

Lorsqu'il y a un déséquilibre dans le jeu de données entre les attaques et les données légitimes, comme cela est le cas en détection d'anomalies, l'*accuracy* perd en pertinence. Par exemple, s'il y a 1% d'attaque dans le jeu de données, alors un système de détection qui ne lève jamais d'alerte aura 99% d'*accuracy*. On préfère dans ces cas là utiliser le *recall* et la *precision* ensemble, ou le F1-score qui sont des métriques plus représentatives de l'efficacité du système de détection dans ce cas-là. Dans l'exemple de ce détecteur, son recall sera nul car il n'y aura aucun vrai positif.

3.2.1 Spécification

La spécification peut être utilisée pour définir le comportement normal d'un système. À partir de cette spécification, des experts pourront en déduire des règles qu'un comportement normal ne devrait jamais violer. Ces méthodes permettent généralement d'obtenir de bonnes performances et notamment peu de faux positifs, car l'expert peut s'assurer que le comportement légitime du système, tel que défini par la spécification, ne peut pas enfreindre les règles qu'il a écrites. Par contre ce type de détection demande l'implication d'un expert et nécessite des mises à jour par cet expert lorsque la spécification du système vient à être modifiée, par exemple suite à une mise à jour logicielle. Enfin, ces méthodes présument l'existence d'une spécification précise et pérenne du système, ce qui n'est pas toujours le cas dans une méthode de développement agile.

Par exemple, les travaux de [Sekar 2002] se basent sur la spécification pour concevoir un système de détection d'intrusion de type NIDS. Le principe est dans un premier temps de définir un modèle de type automate fini étendu à partir des standards et de la documentation de certains protocoles comme TCP par exemple. Les différentes transitions correspondent alors aux messages échangés et les états à l'état de l'objet d'après les spécifications du protocole de communications. Par exemple on peux citer : ACK_WAIT, SYN_RECD, SYN_SENT, CLOSE_WAIT, CLOSE. Ces échanges sont classés par adresse IP. La détection se fait ensuite en utilisant l'automate, c'est-à-dire en utilisant les transitions entre les différents états pour chaque séquence de communication et par adresse IP comme caractéristique à analyser. Cela simplifie la représentation des différents échanges, qui serait plus classiquement sous la forme d'une longue liste. Cette représentation des communi-

cations évite ainsi la surcharge d'informations et d'analyses que peut représenter une longue séquence de paquets échangés. La méthode nécessite une phase d'entraînement et d'étude du protocole (spécifications) avant la phase de détection, afin de décider du comportement légitime des caractéristiques des échanges précédemment choisis.

Les auteurs de [Su 2006] présentent un système de détection d'attaques d'injection de commande (comme les injections SQL par exemple) en se basant sur la spécification de ce langage, ou plus exactement sa grammaire. Les injections de commandes consistent à utiliser des champs d'écritures qui lui sont disponibles afin de manipuler le code et par exemple d'obtenir des accès ou des informations qu'il n'est pas censé avoir. Les auteurs remarquent qu'une entrée utilisateur légitime doit être un sous-arbre de l'arbre syntaxique. Pour détecter ces attaques, ils vérifient donc si l'entrée utilisateur satisfait ou non cette condition.

3.2.2 Détection statistique

La détection statistique classifie les comportements en faisant une étude statistique des différentes valeurs remontées par les capteurs sur le réseau ou sur les hôtes. Dans le cas d'une approche de détection basée sur le comportement, le système de détection compare le profil courant des valeurs avec un profil normal de fonctionnement, statistiquement, et lorsqu'il constate des déviations, le système lève alors une alerte.

Par exemple, les travaux de [Smaha 1988], qui sont parmi les premiers à élaborer un tel système de détection (en 1988), proposent une détection d'utilisateurs malveillants en considérant les différentes mesures comme des variables Gaussiennes indépendantes. Ainsi, si certaines mesures (connexion, déconnexion, activités) sortent des limites "normales", alors le score "d'anormalité" est augmenté en conséquence pour cet utilisateur. Comme les caractéristiques sont considérées indépendantes, les auteurs utilisent ensuite la probabilité de distribution des scores. Cependant ce système de détection n'a pas pu être réalisé en temps réel à cause des performances requises trop élevées pour cette époque. De plus, il faut définir des profils types d'utilisateur puis les maintenir "manuellement", ce qui n'est pas pratique pour les administrateurs qui peuvent avoir des difficultés à déterminer les attributs qui permettent de distinguer différents profils d'utilisateur.

Dans les travaux plus récents, on peut citer la solution proposée par [Ara 2017]. Elle est basée sur l'analyse conjointe de séries temporelles et a été appliquée à des serveurs Web. Les caractéristiques des serveurs surveillés sont les taux d'erreurs de synchronisations et de rejets de connexions. Les auteurs ont pu prévoir avec un certain intervalle de confiance une prédiction de ces données dans le futur. Ainsi il est possible d'utiliser ce modèle de prédiction afin de créer un système de détection d'intrusion. Si les données réelles correspondent (à une marge d'erreur près) à la prédiction c'est qu'il n'y a pas d'attaques, sinon une alerte est levée. Même si les auteurs ont utilisé dans leur cas une régression d'un système à deux variables, ils précisent que leur méthode peut être étendue afin d'y inclure une analyse avec plus

de variables. Il est ainsi possible d'y ajouter d'autres mesures souvent utilisées dans le cadre de la détection d'intrusion des hôtes comme l'activité du processeur, la bande passante, la mémoire utilisée, etc.

On peut également citer des travaux sur les réseaux de véhicules connectés comme ceux de [Zaidi 2015]. Ils se rapprochent ainsi un peu plus du contexte dans lequel nous travaillons. Le contexte présenté repose sur un réseau local de voitures connectées qui communiquent entre elles pour partager des informations sur la circulation. Ainsi, comme les véhicules ont des contraintes environnementales similaires car géographiquement proches, les valeurs partagées (vitesse et nombre de véhicules connectés environnants) devraient être similaires. Les véhicules calculent alors le flux environnant, c'est-à-dire la densité (le nombre de véhicules environnants) multipliée par la vitesse moyenne des véhicules environnants. Les auteurs affirment alors que dans des conditions normales, cette valeur de flux est expérimentalement toujours à moins de deux fois l'écart-type de la distribution de cette valeur parmi tous les véhicules environnants. Les véhicules ayant des valeurs de flux ne respectant pas cette règle sont alors considérés comme des attaquants et leurs valeurs partagées ne sont pas prises en compte dans les calculs de la voiture.

3.2.3 Détection basée sur de l'apprentissage automatique

Ce type de détection se base sur des algorithmes d'apprentissage automatique (*machine learning*) afin de définir le comportement légitime et/ou le comportement des attaquants. Cette catégorie est assez large car il existe beaucoup d'algorithmes de ce type qui peuvent être utilisés dans le cadre de la détection d'intrusions. Nous allons donc différencier les différentes techniques utilisées en fonction des données d'apprentissage nécessaires qui permettent de distinguer les attaques des comportements légitimes. Il y a ainsi trois principaux types d'apprentissage qui peuvent être utilisés : supervisé, semi-supervisé et non-supervisé.

Ces types d'apprentissage se différencient par les données disponibles à l'algorithme d'apprentissage. En apprentissage supervisé, les données contiennent à la fois des observations et leur classe : attaque ou légitime. En apprentissage non-supervisé, la classe des données n'est pas connue et les données mélangent attaques et comportements légitimes. Enfin, en apprentissage semi-supervisé, les observations sont toutes d'une seule classe (légitime, par exemple).

3.2.3.1 Apprentissage supervisé

L'apprentissage supervisé, dans le cas de la détection d'intrusion, utilise des données d'apprentissage du comportement légitime mais également des attaques, afin de construire une frontière entre les deux types de données. Le principe est ensuite d'utiliser le modèle des comportements créé par l'algorithme d'apprentissage pour vérifier la présence ou non d'une attaque. Il y a ainsi une première phase de génération de données d'apprentissage qui se fait en général hors-ligne, puis la phase de détection avec le modèle créé lors de la première phase. Cependant, si le comportement du système change de manière significative, alors il faut exécuter de

nouveau la phase d'apprentissage hors-ligne.

Les travaux de [Premaratne 2009] présentent le cas de l'utilisation d'un arbre de décision afin de créer un système de détection très léger. Le principe est d'entraîner l'arbre de décision avec des attaques et des données légitimes puis de créer des classes par types d'attaques ou de comportements légitimes. Les données en temps réel sont analysées avec cet arbre de décision pour retrouver la catégorie des actions d'un utilisateur. Les différentes catégories utilisées dans ces travaux sont le téléchargement bénin, la navigation légitime, une attaque de type inondation de ping et une attaque brute-force ciblant les mots de passe. Les travaux de [Koroniotis 2019] se focalisent plus sur les données réseaux dans le contexte de l'IoT. Ils conçoivent tout d'abord une base de données relatives à un environnement IoT avant de tester trois méthodes supervisées différentes sur ce jeu de données. Les méthodes supervisées utilisées sont les suivantes : *Support Vector Machine* ou SVM, *Recurring Neural Network* ou RNN, et *Long Short Term Memory RNN* ou LSTM-RNN.

Leurs résultats montrent que SVM est l'algorithme le moins efficace en terme de détection (*accuracy* et *recall* de 88%) mais 8 fois plus rapide que RNN et 10 fois plus que LSTM-RNN. Ces deux derniers quant à eux obtiennent une excellente détection (*accuracy* et un *recall* supérieur à 97%).

[Koc 2012] présente l'utilisation de classificateur bayésien dans le cas de la détection d'anomalies. Il présente ainsi différents types de classificateurs bayésiens avant d'introduire la solution adoptée dans leur travaux : le classificateur bayésien naïf caché (*hidden naïve bayes classifiers*, ou HNB). Un classificateur bayésien associe à chaque classe une probabilité qui dépend des données observées. La version naïve considère que toutes les caractéristiques sont indépendantes et enfin la version naïve cachée ajoute une influence entre les caractéristiques. Par exemple si les 3 caractéristiques étudiées sont : A1, A2 et A3, alors la probabilité de la valeur de A1 sera influencée par la classe et par la combinaison des valeurs de A2 et A3. Cela permet d'alléger l'hypothèse d'indépendance des caractéristiques, qui est rarement vérifiée dans les systèmes.

3.2.3.2 Apprentissage semi-supervisé

L'apprentissage semi-supervisé, dans le cas de la détection d'intrusion, peut être décomposé en deux sous-catégories : la classification à une seule classe (One-class classification), où on ne va utiliser que des données d'une seule classe afin de créer un modèle permettant de délimiter ce comportement, et l'apprentissage à partir d'exemples qui sont soit d'une certaine classe, soit non-labelisés (on ignore s'il s'agit d'exemples d'attaques ou de comportements légitimes), qu'on appelle l'apprentissage PU (Positive/Unlabeled learning). La plupart du temps, l'apprentissage se base sur la modélisation du comportement légitime par l'algorithme. Cela évite la mise en place complexe d'attaques dans le but d'obtenir des données (ce qui est fait dans l'apprentissage supervisé) et d'autre part cela permet au détecteur de découvrir de nouvelles attaques (ce qui est plus difficile en apprentissage supervisé). Cependant si le comportement légitime est modifié via une mise à jour par exemple, alors il faut refaire une phase de collecte de données et d'apprentissage.

[Forrest 1996] présente ainsi une solution de type HIDS, où les appels systèmes sont surveillés. Le principe est de collecter des séquences d'appels systèmes sur des fenêtres glissantes lors de l'exécution d'un programme. Il faut dans un premier temps récupérer les séquences "normales" afin de les enregistrer. Puis les auteurs comparent au moment de l'exécution du programme la séquence actuelle aux séquences normales afin de définir si l'exécution du programme semble légitime. Les auteurs proposent d'utiliser cette technique pour surveiller les programmes ayant un privilège élevé, qui se révèlent plus dangereux que les autres programmes. La difficulté de cette technique repose dans le choix du comportement normal de l'application surveillé. Il est en effet nécessaire d'enregistrer les différentes variantes d'exécutions du logiciel ciblé afin de s'assurer d'avoir caractérisé tous les types de comportements légitimes. Il faut ainsi connaître les applications surveillées afin de d'obtenir une liste exhaustive des comportements légitimes.

[Damien 2020] utilise également des appels systèmes afin de mettre en place un HIDS dans le cadre de l'avionique. L'approche mets un accent particulier sur le système de reconnaissance de l'alerte. Ainsi, l'approche utilise une base de connaissance de signature pour essayer de déterminer dans un premier temps si une alerte semble correspondre à une attaque ou à un incident lié à la sûreté de fonctionnement. En cas d'anomalie ne correspondant à aucune signature, les informations sont sauvegardées pour une investigation approfondie au sol. Cette base de connaissance de signature est dans un premier temps remplie avec des attaques ou incidents connus, avant d'être améliorée et complétée avec les nouvelles données fournies à l'atterrissage des avions. Un classificateur bayésien naïf est utilisé pour l'étape de recherche de correspondance de signature dans la base de connaissance.

Les travaux de [Ravi 2020] montrent l'utilisation d'un algorithme *Deep Neural Network* (DFNN) ou réseau de neurones à apprentissage profond, et une version alternative d'un algorithme de regroupement des données k-means appelé RSS-k-means afin de définir les anomalies. Ils utilisent ainsi des données non-labellisées et affinent les résultats avec quelques données labellisées. Le DFNN utilise ainsi les données avec un label afin de faire une classification des différentes attaques. Les données non-labellisées sont alors utilisées lors de la fin de la phase d'utilisation de RSS-k-means, c'est-à-dire la phase de comparaison des distances euclidiennes, afin de créer les délimitations des groupes d'attaques et de données légitimes en sélectionnant des données des différentes catégories d'attaque, des données légitimes et les données non-labelisées. Les auteurs analysent ainsi le réseau afin de déterminer si les communications venant d'objets connectés distants sont légitimes ou font partie d'une attaque de déni de service distribuée (DDoS).

3.2.3.3 Apprentissage non-supervisé

L'apprentissage non-supervisé se différencie des deux autres types d'apprentissage car il n'y a pas une distinction entre phase d'apprentissage hors ligne et phase de détection en ligne : la détection est faite directement via l'apprentissage.

Il y a deux grandes familles de tâches d'apprentissage non-supervisé : le *clustering*, qui vise à regrouper les données en groupes, et la détection de valeurs aber-

rantes (*outlier*), qui vise à trouver les anomalies au sein d'un groupe de données sans apprentissage préalable. En détection d'anomalie, c'est cette seconde famille de techniques que nous pouvons utiliser.

Comme il n'y a pas de phase hors ligne nécessaire, cela permet de rendre la détection très dynamique et le déploiement et la maintenance beaucoup plus simple. En revanche, ces techniques font l'hypothèse qu'il y a peu de valeurs aberrantes dans le jeu de données. Il faut donc être sûr que les comportements légitimes soient bien en nombre supérieur par rapport aux attaques pour éviter trop de fausses alarmes ou de faux négatifs. Il y a de nombreux algorithmes qui permettent de faire de la détection d'anomalies. Nous n'allons cependant pas présenter exhaustivement ici toutes les possibilités d'algorithmes.

Par exemple [Ertoz 2004] proposent un système de détection utilisant l'algorithme LOF (*Local Outlier Factor*). Cet algorithme va comparer sa densité locale à celle de ses plus proches voisins. Cela permet aux auteurs d'inspecter scrupuleusement uniquement les données considérées anormales par LOF. L'analyse porte sur des fenêtres de temps fixes d'analyse du réseau. Le comportement sera défini par les différents flux de communications. Les travaux de [Zhang 2006] présentent quant à eux l'utilisation d'une forêt d'arbres décisionnels ou forêt aléatoire (*random forest* en anglais). Ils analysent ainsi les communications, divisent les différentes communications en fonction du service correspondant à la communication (http, ftp, ...), puis appliquent l'algorithme pour essayer de détecter tout comportement d'échanges anormaux considérant le service. Afin d'optimiser les paramètres de l'algorithme, un test préliminaire est effectué sur le jeu de données. En effet, les algorithmes de détection de valeurs aberrantes, bien que ne nécessitant pas de phase d'apprentissage, ont souvent des paramètres qui peuvent être adaptés en fonction de la situation. Les résultats présentés montrent une détection similaire mais un taux de faux positifs inférieur à d'autres solutions de détection d'anomalies basée sur le regroupement de données ou sur les plus proches voisins.

Pour se situer un peu plus dans notre contexte, les travaux de [Ayadi 2017] présentent dans le cas des réseaux de capteurs sans fils (WSN) l'utilisation de différents types de détection de valeurs aberrantes. Ils présentent ainsi les besoins du contexte puis les avantages et les inconvénients de plusieurs catégories d'algorithmes de détection de valeurs aberrantes. Les différentes catégories d'approches présentées sont basées sur les statistiques, basées sur les plus proches voisins, basées sur de l'intelligence artificielle, basée sur du regroupement et basée sur la classification.

3.3 Conclusion

Dans ce chapitre nous avons tout d'abord passé en revue un ensemble de solutions de sécurité qui nous paraissaient constituer de bons candidats adaptés à nos objectifs : la diversification logicielle, les fonctions inclonables physiquement et la détection d'intrusion. Après avoir présenté chaque solution, et des exemples de travaux qui ont attiré notre attention, nous avons également relevé les avantages et

inconvénients de chacune de ces solutions par rapport à nos objectifs.

Au final, la détection d'intrusion basée sur de l'apprentissage automatique est selon nous la plus adaptée. Nous avons donc, dans un second temps, présenté les grands principes de la détection d'intrusion ainsi qu'une classification des approches essentielles utilisées aujourd'hui dans ce domaine. Nous distinguons ainsi les approches aux apprentissages supervisés, semi-supervisés et non supervisés. L'objectif **O3** spécifie que la solution doit être simple à déployer et à maintenir, par exemple avec les mises à jour. Il semble ainsi que la meilleure solution pour remplir cet objectif correspond à un apprentissage non-supervisé. Cela permet d'envisager une solution déployable sans collecte massive d'informations au préalable et sans nécessité de faire des collectes supplémentaires de données lors des mises à jour des objets. Autrement dit, au regard de cet objectif, nous préférons ne pas utiliser une approche nécessitant la création de modèle de comportement des objets connectés. Le prochain chapitre est ainsi consacré à la présentation d'une nouvelle approche permettant de réaliser la détection d'anomalies non-supervisée en se basant sur la collecte et l'analyse des compteurs matériels de performance. Cette approche permet d'éviter la création de modèles complexes du comportement de l'application qui s'exécute sur l'objet. Elle s'appuie simplement sur l'observation et l'analyse comparative des comportements des objets sur des fenêtres temporelles courtes, afin de déterminer des valeurs aberrantes.

Chapitre 4

Détection d'anomalies basée sur le comportement du processeur

Sommaire

4.1 Les compteurs matériels de performance 63
4.1.1 Description 64
4.1.2 Les compteurs matériels pour la sécurité 65
4.1.3 Les critiques sur l'utilisation des compteurs 68
4.2 Une approche mêlant HPC et détection d'anomalies 70
4.2.1 Vue d'ensemble 70
4.2.2 Lecture 71
4.2.3 Collecte 73
4.2.4 Extraction des caractéristiques 73
4.2.5 Standardisation 74
4.2.6 Détection d'anomalies 75
4.3 Choix des algorithmes de détection de valeurs aberrantes . 76
4.3.1 Généralités 76
4.3.2 Algorithmes basés sur les plus proches voisins 77
4.3.3 Algorithmes basés sur la classification 79
4.3.4 Algorithmes basés sur du regroupement 80
4.4 Conclusion 81

Ce chapitre débute par un état de l'art concernant les informations que nous avons choisies d'utiliser pour détecter des attaques : les compteurs matériels de performance. Nous présentons ainsi différents travaux de recherche consacrés à l'utilisation de ces compteurs ainsi que sur leur efficacité en ce qui concerne la sécurité. Dans un second temps, grâce à ces informations, nous présentons notre approche et en détaillons chaque étape par ordre chronologique. Enfin nous présentons les différents algorithmes de détection d'anomalies qui peuvent être utilisés dans la dernière étape de notre approche. Nous les présentons en les classant par famille, avec leurs avantages et inconvénients dans le cadre de notre approche.

4.1 Les compteurs matériels de performance

Différents travaux de recherche se sont intéressés aux compteurs matériels de performances. Nous allons dans cette section principalement nous intéresser à leur

utilisation dans le cadre de la sécurité. Ainsi après avoir décrit ces compteurs, nous présentons un état de l'art ainsi que certaines critiques concernant leur utilisation dans ce contexte.

4.1.1 Description

Les compteurs matériels de performance (ou *Hardware Performance Counters*, HPC en anglais) sont des compteurs internes aux processeurs permettant de compter le nombre d'occurrences de certains évènements architecturaux ou micro-architecturaux. Les évènements qu'il est possible de compter peuvent par exemple être le nombre d'accès aux différents caches mémoire du processeur (cache L1, L2, ...), le nombre de *cache-miss*, le nombre d'instructions exécutées, le nombre de branches mal prédites, le nombre de cycles du processeur, etc. Ces compteurs permettent ainsi d'obtenir des informations sur le comportement "bas-niveau" du processeur. Ils sont tout d'abord utilisés pour optimiser les logiciels et avoir une mesure des performances de certains logiciels vis-à-vis de leur exécution sur le processeur. Le nombre et le type d'évènements qu'il est possible de récupérer et de surveiller dépendent du processeur utilisé.

Selon le type de processeur, le nombre d'événements différents que l'on peut surveiller peut varier d'une vingtaine à plusieurs centaines d'évènements. Par ailleurs, le nombre de compteurs est généralement inférieur au nombre d'évènements différents. Donc, il n'est pas possible de collecter tous ces évènements simultanément. Pour surveiller un de ces évènements, il faut configurer des registres spéciaux. Tous les processeurs ne possèdent pas de tels registres, mais ils sont inclus de plus en plus dans les nouveaux processeurs. On trouve en général de 2 à 8 registres spéciaux disponibles par coeur, mais ce nombre tend à augmenter. Dans nos travaux, nous ciblons le processeur ARM, un processeur souvent utilisé dans le cadre de l'IoT. Ce qui suit concerne ainsi principalement ces processeurs, mais le fonctionnement des autres processeurs est similaire.

La modification des registres de configuration des compteurs nécessitent des privilèges spécifiques (de niveau noyau du système). Cependant, il est possible de rendre ces registres accessibles en mode utilisateur grâce à un module noyau qui permet de changer un registre de contrôle d'utilisation des compteurs. Certains outils ont également été proposés afin de récupérer l'évolution des compteurs. On peut citer par exemple perf_event [Weaver 2013a] et PAPI [Mucci 1999].

La plupart des processeurs modernes ont également une unité de surveillance de performance (*Performance Monitoring Unit*, ou PMU). Cette unité permet de déclencher une interruption lorsqu'un seuil de nombre d'évènements est dépassé. Par exemple il est possible de générer une telle interruption toutes les 100 000 instructions exécutées.

Les compteurs matériels de performance permettent ainsi d'obtenir des informations sur le comportement bas-niveau du processeur. Ils représentent ainsi une source d'information intéressante dans le cadre de la sécurité, pour vérifier le bon comportement du processeur vis-à-vis des tâches qu'il doit exécuter. Ainsi, nous pré-

sentons dans la prochaine section différents travaux utilisant ces compteurs dans le cadre de la sécurité.

4.1.2 Les compteurs matériels pour la sécurité

HPCMalhunter [Bahador 2014] est une solution qui utilise de manière supervisée les HPC, la *singular value decomposition* (SVD) et le classificateur SVM afin de créer un modèle de comportement du processeur vis-à-vis de l'exécution d'un logiciel. Afin de représenter le comportement du processeur, quatre évènements sont récupérés toutes les 100 000 instructions sous forme de vecteurs composés des quatre compteurs. Ces vecteurs sont concaténés sous la forme d'une matrice pour former une trace de l'exécution. Les auteurs utilisent la décomposition SVD pour extraire de cette matrice un vecteur de caractéristiques plus petit tout en conservant une grande partie de l'information initiale. La solution est ainsi décomposée en une phase d'entraînement et une phase de détection. Les expérimentations de leur approche ont été menées sur une machine virtuelle et un processeur Intel. Vingt programmes légitimes basiques sont surveillés, comme cp, cat, ls et les attaques utilisées sont elles aussi assez classiques, comme des chevaux de Troie ou des portes dérobées. En comparant SVM à d'autres algorithmes de classification, les auteurs concluent que SVM offre le meilleur compromis alliant bonne détection des attaques et peu de fausses alarmes.

Une autre solution propose d'utiliser les HPC afin de détecter des maliciels [Demme 2013]. Ils étudient ainsi plusieurs cas d'usage : le cas d'un maliciel sur Android, le cas d'un *rootkit* sur Linux, et le cas d'une attaque par canaux auxiliaires. Les évènements stockés dans les HPC sont soit utilisés de manière brute, soit agrégés en fonction des changements de contexte du processeur. Les algorithmes de classifications utilisés sont k-nn, une forêt aléatoire, une forêt d'arbres de décision et un réseau de neurones. Les données utilisées sont cette fois-ci plus nombreuses : plus de 500 maliciels et plus de 200 logiciels bénins sont utilisés et testés. Une interruption est générée en fonction du nombre de cycles écoulés, afin de récupérer l'état des compteurs. Les auteurs concluent que l'utilisation des compteurs pour détecter des maliciels donne de bons résultats mais que la solution proposée mérite de plus amples recherches afin d'optimiser cette détection.

Les auteurs de HPCMalhunter publient plus tard une solution appelée HLMD [Bahador 2019] utilisant cette fois-ci un système de signatures. Ils utilisent toujours la SVD pour réduire la taille des données mesurées mais séparent l'analyse qui suit en deux étapes. La première étape consiste à choisir la catégorie à laquelle l'exécution du logiciel s'apparente, c'est-à-dire bénin ou un des types d'attaques connues. Puis la seconde étape consiste à confronter les signatures de cette catégorie avec les caractéristiques récupérées dès le début d'exécution du programme surveillé. Pour créer les signatures, les auteurs utilisent le même type de données que dans leurs recherches précédentes (pour les programmes et les attaques classiques), mais en plus grand nombre toutefois. Les résultats montrent que cette solution est une amélioration de leur précédente proposition, et indiquent aussi, selon les auteurs,

qu'elle offre de meilleurs résultats que la proposition de [Demme 2013].

Certains travaux se concentrent sur la détection de certaines attaques spécifiques. Par exemple [Prada 2019] cible une attaque utilisant le cache contre l'algorithme de chiffrement symétrique AES. Le principe est d'utiliser le cache partagé entre plusieurs coeurs pour extraire des informations secrètes comme les clés de chiffrement. Cette attaque, sollicitant beaucoup les caches du processeur, va ainsi générer une utilisation "anormale" de ces caches, que l'on peut identifier en consultant les valeurs des certains HPC. Les auteurs utilisent le nombre de *cache-miss* du cache L3 ainsi que le nombre de d'instructions de chargement afin de confronter les deux métriques. Cependant, bien qu'il soit possible de constater une déviation significative du comportement légitime, aucune solution précise d'algorithme de détection n'est décrite.

Les travaux de [Tang 2014] présentent l'utilisation d'une variante de SVM appelée One Class SVM (OCSVM) qui se base uniquement sur les données d'une seule classe. Ainsi, le comportement normal est modélisé via OCSVM et toute déviation significative par rapport aux données légitimes lève une alerte. Il s'agit donc d'un apprentissage semi-supervisé. Il est important de noter que bien que le titre de l'article soit "Unsupervised Anomaly-based Malware Detection using Hardware Features", les auteurs ne font pas d'apprentissage non-supervisé au sens où nous l'avons défini plus tôt car il y a une phase d'apprentissage préalable. Le modèle d'attaque étudié repose sur une attaque ROP (*Return Oriented Programming*) suivie de l'exploitation de la vulnérabilité, le but étant de détecter l'attaque le plus tôt possible. Afin de choisir les compteurs les plus adaptés, une étude est faite sur ceux qui permettent le mieux de distinguer les attaques des comportements légitimes. La lecture des compteurs se fait à chaque *epoch* de temps, c'est-à-dire à la granularité la plus fine possible, afin de détecter le plus tôt possible un comportement pouvant correspondre à une attaque. Deux types d'extraction de caractéristiques sont faites. La première effectue une analyse directe de chaque trace brute une à une, sans prendre en considération l'aspect temporel et l'historique. La deuxième regroupe plusieurs traces consécutives afin de créer un vecteur de caractéristiques plus grand, qui prend ainsi plus en considération l'aspect temporel. Les expérimentations sont ensuite réalisées sur une machine virtuelle avec des programmes souvent utilisés comme vecteurs d'attaques, à savoir *Adobe PDF Reader* et *Internet Explorer*. Les auteurs confrontent ainsi différents paramètres de la détection : évènements architecturaux et micro-architecturaux, analyse temporelle ou non-temporelle, granularité de récupération de l'état des compteurs, étape de détection du maliciel (détection en début ou fin de la phase d'attaque).

Certaines solutions utilisent les HPCs dans le cadre de systèmes embarqués, ce qui correspond plus au contexte présenté dans cette thèse. Par exemple, [Wang 2015] présente l'utilisation des compteurs dans le cas d'un firmware. La solution proposée consiste à modifier le *boot loader* en mémoire morte (ROM) et à y insérer un code spécifique. Cette solution se propose d'insérer des points de contrôle dans le code du logiciel à surveiller afin de récupérer l'évolution à des points précis de l'exécution

du code. Cette insertion ne nécessite pas de connaissances préalables sur le firmware exécuté, ce qui représente un avantage pour le déploiement. Afin d'éviter que les attaquants puissent prévoir les différents points de contrôle et potentiellement détourner le système, les points de contrôle sont insérés à des endroits stratégiques et à des endroits aléatoires. La solution contient une phase d'entraînement et une phase de détection. Lors de la phase d'entraînement, l'évolution des compteurs est enregistrée en fonction des points de contrôle. Lors de la phase de détection, un certain nombre de points de contrôle aléatoires sont choisis et peuvent être changés à chaque redémarrage du sytème ou lorsque le code de la solution prend la main. L'évolution des compteurs en fonction des points de contrôles insérés dans le code est ensuite comparée à l'évolution (la signature) obtenue lors de la phase d'apprentissage. Les auteurs montrent ainsi une solution légère testée sur des processeurs ARM et PowerPC.

[Krishnamurthy 2019] propose une solution pour les systèmes cyber-physiques et plus particulièrement pour surveiller les automates programmables industriels (PLC) brièvement introduits dans le chapitre 1. Le but de cette étude est de prouver l'efficacité des HPC pour détecter des attaques dans ce contexte, afin de favoriser le développement d'un support matériel spécifique permettant la surveillance continue dans les processeurs de futures générations. La particularité de ces travaux est le contexte industriel et l'utilisation en temps réel de processus composés de plusieurs threads. L'approche utilise un modèle OCSVM pour modéliser le comportement des PLCs. Les HPCs sont récupérés régulièrement par une machine distante en fonction du temps écoulé. C'est la machine distante, qui a plus de capacités de calcul que le PLCs, qui analyse le comportement actuel du processeur du PLC pour le confronter au modèle appris grâce à OCSVM. La récupération des compteurs se fait en parallèle de l'exécution du programme, ce qui permet de ne pas avoir à connaître ni à modifier le programme pour utiliser la solution. Chaque *thread* est également inspecté séparément, permettant une analyse approfondie du comportement du processeur. Les attaques considérées sont la modification des paramètres du PLC ou la modification du programme. Les auteurs analysent l'évolution des compteurs sous forme de séries temporelles. Les caractéristiques analysées à partir de ces séries temporelles sont nombreuses : des statistiques descriptives générales aux méthodes polynomiales en passant par des analyses temporelles comme les transformées de Fourier mais aussi les corrélations et auto-corrélations des différents évènements. Les auteurs montrent ainsi une excellente détection, en testant leur solution sur deux PLCs différents.

Au final, on peut citer un bon nombre de travaux de recherche utilisant les compteurs pour la sécurité, mais leur principe général est toujours globalement le même : il consiste à sauvegarder l'évolution des compteurs lors d'exécutions d'un logiciel légitime de façon à obtenir un modèle du comportement légitime et comparer ensuite en temps réel l'évolution de ces mêmes compteurs vis-a-vis de ce modèle. Notre contexte nous rapproche des dernières solutions présentées. En effet les premières approches ont été développées soit pour des systèmes qui ne correspondent

pas aux objets connectés étudiés dans cette thèse, soit dans le cadre de la détection d'une attaque en particulier. De plus, la plupart des solutions ont une phase d'entraînement et une phase de détection. Nous voulons dans notre solution faciliter le plus possible le déploiement et la maintenance de notre solution (objectif **O3**), et ainsi éviter d'avoir à créer un modèle complexe du comportement du logiciel qui s'exécute sur l'objet. En ce sens, la solution que nous proposons ici se démarque des approches précédentes.

Par ailleurs, l'utilisation des HPCs pour la sécurité ne fait pas l'unanimité, et certains travaux soulignent des limitations ou des biais sur la manière dont certaines recherches ont été faites. Nous présentons ainsi dans la suite de cette section ces critiques que nous prendrons en compte dans la suite de notre travail.

4.1.3 Les critiques sur l'utilisation des compteurs

[Weaver 2013b] étudie l'exactitude, le déterminisme et les limites des valeurs données par les HPCs. Leur recherche est focalisée sur les processeurs Intel, qui sont, d'après leurs recherches, les plus concernés par des problèmes de déterminisme. Les auteurs soulignent par exemple que certains évènements peuvent être affectés par des interruptions matérielles, qui sont imprévisibles. De plus ils présentent également certains évènements qui seraient mal comptés, comme des instructions comptées plusieurs fois ou pas comptées. Les auteurs concluent que les compteurs ne sont pas tous fiables et que certaines déviations peuvent se produire, mais qu'il est possible d'améliorer le système de comptage pour avoir plus d'évènements déterministes.

Les travaux de [Zhou 2018] critiquent les travaux qui s'appuient sur les HPCs pour la sécurité. Ils critiquent plus particulièrement l'hypothèse que les programmes légitimes et les programmes malveillants aient des impacts différents sur l'évolution des HPCs. Ils soulignent également certaines faiblesses dans les hypothèses et les expérimentations menées pour la détection de maliciel en utilisant les HPCs. Ils mettent en question notamment l'utilisation de machines virtuelles afin d'étudier les HPCs. En effet, virtualiser les HPCs n'est pas une tâche aisée et les auteurs affirment que l'évolution des HPCs sur machine virtuelle est peu corrélée à l'évolution des HPCs sur machine réelle. Ils mettent en avant également un manque de données et le manque de validation croisée, une méthode permettant d'évaluer la fiabilité d'une approche dans l'apprentissage automatique. Enfin ils précisent que les attaques permettant d'entraîner les algorithmes d'apprentissage automatique ne peuvent pas être exhaustives, ce qui peut provoquer des imprécisions de classifications pour des attaques qui ne seraient pas considérées dans les jeux de données utilisés. Ils ont ainsi créé un jeu de données de 1000 logiciels bénins et 1000 maliciels, qu'ils ont utilisé afin d'entraîner des algorithmes d'apprentissage automatique et faire de la détection. Les différents algorithmes testés sont k-nn, des réseaux de neurones, Adaboost, forêt aléatoire, arbres de décision, classificateur bayésien naïf. Les résultats obtenus n'ont pas été satisfaisants. Ils ont donc conclu qu'il n'est pas possible de détecter efficacement toute sorte de maliciel, un

peu à la manière d'un anti-virus classique, en utilisant les HPCs. Ils ajoutent que les hypothèses des recherches cherchant à détecter des maliciels se basent sur des hypothèses trop optimistes ou des expérimentations qui ne sont pas suffisamment réalistes. Cependant, leur vision concerne la détection de maliciel dans un cas généraliste, où les programmes bénins et malveillants sont considérés en même temps dans l'apprentissage automatique. De plus, les appareils considérés exécutent des systèmes d'exploitation volumineux (Windows notamment) et correspondent à des ordinateurs plutôt qu'à des objets connectés que l'on souhaite considérer dans notre contexte.

[Das 2019] abordent également certaines problématiques de l'utilisation des compteurs dans le cas particulier de la sécurité. Les auteurs affirment que les HPCs sont souvent mal utilisés et que les problèmes de déterminisme et de *overcounting* devraient décourager les solutions utilisant des valeurs précises des compteurs. De plus ils soulèvent les problématiques suivantes :

- Les compteurs sont souvent utilisés pour surveiller le comportement d'un programme. Cependant les changements de contexte du processeur peuvent perturber le relevé des compteurs, il faut donc bien s'assurer de réinitialiser les compteurs après chaque changement de contexte.
- Les solutions qui fonctionnent pour un processeur ne fonctionnent pas forcément sur un autre. Les différents évènements que l'on peut récupérer d'un processeur à un autre ne sont pas forcément les mêmes. De plus, l'implémentation des compteurs peut elle aussi être spécifique. Cela réduit la portabilité des solutions, particulièrement lorsque les solutions étudient précisément quels sont les meilleurs compteurs permettant de distinguer les logiciels légitimes des malveillants.
- Les autres programmes et notamment le système d'exploitation peuvent avoir un impact non-négligeable sur les compteurs. Deux exécutions avec un environnement d'exécution différent vont correspondre à deux évolutions différentes des HPCs.

Malgré ces différentes critiques concernant l'utilisation des compteurs matériels de performance, nous pensons qu'il est toujours possible de les utiliser dans le cadre de la sécurité, moyennant certaines conditions :

— Les objets et logiciels étudiés doivent être suffisamment simples. Il est important que les compteurs reflètent au mieux l'exécution du logiciel sur le processeur. Plus le système est complexe et moins le parallèle est évident. Il faut donc privilégier les environnements simples et bien maîtrisés. Il est préférable d'éviter l'utilisation de ces compteurs dans le cadre de la sécurité classique d'ordinateurs de bureau.

— Lorsque l'on compare les compteurs de différents "objets", il faut également vérifier que les environnements logiciels et matériels de ces objets soient identiques.

Dans notre situation, nos hypothèses **H1** et **H3** stipulent que les objets sont

identiques au niveau du matériel et logiciel, et qu'ils ont, de part leurs ressources limitées, un comportement simple et déterministe. Nous pensons que dans ce contexte précis, les compteurs de performances sont utilisables. Nous allons ainsi chercher dans notre approche à définir un comportement global de l'objet et détecter les déviations significatives via les compteurs matériels de performance. Cette approche est décrite dans la section suivante.

4.2 Une approche mêlant HPC et détection d'anomalies

Notre approche utilise les compteurs matériels de performance car nous pensons qu'ils peuvent refléter le comportement des objets. Elle consiste donc à collecter des informations significatives mais simples sur les objets, afin de les utiliser de manière centralisée pour réaliser de la détection d'intrusion. Nous présentons tout d'abord une vue d'ensemble de notre approche avant de la détailler en présentant chronologiquement ses principales étapes. Ces différentes étapes ont été conçues dans le respect des hypothèses et des objectifs qui ont été présentés dans la section 1.2.3.3.

4.2.1 Vue d'ensemble

Dans un premier temps, rappelons le contexte dans lequel se situe notre approche. Nous envisageons le déploiement massif d'objets connectés identiques, à travers une large zone géographique comme celle d'une région ou d'un pays par exemple. Ces objets sont tous connectés à des serveurs centraux, qui sont utilisés pour le contrôle à distance ou la maintenance (mise à jour). Ces objets ont des capacités limitées en terme de calcul ou de mémoire. Différentes approches ont déjà été considérées et étudiées, mais aucune suffisamment satisfaisante au regard de nos hypothèses et objectifs. Conformément à nos objectifs, nous proposons une solution légère, sans modification matérielle, simple à déployer et à maintenir. Conformément à nos hypothèses de menace, il s'agit principalement de détecter tout abus d'utilisation des ressources de l'objet (CPU, mémoire, réseau), en considérant les communications comme sécurisées et le déni de service comme détecté par défaut puisque nous considérons qu'il existe déjà des mécanismes de surveillance qui permettent de détecter qu'un objet ne rend plus du tout son service.

Nous avons donc choisi d'utiliser les compteurs matériels de performance comme source d'information sur les appareils déployés. Ils permettent de récupérer des informations significatives suffisamment simplement. Ces informations sont envoyées aux serveurs centraux sans aucune modification. Ainsi les objets ont peu de ressources à mobiliser dans le cadre de notre solution de sécurité. C'est le serveur qui est en charge d'analyser les informations de tous les objets. Concernant l'analyse, nous voulons éviter de construire un modèle complexe du comportement du logiciel des objets basé sur l'évolution des HPCs. En effet, la création d'un tel modèle de

comportement peut se révéler complexe à déployer et à maintenir dans le cas de mises à jour. Nous prenons ainsi le comportement de la majorité comme comportement légitime et utilisons des algorithmes de détection d'anomalies en temps réel. Cela est conforme avec nos hypothèses de départ dans lesquelles nous considérons que chaque comportement légitime des objets massivement déployés est largement représenté et que tous les objets ne peuvent pas être corrompus simultanément.

Comme illustré dans le schéma 4.1, notre approche est constituée de 5 grandes étapes :

- Sauvegarde de l'évolution des compteurs : cette sauvegarde est effectuée de manière régulière sur l'objet lui-même (nous supposons qu'il dispose d'un système de stockage local).
- Collecte régulière des fichiers de compteurs des différents objets déployés : chaque objet envoie régulièrement aux serveurs centraux les valeurs de ses compteurs depuis la dernière remontée d'information pour qu'ils puissent effectuer une analyse.
- Extraction des caractéristiques : les données brutes reçues par les serveurs sont transformées en caractéristiques représentatives pour des algorithmes de détection d'anomalies.
- Mise en forme : les caractéristiques sont mises en forme via une standardisation afin de les mettre à la même échelle pour la détection d'anomalies.
- Détection de valeurs aberrantes : les différentes données issues de la mise en forme sont fournies en entrée d'un algorithme de détection de valeurs aberrantes en charge de lever une alerte en cas de détection de comportements jugés anormaux (déviant significativement des comportements de la majorité des objets).

Nous n'étudions pas dans ce document la nature de la réaction face à la détection. Il est cependant possible de penser à une réinstallation du programme original sur l'objet, d'une manière similaire à ce que propose une des solutions décrites dans le chapitre précédent (PoSE [Perito 2010]). Nous allons présenter plus en détails chacune de ces étapes dans les sous-sections suivantes.

4.2.2 Lecture

Lors de cette étape, plusieurs choix s'offrent à nous : il est possible d'utiliser un des outils permettant de récupérer les compteurs ou de réaliser un module noyau nous-mêmes. Afin de maîtriser et de réduire au plus l'impact de la solution sur les objets, nous choisissons de développer un module noyau qui nous permettra de récupérer les valeurs des compteurs.

Il nous faut ensuite aborder le choix de l'intervalle de collecte des compteurs. Il est possible de le baser sur le temps ou d'utiliser certains évènements comme le nombre de cycles ou d'instructions exécutées. Nous avons choisi dans notre cas de nous baser sur le temps. Les valeurs de différentes compteurs sont donc envoyées de façon périodique aux serveurs centraux.

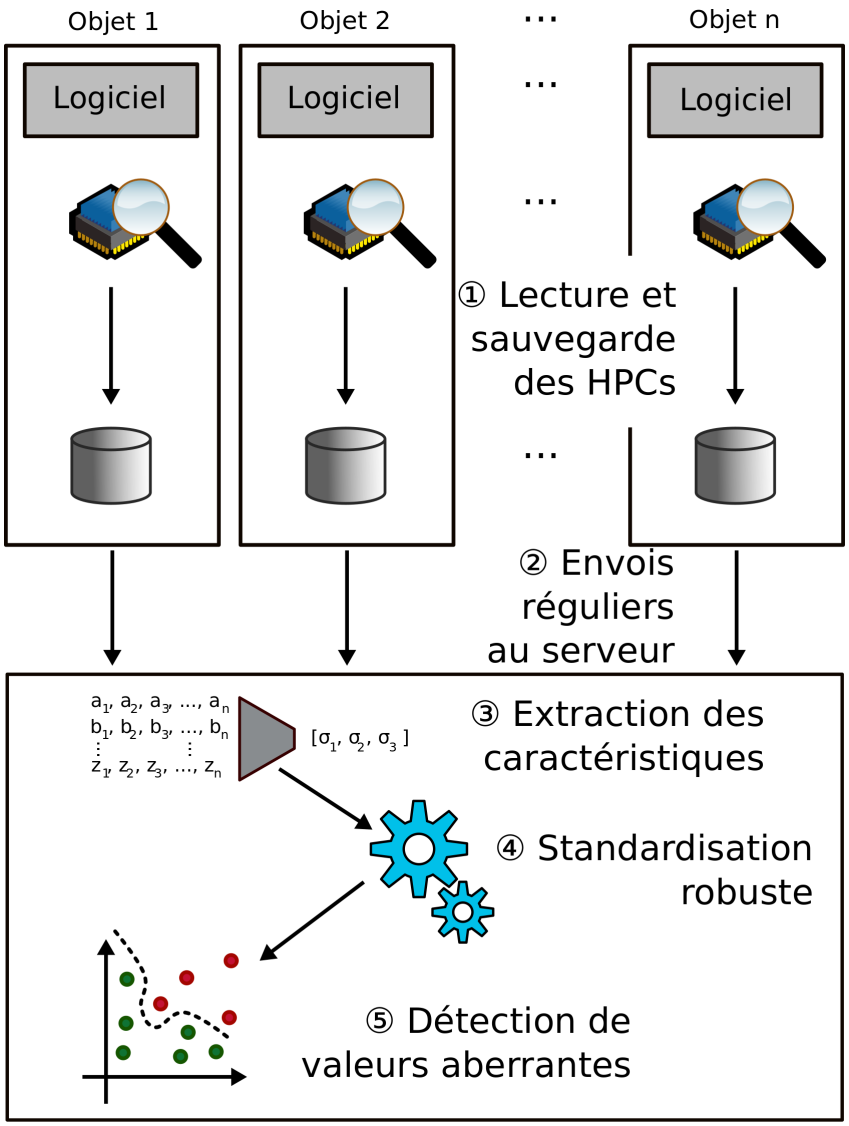


FIGURE 4.1 – Une vue d'ensemble de notre approche

Dans la suite D désigne l'ensemble des appareil déployés, C l'ensemble des compteurs utilisés, d désigne un appareil et c un compteur. Pour les séries temporelles ou un vecteur A, $A[i]$ désigne le i-ème élément. L'évolution des compteurs est sauvegardée régulièrement, toutes les Δ_p secondes, dans un fichier que l'on nomme dans la suite "trace" et noté T_d. $T_{c,d}$ désigne ainsi la série temporelle du compteur c et de l'appareil d. Les compteurs sont périodiquement collectés par le module noyau et stockés localement dans un fichier. Les objets exécutent des tâches simples et répétitives et notre objectif et de repérer les déviations de comportement. Nous pensons que les compteurs peuvent ainsi nous donner une idée de l'activité du processeur, et repérer une activité anormale sur l'objet. Ils ne sont pas remis à zéro à chaque lecture.

Concernant le code s'exécutant sur les objets surveillés permettant la récupération des valeurs des compteurs, nous faisons l'hypothèse qu'il est stocké dans une mémoire morte (ROM) et qu'il ne peut donc pas être modifié. Ainsi, notre solution n'implique pas de modification ou de connaissance supplémentaire sur le logiciel surveillé, au delà de l'observation des valeurs des HPCs. Cela facilite la phase de déploiement des objets.

4.2.3 Collecte

Les serveurs centraux s'occupent de collecter régulièrement, toutes les Δ_s secondes, les enregistrements de HPCs (appelés traces) de l'ensemble des objets D. Ce sont les objets eux-mêmes qui envoient périodiquement les traces T_d qui ont été générées et stockées localement, sans autre tâche d'analyse à faire. Toutes les analyses et transformations des données brutes se font du côté du serveur, afin de solliciter le moins possible les objets et rendre notre solution la plus légère possible du côté des objets. Comme nous faisons l'hypothèse que les communications sont sécurisées, nous considérons que les messages et donc les traces ne peuvent être lus par un attaquant sur le réseau. Le pas Δ_s entre chaque collecte doit être déterminé de façon à obtenir des traces représentatives du comportement global du processeur et du logiciel exécuté. Ce paramètre est également important pour éviter les problèmes d'engorgements des données et la rapidité de détection. En effet plus Δ_s est grand et plus les traces T_d sont lourdes et la détection lente. D'un autre côté, si Δ_s est trop petit, les séries temporelles $T_{c,d}$ risquent d'être trop courtes, et pas suffisamment représentatives du comportement de l'objet et ainsi difficiles à comparer. Il est donc nécessaire de fixer cette valeur de façon à avoir un bon compromis entre efficacité et rapidité de la détection.

4.2.4 Extraction des caractéristiques

Les registres stockant les compteurs font 32 ou 64 bits. Ainsi lorsqu'un compteur atteint sa valeur maximale, il recommence à zéro. Comme nous souhaitons comparer des milliers d'objets en même temps, comparer toutes les séries temporelles brutes serait trop complexe et coûteux en ressources mémoires et en calculs. De plus, la valeur absolue du compteur ne donne aucune information intéressante, car nous ne remettons pas la valeur à zéro à chaque lecture. Ainsi dans cette phase d'extraction des caractéristiques, une première étape consiste donc à transformer ces séries temporelles en valeurs qui sont plus facilement comparables. Cette étape consiste donc à étudier l'évolution des compteurs plutôt que leurs valeurs absolues. Nous calculons ainsi à partir des $T_{c,d}$ la dérivée que nous noterons $T'_{c,d}$.

Nous avons ensuite choisi d'étudier des caractéristiques descriptives globales des séries temporelles. Ces caractéristiques sont les statistiques descriptives suivantes : moyenne (mean), maximum (max), minimum (min), médiane (med), écart-type (std), premier et troisième quartiles (1Q et 3Q). Il est possible d'étudier plus de caractéristiques, comme décrit dans les travaux [Krishnamurthy 2019] présentés précé-

demment. Cependant cette étape est à dimensionner en conjonction avec les étapes suivantes (la détection d'anomalies) et le nombre d'objets, et donc de données, à analyser. Nous avons choisi cet ensemble de caractéristiques classiques pour avoir une représentation de la distribution de l'évolution des données, sans pour autant faire exploser le nombre de caractéristiques à calculer puis à comparer via les algorithmes de détection d'anomalies. L'ensemble des caractéristiques d'un compteur HPC c et d'un objet d est noté $F_{c,d}$. Nous obtenons ainsi :

$$F_{c,d} = (\max(T'_{c,d}), \text{mean}(T'_{c,d}), \min(T'_{c,d}), \text{std}(T'_{c,d}), \text{med}(T'_{c,d}), 1\text{Q}(T'_{c,d}), 3\text{Q}(T'_{c,d}))$$

Algorithme 1 : Extraction des caractéristiques des données par le serveur

for $d \in D$ **do**
  for $c \in C$ **do**
    for i **from** 1 **to** $|T_{c,d}| - 1$ **do**
      $T'_{c,d}[i] \leftarrow (T_{c,d}[i+1] - T_{c,d}[i])$
    $F_{c,d} \leftarrow$
      $(\max(T'_{c,d}), \text{mean}(T'_{c,d}), \min(T'_{c,d}), \text{std}(T'_{c,d}), \text{med}(T'_{c,d}), 1\text{Q}(T'_{c,d}), 3\text{Q}(T'_{c,d}))$

4.2.5 Standardisation

L'étape de standardisation est le dernier traitement que nous réalisons sur les caractéristiques $F_{c,d}$. Elle est nécessaire pour que l'algorithme de détection puisse s'exécuter de façon efficace. En effet, sans standardisation des problèmes d'échelles différentes pourraient conduire à un déséquilibre dans la prise en compte des caractéristiques par l'algorithme de détection (certaines caractéristiques pourraient influer plus que d'autres sur le résultat de la détection). On peut par exemple rencontrer des différences de l'ordre d'un facteur 100 concernant l'occurrence de certains évènements. Nous standardisons donc chacune des caractéristiques compte tenu de la population globale. La formule de standardisation classique est la suivante :

$$\frac{F_{c,d}[i] - \text{mean}(\{F_{c,e}[i] \mid e \in D\})}{\max(\{F_{c,e}[i] \mid e \in D\}) - \min(\{F_{c,e}[i] \mid e \in D\})}$$

Comme on peut le voir dans cette formule, la standardisation consiste à transformer chaque caractéristique via une transformation affine. Autrement dit, pour chaque caractéristique c, il faut trouver deux nombres m et v pour transformer la caractéristique brute c en caractéristique normalisée $\frac{c-m}{v}$. La standardisation présentée précédemment utilise la moyenne de c pour m et la variation maximale au sein de c pour v. Étant donné que ces deux valeurs ne sont pas robustes en cas d'anomalie (une valeur aberrante très élevée modifierait complètement la valeur de v par exemple), nous avons choisi d'utiliser une version robuste [Rousseeuw 1993]

qui s'appuie sur la médiane et un écart intercentile. Au final, le vecteur ainsi obtenu sera noté $V_{c,d}$ et respecte la formule suivante :

$$V_{c,d}[i] = \frac{F_{c,d}[i] - \text{median}(\{F_{c,e}[i] \mid e \in D\})}{\text{P}_{95}(\{F_{c,e}[i] \mid e \in D\}) - \text{P}_5(\{F_{c,e}[i] \mid e \in D\})}$$

Même si la mesure de la médiane est très robuste en présence d'anomalies, l'écart intercentile peut tout de même être influencé par les anomalies s'il y en a suffisamment. En effet, dans notre cas, nous nous appuyons sur le 5e centile (c'est-à-dire le seuil au-dessous duquel se placent 5% des valeurs de la caractéristique) et du 95e centile (c'est-à-dire le seuil au-dessus duquel se place 5% des valeurs de la caractéristique). Autrement dit, à partir de 5% d'anomalies dans une population, ces valeurs peuvent être perturbées par une attaque. Il serait bien sûr possible de prendre un autre écart intercentile (par exemple le 10e et le 90e centile), mais cela a tendance à rendre la standardisation moins fiable : les valeurs sont plus approximatives et la standardisation est de moins bonne qualité. Ceci peut affecter négativement les performances des détections de valeurs aberrantes.

L'algorithme 2 résume ainsi les étapes d'extraction des données et de standardisation robuste.

Algorithme 2 : Standardisation des données par le serveur

```
for c ∈ C do
    for i from 1 to |F_{c,d}| do
        m ← med({F_{c,e}[i] | e ∈ D})
        d ← P_95({F_{c,e}[i] | e ∈ D}) − P_5({F_{c,e}[i] | e ∈ D})
        for d ∈ D do
            V_{c,d}[i] ← (F_{c,d}[i] − m)/d
return apply outlier detection algorithm on V
```

4.2.6 Détection d'anomalies

La dernière étape de notre approche consiste à effectuer la détection à partir de nos données transformées. Comme nous l'avions précisé dans le chapitre 3, nous préférons une solution en temps réel, sans phase préalable d'apprentissage, afin de faciliter le déploiement de notre solution de sécurité. La dernière étape consiste ainsi à appliquer un algorithme de détection de valeurs aberrantes non supervisé directement sur nos séries temporelles standardisées. Nous n'avons ainsi pas de modèle complexe de comportement à apprendre et pas de phase d'apprentissage.

Notre approche est ainsi simple à déployer et à adapter à un système global qui pourrait changer dans le temps en fonction des mises à jour par exemple. L'efficacité de notre approche dépend cependant du fait que la majorité des objets (les objets légitimes) aient un comportement similaire ce qui est bien le cas dans le contexte spécifique de nos travaux, dans lequel les objets déployés sont en tout point identiques

et dans lequel nous considérons qu'un attaquant ne peut pas corrompre simultanément tous les objets. Comme l'efficacité des algorithmes de détection d'anomalie dépend des données analysées, nous avons expérimenté dans notre approche plusieurs familles d'algorithmes. Nous décrivons plus précisément dans la prochaine section ces différents algorithmes.

4.3 Choix des algorithmes de détection de valeurs aberrantes

Une composante importante de notre approche est la partie détection de valeurs aberrantes. Nous avons donc choisi dans nos expérimentations différents algorithmes. Nous présentons dans cette partie quelques principes de base de la détection de valeurs aberrantes avant de présenter, dans trois sous-sections, les trois grandes familles d'algorithmes que nous avons choisis de tester, ainsi que les algorithmes les plus utilisés par famille.

4.3.1 Généralités

La détection de valeurs aberrantes consiste à identifier toute donnée qui dévie significativement de la majorité des autres. Cette technique est utilisée dans différents domaines d'application comme le domaine médical, météorologique, la détection de fraudes, et la thématique qui est au coeur de ces travaux, la détection d'intrusion. Les valeurs aberrantes peuvent ainsi refléter différents types de phénomènes, comme des comportements anormaux résultant de potentielles malveillances, mais aussi des évènements atypiques permettant de détecter une maladie ou un changement environnemental significatif par exemple. La figure 4.2 illustre la présence d'anomalies dans des données légitimes.

Il existe deux types de sorties d'un algorithme de détection de valeurs aberrantes : binaire ou continue. Soit une donnée est labellisée comme "normale" ou "anormale" (binaire), soit un score d'anormalité lui est associé (continue). Dans le cas de la détection d'intrusion, il est ainsi possible soit de lever une alerte pour chaque donnée labellisée comme "anormale" dans le cas binaire, soit de construire différents niveaux d'alertes en fonction du score d'anormalité dans le cas continu.

Les algorithmes de détection de valeurs aberrantes sont utilisés dans un cadre non-supervisé. Néanmoins, ces algorithmes sont parfois des adaptations d'autres algorithmes qui sont utilisés en apprentissage supervisé. Nous ferons ainsi mention des algorithmes k-NN et OCSVM qui sont des algorithmes de classification supervisés qui ont été modifiés pour un contexte non-supervisé.

L'efficacité des différents algorithmes dépend de la distribution des données. Dans notre approche, nous testons plusieurs algorithmes pour déterminer ceux qui semblent les plus adaptés étant donné la nature des données de nos expérimentations. Cependant, comme il existe une myriade de variantes d'algorithmes différents, nous expérimentons seulement ceux qui nous semblent les plus pertinents, qui sont

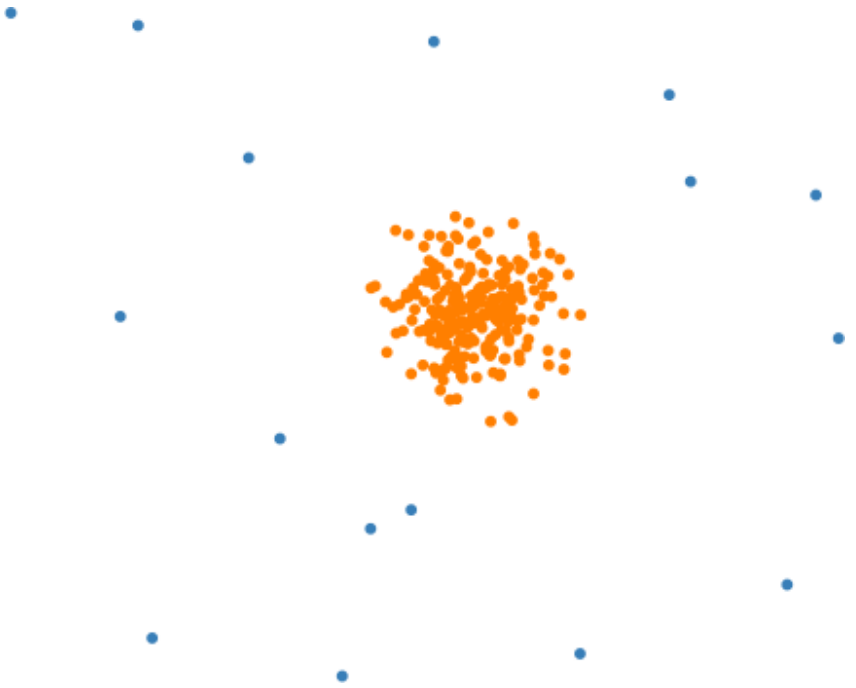


FIGURE 4.2 – Un exemple de mesures avec des valeurs légitimes (en orange, au centre) et des valeurs aberrantes (en bleu, tout autour)

alors paramétrés selon les recommandations usuelles. Nous présentons dans la suite trois différentes familles d'algorithmes : les algorithmes basés sur les plus proches voisins, les algorithmes basés sur de la classification et les algorithmes basés sur du regroupement.

4.3.2 Algorithmes basés sur les plus proches voisins

Les algorithmes basés sur les plus proches voisins utilisent des métriques liées à un point et à ses plus proches voisins.

k-NN (k Nearest Neighbors) [Ramaswamy 2000] est un algorithme de classification qui utilise les plus proches voisins (au sens d'une métrique comme la distance euclidienne) pour prédire la classe d'un nouveau point. Cet algorithme a été adapté à la détection de valeurs aberrantes en deux variantes : MeanDIST et KDIST. L'idée sous-jacente à ces deux variantes est qu'une valeur aberrante est éloignée de son voisinage et que donc sa distance aux plus proches voisins est élevée. On le voit très bien sur la figure 4.2 : la distance d'un point orange à ses voisins est plus faible que la distance d'un point bleu à ses voisins. La taille de ce voisinage, notée k, est un paramètre crucial. En effet si celui-ci est trop petit, alors un groupement de k anomalies pourrait être considéré comme des données normales car proche les unes des autres. Si ce paramètre est trop grand, alors le voisinage considéré pourrait être si grand qu'il inclurait des points non-pertinents. Ce paramètre impacte également la complexité d'exécution de l'algorithme.

Il existe deux variations de cette adaptation de k-NN à la détection de valeurs aberrantes : MeanDIST et KDIST qui reposent respectivement sur la moyenne des distances du k-voisinage et sur le maximum des distances du k-voisinage. Plus exactement, en notant $N(i)$ le i^{e} voisin plus proche de N et $d(N_1, N_2)$ la distance entre deux points N_1 et N_2 (typiquement la distance euclidienne), l'algorithme MeanDIST associe à un point N le score :

$$MeanDIST_k(N) = \frac{1}{k}\sum_{i=1}^{k} d(N, N(i))$$

et l'algorithme KDist associe à un point N le score :

$$KDIST_k(N) = d(N, N(k))$$

L'algorithme LOF (Local Outlier Factor) [Breunig 2000] utilise également le principe des plus proches voisins mais s'appuie sur une notion de la densité locale pour attribuer un score d'anormalité. Le principe est donc de comparer la densité locale d'un point à la densité locale de ses k plus proches voisins. Dans l'exemple de la figure 4.2, on peut par exemple constater que la densité autour des points bleus est plus faible que la densité autour des points oranges.

Les formules que nous allons présenter peuvent être rencontrées dans une forme légèrement plus complexe dans la littérature (notamment [Breunig 2000]) pour prendre en compte le fait qu'il peut y avoir des égalités de distance lors de la constitution du voisinage. Afin de simplifier leur explication, nous avons décidé de retirer ces cas particuliers.

LOF s'appuie sur une distance lissée afin d'éviter d'obtenir des densités très grandes quand deux points sont très proches. Plus précisément, cet algorithme définit la distance d'accessibilité d_k (*reachability distance* en anglais) définie de la manière suivante :

$$d_k(N_1, N_2) = \max(KDIST_k(N_2), d(N_1, N_2))$$

À partir de cette distance d'accessibilité est définie la densité locale $l_k(N)$ comme étant l'inverse des distances d'accessibilité moyenne des plus proches voisins de N :

$$l_k(N) = \left(\frac{\sum_{i=1}^{n} d_k(N, N(k))}{k}\right)^{-1}$$

Le score final est la moyenne des rapports entre la densité du voisinage sur la densité locale :

$$LOF_k(N) = \frac{1}{k}\sum_{i=1}^{k} \frac{l_k(N(i))}{l_k(N)}$$

Ainsi, plus LOF a une valeur éloignée de 1, plus la densité locale d'un point diffère de celle de son voisinage. Notamment, si cette valeur est particulièrement

élevée, cela signifie que le point considéré est dans une région peu dense, et est donc probablement une valeur aberrante.

ODIN (Outlier Detection using Indegree Number) [Hautamaki 2004] est un algorithme qui utilise le principe de plus proches voisins mais avec une stratégie différente. Ainsi un point n'est pas jugé en fonction de ses plus proches voisins mais en fonction du nombre de points dont il fait partie des k plus proches voisins. Un seuil est défini sur le nombre minimum de points dont il faut être le plus proche voisin afin d'être considéré comme "normal".

4.3.3 Algorithmes basés sur la classification

Dans le cadre de nos travaux, nous avons également testé deux algorithmes qui sont des adaptations d'algorithmes utilisés en classification. Les *Isolation Forest* sont une variante des *Random Forest* et l'utilisation des OCSVM dans ce contexte non-supervisé consiste à utiliser OCSVM comme dans un contexte semi-supervisé, en apprenant l'hypothèse que toutes les données sont légitimes.

Isolation Forest [Liu 2008] est un algorithme utilisant des arbres de décision aléatoires afin de déterminer si une donnée est isolée des autres. En effet, une hypothèse importante soulignée par les auteurs est que les données atypiques sont en général peu nombreuses et avec des valeurs significativement différentes des autres données normales. Ces valeurs atypiques ou aberrantes sont donc a priori plus faciles à isoler que les valeurs normales.

Les arbres de décision aléatoires divisent les données aléatoirement en prenant une valeur frontière aléatoire sur une dimension aléatoire permettant de séparer les données en deux groupes distincts. Par exemple, si on imagine un ensemble de points dans le plan (X,Y), on va d'abord choisir (aléatoirement) une dimension (X par exemple) ainsi qu'une valeur s comprise entre X_{min} et X_{max} et découper les points en 2 ensembles : les points dont les abscisses sont comprises entre X_{min} et s et les points dont les abscisses sont comprises entre s et X_{max}. Ce processus est ensuite répété itérativement jusqu'à ce que chaque point se retrouve seul dans un ensemble et donc isolé (d'où la terminologie *Isolation Forest*). L'arbre obtenu représente l'ensemble des séparations qui ont été effectuées. Comme le processus est aléatoire, la forêt aléatoire est construite de nombreuses fois (au moins une centaine), et la moyenne du nombre de séparations afin d'isoler chaque donnée est utilisée afin de donner un score d'isolation ou d'anormalité de chaque donnée entre 0 et 1. Un score proche de 1 correspond à une donnée atypique et un score inférieur à 0,5 est une donnée normale. Ce score reflète en réalité la profondeur de l'arbre en chaque point, l'idée étant que si la profondeur est faible, l'isolation a été rapide et donc qu'il s'agit d'une valeur atypique. Si aucune donnée n'est anormale, les scores auront une valeur avoisinant 0,5.

Les OCSVM (One-Class Support Vector Machine) [Amer 2013] sont une variante des classifieurs SVM (Support Vector Machine). Un SVM est un modèle qui sépare deux classes (logiciel bénin ou malveillant, personne malade ou saine, etc.) en séparant l'espace des caractéristiques avec un hyperplan (un ligne dans le cas du

plan, un plan dans le cas de l'espace, etc.). Étant donné que les classes sont rarement séparables par un hyperplan (que se passe-t-il si une classe entoure une autre par exemple ?), les SVM utilisent le *kernel trick* (astuce du noyau). Il s'agit d'une astuce mathématique qui modifie légèrement les calculs du SVM et qui permettent d'obtenir des formes beaucoup plus complexes. Ces formes dépendent du noyau utilisé ; les plus utilisés sont les noyaux linéaires et les noyaux gaussiens. Les OCSVM sont une modification semi-supervisée des SVM : ils peuvent être appris avec des exemples d'une seule classe. Au lieu d'apprendre un hyperplan pour séparer deux classes, les OCSVM apprennent une boule qui englobent les données d'apprentissage. Via la même astuce du noyau, la forme apprise peut être plus complexe qu'une simple boule.

Les algorithmes OCSVM sont conçus pour la classification semi-supervisée et ont donc besoin d'une phase d'apprentissage. Étant donné que nous n'avons pas de phase d'apprentissage dans notre détecteur, nous proposons d'utiliser les OCSVM de manière atypique : l'apprentissage et la détection de valeurs aberrantes se font sur les mêmes données. Il y a, techniquement parlant, une phase d'apprentissage du modèle, mais ce modèle est à usage unique : dès que de nouvelles données parviennent au serveur, un modèle est appris, est utilisé pour détecter les anomalies, et est ensuite effacé.

4.3.4 Algorithmes basés sur du regroupement

Le principe des algorithmes de détection de valeurs aberrantes basés sur le regroupement est de créer des groupes adjacents avec les données dont on dispose tout en laissant potentiellement des points hors des groupes. Ainsi toute donnée située hors des groupes ainsi créés est considérée comme une anomalie.

Une des méthodes de regroupement de données classique est nommé "k-moyennes" (k-means en anglais). L'idée est de regrouper les données en k boules, en minimisant la distance entre les points à l'intérieur de chacune des boules tout en maximisant la distance entre les points de boules différentes. Comme nous considérons que les objets que nous observons sont homogènes, nous prenons le cas particulier de k égal à 1. Nous fixons le rayon de cette boule afin de déterminer si un point est une anomalie ou non.

DBSCAN est un algorithme de regroupement basé sur la densité des points. DBSCAN détermine des groupes de manière itérative en procédant point par point. Si, pour un point N, il y a au moins nb_min points dans un rayon de ε autour de N, alors tous ces points appartiennent au même cluster. La recherche se poursuit jusqu'à ce que tous les points soient traités. Le nombre de groupes dépend donc fortement de ces deux paramètres nb_min et ε. Par exemple, si ε est trop grand, on risque de ne générer qu'un seul gros groupe ; s'il est trop petit, on risque de générer trop de petits groupes, voire d'en rater certains. Il est difficile de trouver la bonne combinaison de ces deux paramètres car ils s'influencent mutuellement : on ne peut pas simplement optimiser l'un puis l'autre. De plus, les points qui n'ont pas nb_min points dans un rayon de ε sont considérés comme des valeurs aberrantes.

HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) [McInnes 2017b] [McInnes 2017a] est un algorithme proche de DBSCAN qui ne nécessite plus le choix que d'un seul paramètre. Le fonctionnement de HDBSCAN est significativement plus complexe que DBSCAN et s'appuie sur une distance d'accessibilité comme LOF, puis procède à la recherche d'un arbre couvrant de poids minimum pour extraire la structure sous-jacente aux points avant d'utiliser du clustering hiérarchique pour trouver les frontières des clusters. Au final, cet algorithme a des performances similaires à DBSCAN mais est bien plus facile à paramétrer car il y a un seul paramètre principal : la taille minimale des clusters.

Tous les algorithmes présentés dans cette section ont été testés dans nos travaux, dans différentes expérimentations qui sont décrites dans le chapitre suivant. Grâce aux résultats obtenus, nous indiquerons les algorithmes qui nous semblent les plus intéressants dans notre contexte. Cependant l'efficacité de ces algorithmes peut varier en fonction de la nature des données. Ainsi, il peut être pertinent de faire une première analyse préliminaire avant de choisir définitivement l'algorithme à utiliser.

4.4 Conclusion

Dans cette partie, nous avons présenté les deux éléments principaux qui forment notre approche : les compteurs matériels de performance et la détection d'anomalie. Pour les compteurs de performances matériels, nous avons décrit les différentes utilisations de ces compteurs dans les travaux de recherche récents pour la sécurité. Après avoir exposé ensuite certaines critiques à l'égard de ces travaux, nous avons précisé les hypothèses et les conditions dans lesquelles la méthodologie est selon nous pertinente. Puis nous avons présenté notre approche de détection d'anomalie basée sur les compteurs matériels de performances pour des objets identiques, simples et massivement déployés. Nous avons pour cela décrit les principales étapes de notre approche : collecte de compteurs, extraction des caractéristiques, standardisation et détection de valeurs aberrantes. Dans cette approche, les objets transmettent les valeurs des compteurs à un serveur centralisé qui s'occupe d'exécuter les algorithmes de détection. Enfin, concernant la détection de valeurs aberrantes elle-même, nous avons présenté les principes de base de cette technique, les grandes familles d'algorithmes les plus utilisés ainsi que ceux que nous avons au final retenus dans notre travail. La prochaine section décrit les expérimentations que nous avons réalisées pour valider notre approche et comparé les différents algorithmes, ainsi que les résultats associés.

Chapitre 5

Expérimentations

Sommaire

5.1 Mise en place et validation de la plateforme 83
5.1.1 Environnement matériel et logiciel 84
5.1.2 Tests préliminaires 85
5.1.3 Preuve de concept 87
5.1.4 Amélioration de l'approche et des expérimentations 92
5.2 Mise à l'échelle et améliorations des expérimentations . . . 95
5.2.1 Nouvelle plateforme d'expérimentations 95
5.2.2 Analyses et résultats 98
5.2.3 Conclusion et discussions 106
5.3 Conclusion 108

Nous présentons dans ce chapitre les différentes expérimentations que nous avons menées afin de valider notre approche. Pour ces expérimentations nous avons mis en place deux plateformes de test différentes. La première, de petite envergure, comprenant 10 objets connectés, nous a permis de valider certaines hypothèses et de réaliser nos premiers tests de détection. La deuxième, plus élaborée, comprend cette fois-ci 100 objets et nous a permis de faire des expérimentations avec un volume de données plus conforme à nos hypothèses. Nous avons fait le choix de présenter chronologiquement nos expérimentations sur ces deux plateformes afin de montrer comment nous avons enrichi au fur et à mesure notre méthodologie, en fonction des retours de ces expérimentations. Nous allons ainsi décrire dans un premier temps les expérimentations utilisant la première plateforme, nous menant jusqu'à une preuve de concept élaborée. Nous présentons dans un second temps l'amélioration de notre plateforme pour la mise à l'échelle de nos expérimentations et grâce à laquelle nous avons pu mener deux expérimentations majeures permettant de valider notre approche.

5.1 Mise en place et validation de la plateforme

Dans cette section, nous présentons les expérimentations préliminaires qui nous ont permis de développer notre approche et de réaliser une première preuve de concept.

5.1.1 Environnement matériel et logiciel

Pour nos expérimentations, nous n'avions pas à notre disposition des objets connectés tels que réellement utilisés et déployés par EDF et il nous a fallu donc choisir des objets suffisamment représentatifs. Nous avons choisi d'utiliser des Raspberry Pi 3B+, équipés de processeurs ARM, qui sont souvent utilisés afin de réaliser des prototypes d'objets connectés. La version 3B+ possède ainsi un processeur ARM A53 (ARMv8). Concernant le système d'exploitation, nous avons simplement choisi d'utiliser Raspberry Pi OS. Pour vérifier que différents processeurs du même modèle ont des comportements similaires en ce qui concerne leurs compteurs, nous avons choisi dans nos premières expérimentations d'en utiliser 10 afin de générer nos différentes traces.

Il est nécessaire de collecter les valeurs des compteurs matériels de performance. Pour cela, comme expliqué dans le chapitre précédent, nous avons choisi de développer notre propre code, pour obtenir un meilleur contrôle et une meilleure compréhension sur les HPCs. Il y a ainsi :

- Un registre à modifier pour garantir les accès en mode utilisateur. (PMUSERNR)
- Un registre de contrôle global des registres des HPCs permettant notamment l'activation des registres comptant les évènements ou la remise à zéro de tous ces registres. (PMCR)
- Un registre permettant d'autoriser le paramétrage des compteurs. (PMCNTENSET)
- Les registres de paramétrage des registres de compteurs, au nombre de 6 dans notre cas. (PMEVTYPER 0-5)
- Les registres des compteurs, au nombre de 6 dans notre cas. (PMEVCNTR 0-5)
- Enfin, un dernier registre qu'il n'est pas possible de configurer et qui contient automatiquement le nombre de cycles du processeur. (PMCCNTR)

Nous avons donc créé un module noyau donnant l'accès aux registres permettant de configurer les évènements à surveiller et aux registres contenant le nombre de ces évènements. Nous avons ensuite développé un logiciel en langage C permettant de paramétrer les registres PMEVTYPER afin de choisir les évènements à surveiller. Ce logiciel permet également de choisir un pas de temps de lecture des registres d'évènement (le Δ_p secondes du chapitre 4 précédent), afin de les enregistrer dans un fichier particulier. Dans toutes nos expérimentations, nous avons choisi un Δ_p de 5 secondes. Nous avons choisi ce pas en fonction des programmes que nous avons surveillés ; cela correspond à un compromis entre la granularité de surveillance du programme, la taille du fichier de traces à l'envoi et les ressources CPU mobilisées sur l'objet par cette sauvegarde. Enfin, le fait de garder le même pas de lecture nous a permis de comparer plus facilement les résultats et traces que nous avons générés. Enfin, comme les compteurs sont reliés à un coeur du processeur, nous avons forcé l'utilisation d'un seul coeur.

Nous avons choisi de surveiller les 6 compteurs suivants :

- Le nombre de *refill* du cache L1 qui correspond au nombre d'échecs de cache (*cache-miss*) (L1D_CACHE_REFILL)
- Le nombre d'accès au cache L1 (L1D_CACHE)
- Le nombre d'instructions exécutées (INST_RETIRED)
- Le nombre d'exceptions du processeur (EXC_TAKEN)
- Le nombre de branches mal prédites (*misprediction*) (BR_MIS_PRED)
- Le nombre d'accès au bus (BUS_ACCESS)

Nous avons choisi ces compteurs pour plusieurs raisons. Comme nous ne travaillons pas sur les "réels" objets connectés qui seront déployés dans le futur par EDF et que les compteurs sont très dépendants du processeur utilisé, nous avons fait le choix de compteurs "génériques" qui existent ou qui ont un équivalent sur la plupart des processeurs. Le deuxième critère de choix concerne la représentativité des différentes fonctions de la microarchitecture. Nous souhaitions que les compteurs choisis couvrent au maximum ces différentes fonctions. En effet l'objectif est de détecter le plus de comportements anormaux possible, sans forcément en connaître l'origine. Nous avons donc voulu être assez large dans le choix des différents compteurs.

Enfin nous n'avons pas fait d'étude des "meilleurs" compteurs, c'est-à-dire des compteurs permettant de détecter au mieux les différences entre notre programme légitime et le programme attaquant, et ceci pour plusieurs raisons. La première a déjà été évoquée précédemment, les compteurs sont très dépendants du processeur, ainsi les meilleurs compteurs pour distinguer les attaques sur un processeur ne seront pas forcément les mêmes sur un autre. De plus, les nouvelles futures attaques ne seront pas forcément distinguables de la même manière que celles que l'on teste dans nos expérimentations, nous voulons encore une fois rester généraux dans notre approche et détecter tout type de comportement anormal. Cette étude nous a donc semblé extrêmement consommatrice de temps pour un gain probablement très difficile à évaluer et très incertain. Nous avons donc préféré rester sur l'approche consistant à essayer de choisir des compteurs couvrant au mieux les différentes fonctionnalités du processeur.

5.1.2 Tests préliminaires

Pour nos premières expériences, nous avons souhaité vérifier que le comportement des compteurs changeait bien significativement en fonction du logiciel exécuté sur les objets surveillés. Nous avons donc exécuté en parallèle de notre programme de surveillance, divers programmes tels que *PiHole*, un programme de type pare-feu, ou *cpu-miner*, puis collecté les compteurs toutes les 5 à 60 secondes. L'ajout de ces programmes doit a priori être visible dans les compteurs de performance car ils utilisent nécessairement diverses ressources du processeur. Les graphiques 5.1 et 5.2, montrent respectivement les valeurs de l'évolution ($T'_{c,d}$) des évènements (ou compteurs c) INST_RETIRED et L1D_CACHE_REFILL en fonction des évènements

BUS_ACCESS et BR_MIS_PRED. Chaque combinaison d'une couleur et d'une forme correspond aux valeurs d'un même objet d. Nous avons également collecté les valeurs des compteurs lorsqu'aucun logiciel légitime à surveiller n'est exécuté sur l'objet (indiqué par la croix bleue). Comme attendu, il y a bien moins d'évènements dans ce cas. On peut remarquer de manière plus générale que l'exécution des croix bleues génère moins d'évènements que les carrés oranges (correspondant à une application avec peu d'activités) qui génèrent moins d'évènements que les triangles gris (correspondant à l'application `cpu-miner`).

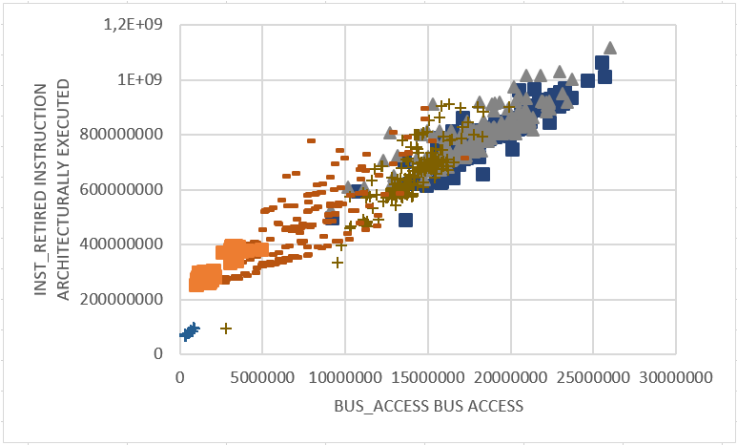


FIGURE 5.1 – Évolution de l'évènement INST_RETIRED en fonction de l'évènement BUS_ACCESS pour différents programmes communs

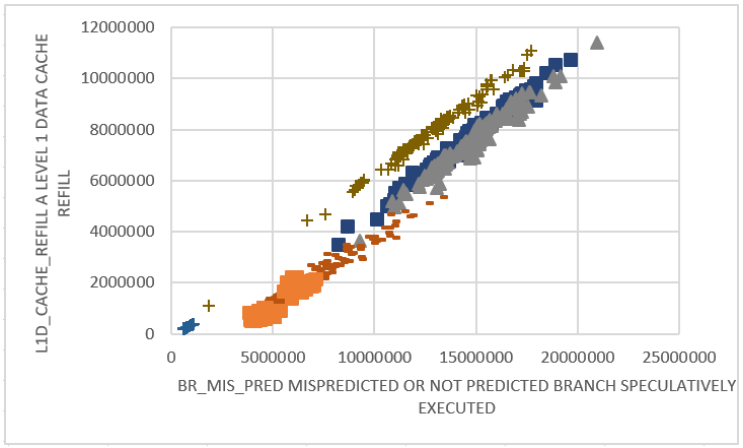


FIGURE 5.2 – Évolution de l'évènement L1D_CACHE_REFILL en fonction de l'évènement BR_MIS_PRED pour différents programmes communs

Dans nos expériences nous avons exécuté les différents programmes sur toutes les Raspberry Pi et comparé les résultats. Nous avons également redémarré les appareils à chaque collecte, pour être sûr que tous les objets soient dans le même état initial. Ces autres résultats sont similaires à ceux présentés précédemment. Ils

confirment que l'impact de l'exécution des programmes sur les HPC est visible. Néanmoins, même si certains programmes sont clairement distinguables visuellement (par exemple les carrés orange et les triangles gris), certains programmes ont des compteurs assez proches (par exemple les triangles gris et les carrés bleus). Ce dernier cas est problématique, car cela signifie que les compteurs ne permettent pas toujours de différencier deux charges similaires. Autrement dit, il pourrait être difficile de détecter un programme malveillant qui influence peu les compteurs.

Ces premiers résultats sont encourageants mais nous avons encore de la latitude pour améliorer l'identification des logiciels via leurs traces. En effet, les logiciels que nous avons exécutés dans ces premières expérimentations, ainsi que le système d'exploitation support, n'ont pas été spécifiquement paramétrés pour être aussi proches que possible d'un objet connecté dans un contexte de déploiement industriel tel que nous l'envisageons pour le contexte de l'énergie.

A cet effet, nous avons donc spécifiquement développé notre propre logiciel, avec un comportement déterministe simple, plus fidèle au comportement d'un objet connecté envisagé dans notre contexte, envoyant des données aléatoires à un serveur central et les sauvegardant dans des fichiers d'archives de manière répétitive. Ce logiciel dispose de trois paramètres que nous avons fait varier : la fréquence de répétition des actions (variant de une fois toutes les secondes toutes les 10 secondes), le nombre de messages envoyés (de 1 à 100), et le nombre de données sauvegardées (dans 1 à 100 fichiers). Les résultats que l'on peut obtenir sont présentés dans les schémas 5.3 et 5.4. Encore une fois, tous les résultats ne sont pas présentés en même temps sur ces graphiques par soucis de clarté, et les valeurs de mêmes couleurs et mêmes formes correspondent aux valeurs de l'évolution $T'_{c,d}$ (cf. chapitre précédent 4) pour un objet d donné. Ces expérimentations, plus en phase avec nos hypothèses, laissent apparaître que les exécutions d'un même programme avec des paramètres différents sont distinguables grâce aux HPCs. Cela correspond à notre hypothèse de menace stipulant qu'un attaquant ne désactive pas le programme surveillé. Il semble donc (visuellement) qu'il soit possible de détecter les changements de paramétrage de notre logiciel, et donc de détecter les attaquants ajoutant une charge au processeur.

Cependant, cette visualisation, même si elle nous conforte dans le choix de notre approche, n'est pas suffisante pour conclure. Dans cet esprit, nous abordons dans la suite nos premières expérimentations dans lesquelles nous avons appliqué un algorithme de détection de valeurs aberrantes sur ces traces.

5.1.3 Preuve de concept

Après cette première phase d'analyse de l'évolution des compteurs, la seconde partie de l'approche concerne la détection d'anomalie sur le serveur. Comme nous avons fait le choix dans ce chapitre de décrire chronologiquement les différentes expérimentations que nous avons menées, ainsi que les leçons tirées de chaque expérimentation, nous sommes donc toujours à ce stade dans une phase de construction de l'approche dans laquelle la phase de transformation des données n'était donc pas

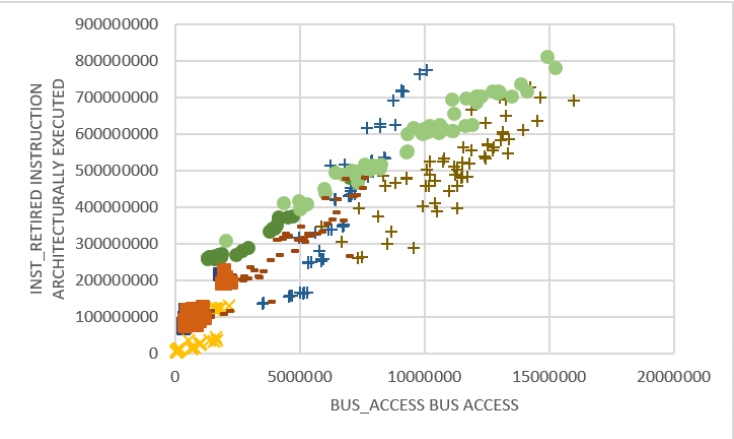


FIGURE 5.3 – Évolution de l'évènement INST_RETIRED en fonction de l'évènement BUS_ACCESS pour différents paramétrage d'un logiciel simple et répétitif

encore complète en comparaison de l'approche définitive présentée dans le chapitre précédent. Dans cette expérimentation, nous n'avons notamment utilisé qu'un seul algorithme de détection de valeurs aberrantes, LOF. Nous avons choisi cet algorithme dans un premier temps car il permet d'attribuer un score aux données, ce qui nous semblait intéressant pour évaluer dans un premier temps nos anomalies. Le but de ces expérimentations est donc de déterminer si un algorithme de détection de valeurs aberrantes peut nous permettre de distinguer des programmes exécutant le même type d'activité mais avec des intensités différentes.

Pour ces expérimentations, nous avons choisi de comparer 100 traces générées par nos 10 objets. Nous avons inclus dans ces 100 traces, 10 traces d'objets corrompus. Les logiciels légitimes sont les mêmes que ceux cités dans la partie précédente. Le logiciel malveillant qui s'exécute en plus sur les objets corrompus envoie une dizaine de messages supplémentaires à la même fréquence que l'envoi de données légitimes. Nous sauvegardons l'état des compteurs toutes les 5 secondes et récupérons les fichiers contenant les traces toutes les 30 minutes. Nous montrons dans cette partie, à titre d'illustration, une seule des expérimentations que nous avons menées, mais nous avons en réalité essayé différents paramétrages de notre logiciel légitime et exécuté d'autres logiciels malveillants comme un algorithme de minage ou l'envoi intempestif de messages, avec des résultats similaires. L'idée sous-jacente à l'utilisation de différents programmes malveillants était de solliciter les différentes ressources du processeur (CPU, réseau, stockage) afin de vérifier si, à chaque fois, nous étions dans la capacité de détecter des anomalies.

Comme précisé dans l'approche, nous calculons alors les 7 statistiques descriptives de chaque trace : minimum, moyenne, maximum, écart-type, 1^{er} quartile, médiane, $3^{ème}$ quartile. Nous n'avons pas procédé à d'autres transformations pour la preuve de concept à ce stade. Les deux graphiques 5.5 et 5.6 représentent 2 des 49 caractéristiques (7 statistiques pour chacun des 7 compteurs) étudiées et sont représentatifs des autres résultats obtenus (caractéristiques nommées $F_{c,d}$ dans le

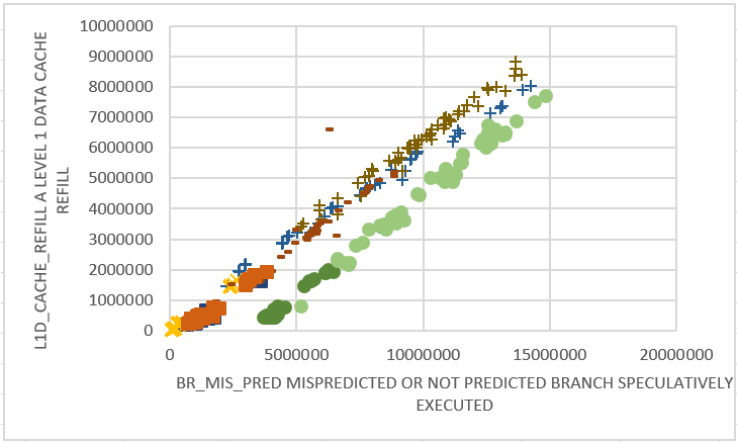


FIGURE 5.4 – Évolution de l'évènement L1D_CACHE_REFILL en fonction de l'évènement BR_MIS_PRED pour différents paramétrage d'un logiciel simple et répétitif

chapitre précédent). Ces graphiques représentent la valeur d'une caractéristique en fonction du numéro de la trace. Nous avons appliqué LOF sur ces données à une seule dimension et les valeurs aberrantes telles que détectées par LOF, qui a été paramétré comme la littérature le suggère, sont affichées en rouge.

Dans la figure 5.5, il est aisé de distinguer les attaques, qui sont les traces numérotées de 71 à 80. De nombreuses caractéristiques fournissent des résultats identiques pour les différentes attaques que nous avons effectuées pour la preuve de concept. Il est aussi possible de remarquer que LOF souffre d'un certain nombre de faux positifs, pour les données légitimes les plus élevées et pour les moins élevées. Cependant, les attaques se distinguent largement dans ce cas là et il est donc possible d'améliorer les résultats en changeant les valeurs de seuils de détection de LOF.

Toutes les caractéristiques ne sont pas nécessairement aussi efficaces les unes que pour les autres pour la détection. Par exemple, la figure 5.6 montre que la médiane ne permet pas facilement de distinguer les attaquants dans ce jeu de données. En effet, lorsque l'on applique LOF sur les données de cette figure, avec les paramètres conseillés, les données des attaquants sont bien considérées comme des attaques, mais le nombre de faux positifs est élevé. Comme l'attaque considérée ici consiste à envoyer des paquets supplémentaires qui s'ajoutent aux messages légitimes, le maximum et la moyenne des évènements sont plus fortement affectés que la médiane, le minimum ou les quartiles pour nos attaques.

Nous avons ensuite décidé de considérer le cumul de plusieurs scores LOF associés à différentes caractéristiques et de considérer comme positif les points dont le score cumulé dépasse un certain seuil. Avec différents algorithmes LOF exécutés sur les caractéristiques en 1 dimension, nous avons ainsi additionné les scores de chaque trace afin d'obtenir un score d'aberrance global. Le score LOF est normalement positif mais, pour des raisons de cohérence avec d'autres algorithmes de détection d'outlier, l'implémentation que nous utilisons (celle de *scikit-learn*

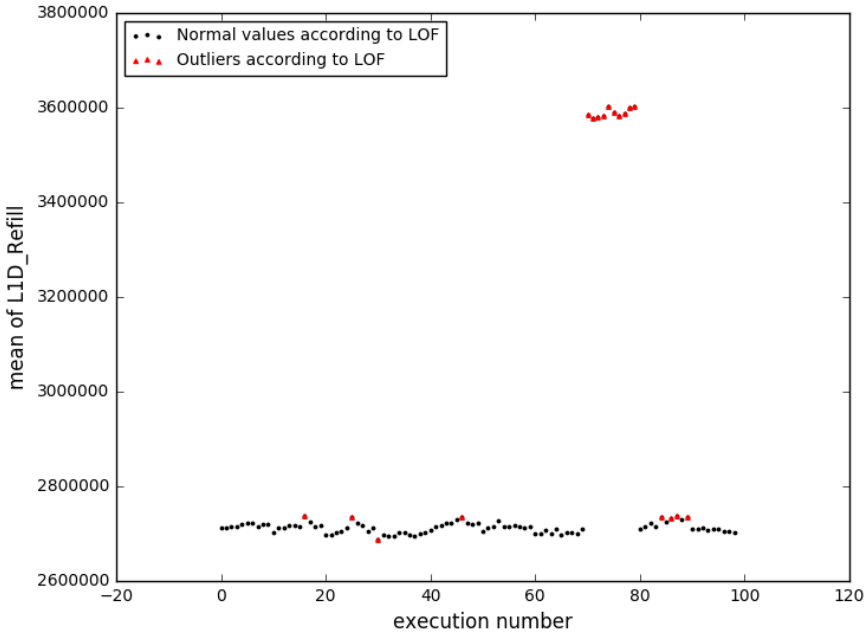


FIGURE 5.5 – Moyennes de l'évènement L1D_Refill en fonction du numéro de la trace. Les attaques sont aux indices 71 à 80 et les points rouges ont été identifiés comme anomalies par LOF.

[Pedregosa 2011]) renvoie le score opposé (c'est-à-dire un nombre négatif). Nous obtenons ainsi le schéma 5.7. Il est possible de constater que même si les attaques sont facilement distinguables grâce aux scores, certaines traces légitimes ont également des scores d'aberrance élevés. Néanmoins, il est possible dans cet exemple de trouver un seuil (vers -500) qui permette de séparer exactement les attaques des charges légitimes.

Toujours dans l'optique de pouvoir visualiser nos résultats, nous avons également étudié les caractéristiques en deux dimensions. En effet, augmenter les dimensions permet de mener une analyse plus complète et de mieux distinguer des valeurs aberrantes indétectables lors de l'utilisation d'une seule dimension. Toutefois, afin que les résultats puissent être analysés visuellement, nous nous sommes contentés de mener des expérimentations à deux dimensions seulement. Nous avons par exemple obtenu le graphique 5.8 en comparant les maximums des évènements EXC_TAKEN et BR_MIS_PRED des différentes traces. Comme pour les autres graphiques, les points de couleur rouge montrent les éléments considérés comme des valeurs aberrantes pour LOF. Nous nous servirons de ce graphique dans les discussions sur les améliorations apportées à notre preuve de concept, qui donneront lieu à notre approche complète. Ces graphiques ont été utilisés afin de comprendre les données et d'améliorer notre approche. Nous discutons de cette étape d'amélioration dans la prochaine sous-section.

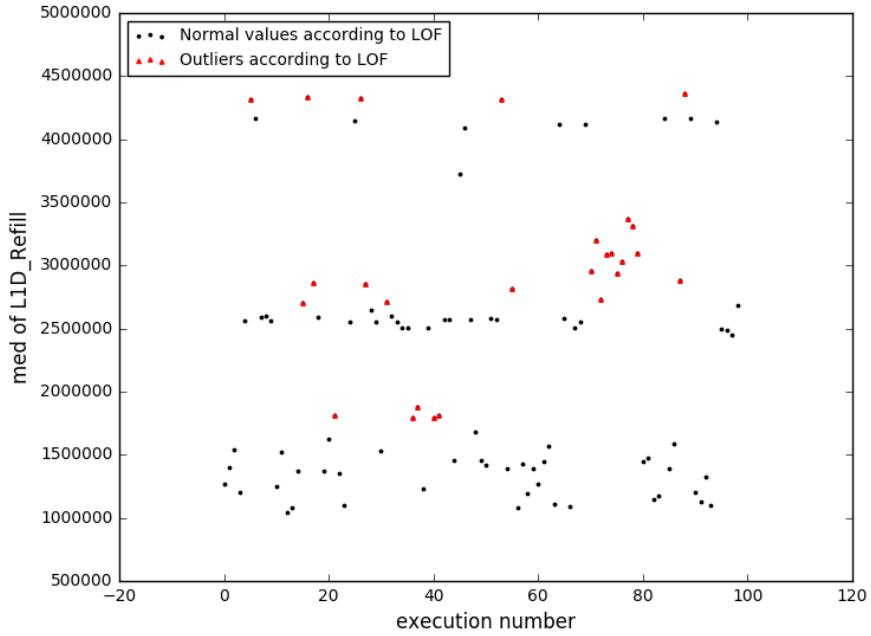


FIGURE 5.6 – Médianes de l'évènement L1D_Refill en fonction du numéro de la trace. Les attaques sont aux indices 71 à 80 et les points rouges ont été identifiés comme anomalies par LOF.

Enfin nous avons appliqué LOF sur les vecteurs composés de toutes les caractéristiques (49) sur le graphique 5.9. Comme nous l'avons précisé dans le chapitre 4, c'est l'approche que nous adopterons finalement. Néanmoins, dans cette expérience préliminaire, nous n'avons pas réalisé de standardisation des données, ce qui explique la présence d'un certain nombre de faux positifs. La standardisation n'était pas appliquée dans les expériences précédentes où nous nous appuyions sur la somme de scores LOF car les scores LOF sont normalisés (comme expliqué dans la section 4.3.2). Le fait que la standardisation ne soit pas nécessaire est un avantage. En effet, comme nous l'avons expliqué dans la section 4.2.5, il est difficile d'effectuer une standardisation robuste en présence de valeurs aberrantes très élevées par exemple, et nous avons donc dû nous appuyer sur un algorithme de spécifiquement adapté pour cela, que nous détaillé dans cette section. Dans le cadre de ces premières expérimentations, nous n'avions donc pas encore réalisé cette standardisation et nous nous attendions donc à des faux positifs relativement élevés. Le graphique 5.9 nous montre d'ailleurs que se passer de standardisation est délicat dans l'approche qui manipule directement les 49 features et ne procède pas à une somme. Pour autant, cette méthode nous a montré que même en l'absence de standardisation, nous obtenions un nombre limité de faux positifs, ce qui était encourageant.

Entre ces deux méthodes, l'utilisation de la somme ou un traitement en dimension 49, nous avons privilégié la seconde. Bien qu'elle nécessite une standardisation,

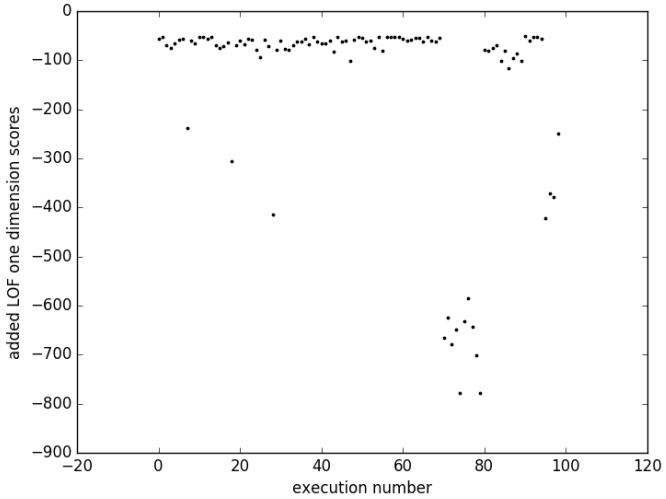


FIGURE 5.7 – Addition des scores LOF pour les caractéristiques seules. Les attaques sont aux indices 71 à 80.

la manipulation en dimension 49 permet de détecter des anomalies qui ne peuvent pas être détectées en inspectant individuellement chaque dimension. Nous nous appuyons également sur l'hypothèse de modèle de menace 4 selon laquelle il y a peu d'anomalies dans une population, ce qui réduit ainsi la probabilité qu'il y ait tant d'anomalies que la standardisation serait si peu fiable que la détection s'en retrouverait dégradée.

5.1.4 Amélioration de l'approche et des expérimentations

Cette première série d'expérimentations nous a été fort utile pour nous conforter dans la pertinence de notre approche. Mais il reste des points d'améliorations nécessaires. Nous les présentons dans cette section.

Le premier point que nous abordons ici concerne la mise à l'échelle. En effet, nous avons utilisé, dans la preuve de concept, seulement des données de 10 objets et comparé 100 traces. Cependant, notre contexte correspond à un déploiement massif d'objets connectés et donc nécessite de pouvoir réaliser des expérimentations sur plus de traces. Dans nos prochaines expérimentations, il est donc important de prendre cet aspect en compte. Pour cela, nous avons fait en sorte de multiplier par 10 le nombre d'objets générant des traces et réaliser des analyses sur 10 000 traces simultanément. Nous avons pour cela déployé une baie comprenant 100 Raspberry Pi (cf. figure 5.10), grâce auxquels nous avons pu mener des expérimentations plus massives, plus représentatives des cas d'usage réels que l'on envisage.

Le second point concerne l'absence de standardisation, pour des raisons que nous avons évoquées dans la section précédente. Nous avons donc déterminé un algorithme de standardisation robuste adapté à nos expérimentations et l'avons

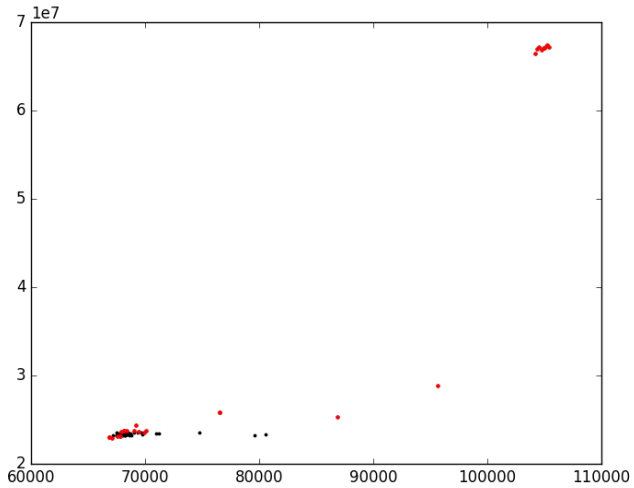


FIGURE 5.8 – Maximums de l'évènement EXC_TAKEN en fonction des maximums de l'évènement BR_MIS_PRED

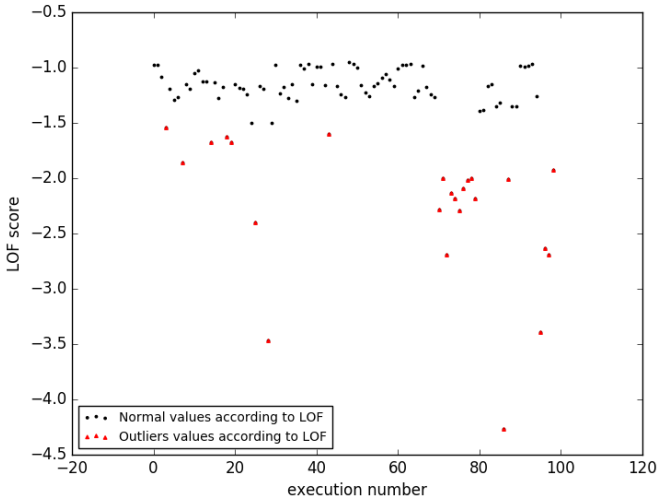


FIGURE 5.9 – Score de l'algorithme LOF exécuté sur toutes les caractéristiques des traces

systématiquement utilisé dans la suite sur la nouvelle plateforme.

De la même manière, en regardant les différents scores obtenus sur cette expérience (graphiques 5.7 5.8 ou 5.9), ainsi que sur celles que nous n'avons pas représentées ici, des faux positifs que l'on peut éliminer apparaissent régulièrement. En effet, nous avons identifié que certains faux positifs étaient en réalité dus en particulier à l'exécution de certains services du système d'exploitation, qui en réalité ne correspondaient pas à des services susceptibles de s'exécuter sur des objets

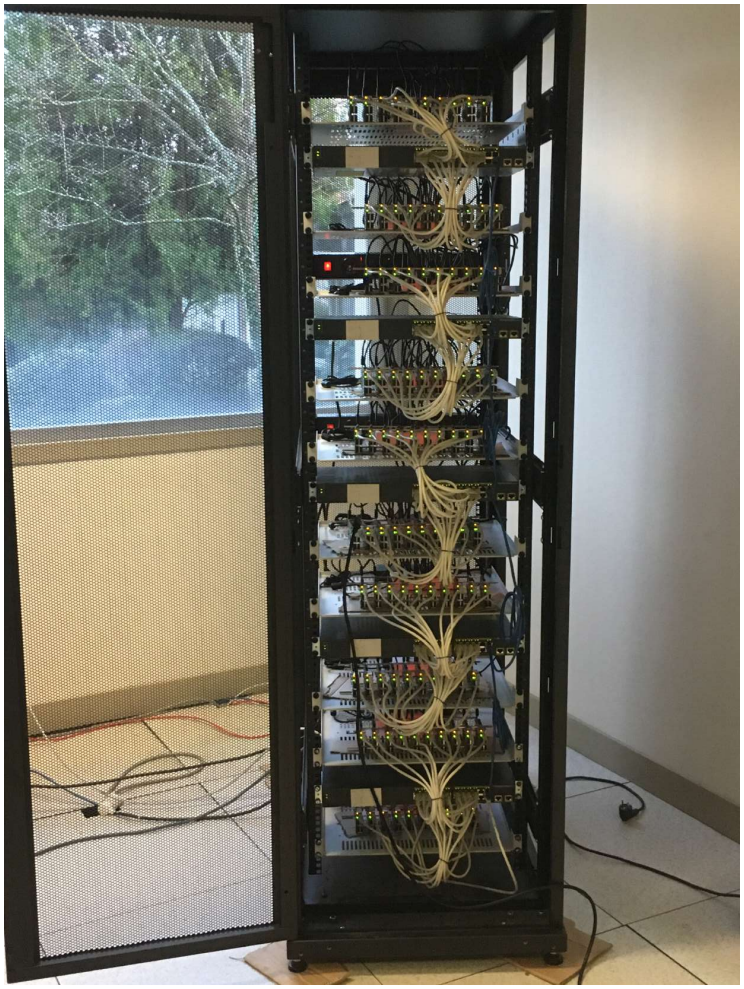


FIGURE 5.10 – Photographie de la baie composée de 100 Raspberry Pi

connectés envisagés dans notre étude. Comme décrit dans le chapitre précédent, les objets envisagés ne sont pas des systèmes complexes et exécutent des applications très simples. Ainsi le système d'exploitation sur lequel nous exécutions nos programmes jusqu'alors n'était pas suffisamment adapté à nos hypothèses. Dans la suite des expérimentations nous nous sommes donc spécifiquement penchés sur la configuration de ce système d'exploitation de manière à désactiver tous les services qui nous semblaient inopportuns pour nos cas d'usage, de façon à nous approcher autant que possible de la configuration simple d'un objet connecté.

Enfin concernant la partie détection de valeurs aberrantes de cette première série d'expérimentation, le seul algorithme que nous avons testé dans la preuve de concept est LOF. Cependant, il est tout à fait possible que d'autres algorithmes soient plus adaptés et donnent des meilleurs résultats. Nous avons donc testé dans la suite différents algorithmes de détection de valeurs aberrantes (tous les algorithmes

présentés dans le chapitre précédent). Nous avons décidé d'appliquer directement nos algorithmes sur toutes nos caractéristiques, l'utilisation et l'agrégation de score étant trop spécifiques à certains algorithmes et rendant la tâche bien plus complexe pour des résultats similaires.

Nous présentons ainsi dans la suite de ce chapitre une nouvelle série d'expérimentation, pour lesquelles nous avons concrètement apporté des améliorations afin 1) de rendre notre plateforme plus proche des hypothèses de déploiement envisagé, et 2) de rendre la détection plus efficace.

5.2 Mise à l'échelle et améliorations des expérimentations

Dans cette partie, les expérimentations sont donc totalement conformes à l'approche présentée dans le chapitre précédent. Nous présentons, de la même manière que précédemment, les différentes expérimentations que nous avons menées, les résultats et conclusions associés.

5.2.1 Nouvelle plateforme d'expérimentations

Conformément aux points d'amélioration que nous avons identifiés, nous avons modifié certaines parties de l'implémentation de nos expérimentations et de la plateforme de test. En premier lieu, nous avons utilisé 100 Raspberry Pi 3B, avec un processeur ARM Cortex A53 (ARMv8), toujours avec le système d'exploitation Raspberry Pi OS et un seul cœur activé. Cependant le système d'exploitation a été soigneusement paramétré, c'est-à-dire que nous avons activé seulement un minimum de services et de processus d'arrière plan, pour être plus fidèle à l'environnement d'exécution d'un objet connecté de notre contexte d'étude. Les compteurs analysés et les caractéristiques ($F_{c,d}$) restent les mêmes.

Concernant les logiciels surveillés, nous avons considéré quatre profils différents que l'on note $S1$ à $S4$:

- $S1$ se focalise sur la collecte et la sauvegarde d'informations, pour les transmettre régulièrement au serveur central. Il collecte des informations comme la température, la fréquence du processeur et des informations et journaux systèmes, qui sont sauvegardés comme archive sur l'objet avant de les envoyer au serveur.
- $S2$ est un programme léger qui collecte certaines valeurs venant du système pour y appliquer des calculs simples et envoie les entrées et les résultats au serveur central sous la forme d'un seul fichier. L'activité I/O est ainsi près de 5 fois inférieure à celle de S1, l'utilisation du CPU légèrement inférieure, et l'utilisation du réseau plus de 10 fois inférieure.
- $S3$ est une application plus intensive pouvant correspondre à un objet surveillant continuellement la consommation d'un bâtiment pour envoyer des

rapports réguliers au serveur central. Ce logiciel à l'activité CPU la plus intense, près de deux fois celle de S1. L'activité I/O est cependant plutôt faible, un peu supérieure à celle de S2. Finalement le débit de communication est près de 3 fois supérieur à S2 mais le nombre de paquets échangés inférieur.

- *S*4 est un programme utilisant plus intensivement le réseau, et pourrait correspondre à une simple passerelle pour des capteurs ou des actionneurs par exemple. Ce logiciel à une utilisation activité CPU globale supérieure à S1, avec légèrement moins d'activité I/O et près de 5 fois plus d'activité réseau.

Pour valider l'efficacité de notre détection, nous considérons des activités malveillantes qui pourraient émerger lorsque les objets sont compromis. Dans nos expérimentations, nous avons défini différentes charges malveillantes correspondant aux différents objectifs des attaquants. Par exemple un attaquant pourrait compromettre les objets pour mener ensuite des attaques massives avec les objets compromis, comme avec l'attaque Mirai présentée dans le chapitre 1. Pour ce type d'attaque, nous avons pensé à deux utilisations différentes des objets. La première est le minage intensif de crypto-monnaies par les objets compromis. La deuxième, plus classique, est l'utilisation de la fonction de communication des objets dans le cadre d'attaques de type DDoS. Nous nous sommes également inspirés de la méthode de propagation de Mirai pour créer une charge malveillante où les objets compromis essaient de compromettre d'autres appareils connectés en effectuant des attaques de type force brute en utilisant le protocole SSH. Pour cela nous avons utilisé l'outil *Hydra*. Nous avons également collecté les traces des objets qui subissaient de telles attaques par force brute. De plus, nous avons considéré une attaque dans laquelle un attaquant, après avoir compromis un objet, effectue une phase de reconnaissance et d'étude de l'objet. Pour cela, nous nous sommes inspirés d'une étude menée dans notre laboratoire quelques années auparavant ([Nicomette 2011]), dans laquelle nous avions étudié le comportement de tels attaquants sur des "pots de miel". Ces attaquants, une fois connectés sur leur cible, effectuent en général une phase de reconnaissance, puis essaient d'identifier des fichiers ou dossiers vulnérables, ou essaient d'augmenter leurs privilèges ou bien encore essaient d'attaquer d'autres objets depuis cette cible. Ces attaques peuvent se révéler plus subtiles que les autres mais très intéressantes à détecter. Enfin, nous avons considéré une dernière attaque consistant simplement à une utilisation plus intensive du CPU, grâce à l'outil *stress-ng*.

Nous avons donc créé 8 charges malveillantes notés P1 à P8 pour nos expériences, basées sur des outils et des attaques réelles :

- P1 : La charge correspond à une attaque DDoS. Elle envoie de nombreux messages en UDP. Pour cela nous avons utilisé l'outil *Hping3* et envoyé des paquets UDP de 1kB toutes les secondes.
- P2 : Cette charge est la même que la précédente, mais de manière encore plus intensive. Nous avons cette fois-ci envoyé les mêmes paquets UDP toutes les millisecondes.

- P3 : Cette charge utilise un programme [1] pour miner de la crypto-monnaie.
- P4 : Cette charge simule celle d'un attaquant effectuant une étape d'analyse d'une cible après avoir réussi à s'y connecter. Cette phase est composée ici d'une quarantaine de commandes espacées sur 3 minutes, en utilisant les commandes basiques retrouvées dans les fameux pot de miel de la recherche de [Nicomette 2011].
- P5 : Cette charge est la même que la précédente mais de manière plus intensive. Plus concrètement, nous avons répété l'attaque précédente en espaçant chaque occurrence des attaques d'une minute.
- P6 : Cette charge simule une attaque par force brute de mots de passe en utilisant le protocole SSH. Ces attaques sont effectuées d'une Raspberry Pi à une autre. Pour cela nous avons utilisé l'outil *Hydra*.
- P7 : Cette charge correspond à la trace des activités d'une machine victime de l'attaque présentée en P6.
- P8 : Cette charge utilise les ressources du processeur. Pour cela nous avons utilisé l'outil *stress-ng* et fait varier le pourcentage d'utilisation du processeur entre 30, 50 et 70% toutes les minutes.

Nous avons réalisé plusieurs versions d'une même attaque pour vérifier que dans le cas de différents paramétrages d'une même attaque, celle-ci reste toujours détectable.

Enfin, concernant la mise à l'échelle de l'étude ainsi que son adéquation avec un cas réel, nous étudions maintenant 10 000 traces simultanément. Pour chacune des traces, les objets exécutent soit un programme légitime S, soit un programme légitime S plus une charge malveillante P. Nous avons donc généré un plus grand nombre de traces à partir de nos 100 objets. Nous avons ensuite échantillonné nos traces afin de mener nos expériences et exécuter notre algorithme de détection. Nous avons basé cet échantillonage sur deux paramètres : la proportion de la population compromise μ et la taille de la population étudiée n (ici 10 000). Nous tirons ainsi aléatoirement des populations contenant $(1 - \mu)n$ traces d'objets légitimes et μn traces d'objets compromis. Nous avons étudié dans nos expériences les résultats pour des attaques homogènes (tous les objets compromis de la même manière exécutent la même charge) et hétérogènes (les objets compromis exécutent des charges malveillantes différentes tirées équiprobablement).

Les algorithmes de détection de valeurs aberrantes que nous avons choisis possèdent certains paramètres qu'il est nécessaire de choisir. Nous choisissons certains paramètres, comme le nombre de voisins de LOF par exemple, en fonction de ce que suggère la littérature. Cependant, pour d'autres paramètres, une étape de calibrage était nécessaire. Étant donné que calibrer ce paramètre sur une application rendrait la détection adaptée à cette application spécifiquement, nous allons évaluer la transférabilité des paramètres entre plusieurs applications. Autrement dit, évaluer l'utilisation d'un paramètre appris avec une première application pour surveiller

1. `https://github.com/pooler/cpuminer`

une seconde application afin de se placer dans des conditions plus réalistes, voire pessimistes, de l'utilisation de notre détecteur. Cette étape de calibrage est, dans la réalité, exécutée une fois pour toutes avant le déploiement des objets.

Grâce à ces expérimentations, nous avons cherché à répondre aux questions suivantes :

- Quels sont les algorithmes les plus efficaces, indépendamment de leurs paramètres ?
- Quel est l'impact de la proportion d'attaque μ sur l'efficacité des algorithmes ?
- Comment un algorithme calibré avec un logiciel performe sur un autre logiciel ?
- Quel est l'impact de la mise à l'échelle sur de tels algorithmes ?

La génération de ces grandes quantités de données nous permet d'avoir des données plus réalistes et plus proches des cas d'usage envisagés et nous permet ainsi de pouvoir apporter des réponses solides à ces différentes questions. Après avoir récolté ces données, nous avons utilisé plusieurs algorithmes de détection de valeurs aberrantes décrits dans la description de l'approche du chapitre précédent (section 4.3), afin d'avoir une vision plus complète des possibilités de l'analyse de nos données.

5.2.2 Analyses et résultats

Nous présentons les résultats de ces expériences en plusieurs étapes. Tout d'abord, nous réalisons une étude du "potentiel" des détecteurs sans calibration, à partir de courbe ROC (section 5.2.2.1), puis nous détaillons l'étape de calibration (section 5.2.2.2) qui permet de choisir les paramètres pour les détecteurs retenus. Nous présentons ensuite les performances de nos détecteurs (section 5.2.2.3) et nous finissons avec une analyse de leur complexité algorithmique et de leur passage à l'échelle (section 5.2.2.4).

5.2.2.1 Courbes ROC

Afin d'analyser la performance des différents algorithmes en fonction des autres paramètres (comme le seuil de détection), nous utilisons ici les courbes ROC (Receiver Operating Characteristic), traçant le taux de vrai positifs en fonction du taux de faux positifs en faisant varier les paramètres détection de ces algorithmes. Dans ces expériences, nous considérons 1% d'objets compromis. Les graphiques 5.11, 5.12, 5.13 et 5.14 représentent les courbes ROC pour chacun des logiciels $S1$ à $S4$. Chaque point d'une courbe correspond aux performances d'un détecteur (en fonction de son taux de vrais positifs, en ordonnée, et de son taux de faux positifs, en abscisse) pour un certain paramètre. Ces courbes permettent donc d'évaluer le "potentiel" de détection des algorithmes indépendamment de la valeur de leurs paramètres. Les points d'intérêts de telles courbes se situent donc en haut à gauche de celle-ci, le but étant d'obtenir le meilleur taux de détection des attaques pour le moins de fausses alertes possible. Par exemple, il est clair sur la courbe 5.13 que

HDBSCAN (en jaune) possède un potentiel moins intéressant en terme de détection que les autres algorithmes. De la même manière, l'algorithme Isolation Forest est souvent en-deçà des autres, tout comme ODIN. Il serait pourtant erroné de conclure que les algorithmes qui ont les meilleures courbes ROC sont toujours les plus efficaces. En effet, ces courbes n'indiquent en rien la sensibilité de l'algorithme à ses paramètres. Par exemple, peut-être que le seuil de détection pour LOF donnant le meilleur résultat est 2 dans le cas de S1 et 3 dans le cas de S2, et qu'utiliser la valeur 2 pour S2 ou la valeur 3 pour S1 aboutit à de mauvaises performances. Ainsi ces courbes nous permettent uniquement de conclure qu'il est possible pour chacun des logiciels séparément, de trouver un paramétrage permettant une bonne détection. Pour cette raison, de cette étude des courbes ROC nous ne disqualifions que l'algorithme HDBSCAN dont les performances potentiellement sont significativement plus basses que celles des autres.

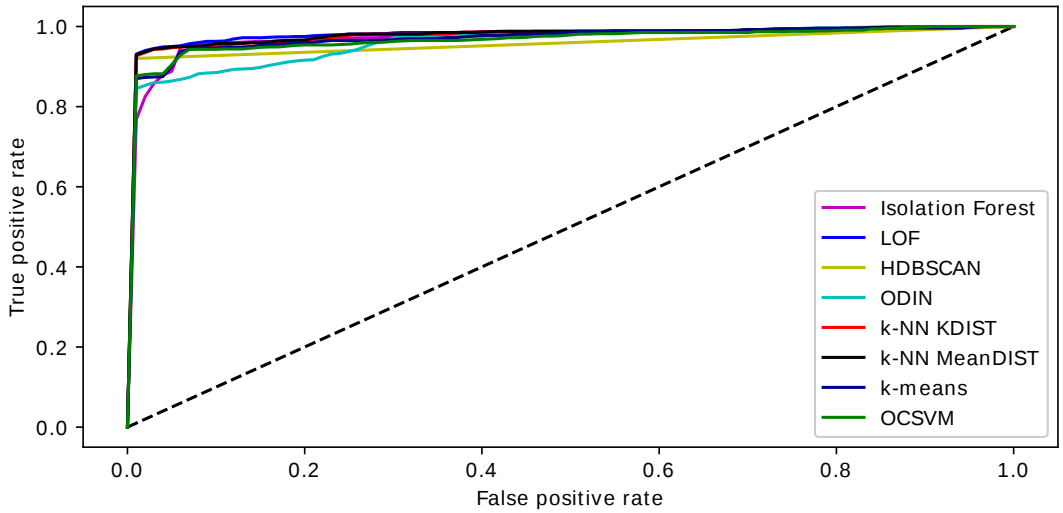


FIGURE 5.11 – Graphique ROC pour le logiciel légitime S1

5.2.2.2 Calibration

L'étude ROC étant terminée, nous détaillons comment sélectionner les paramètres pour les algorithmes que nous évaluons. Certains algorithmes de détection attribuent un score à chacune des données. En général plus le score est élevé et plus la donnée est aberrante. Il faut donc définir le seuil de score qui déclenche la levée d'alerte. Ce seuil peut être fixe ou il est possible également de définir une heuristique afin d'en avoir une estimation à partir de la population étudiée. La seule heuristique dont nous avons connaissance est utilisée dans le cas des algorithmes KDIST et MeanDIST. Cette heuristique, décrite dans [Hautamaki 2004], que nous appellerons HKF (du nom des auteurs), donne de mauvais résultats car elle n'est pas assez robuste, qui est très variable dans notre cas à cause des attaques. Nous avons donc de la même manière que pour la standardisation robuste, utilisé le $95^{ème}$ centile plutôt que le maximum utilisé dans l'article initial. Nous appellerons cette

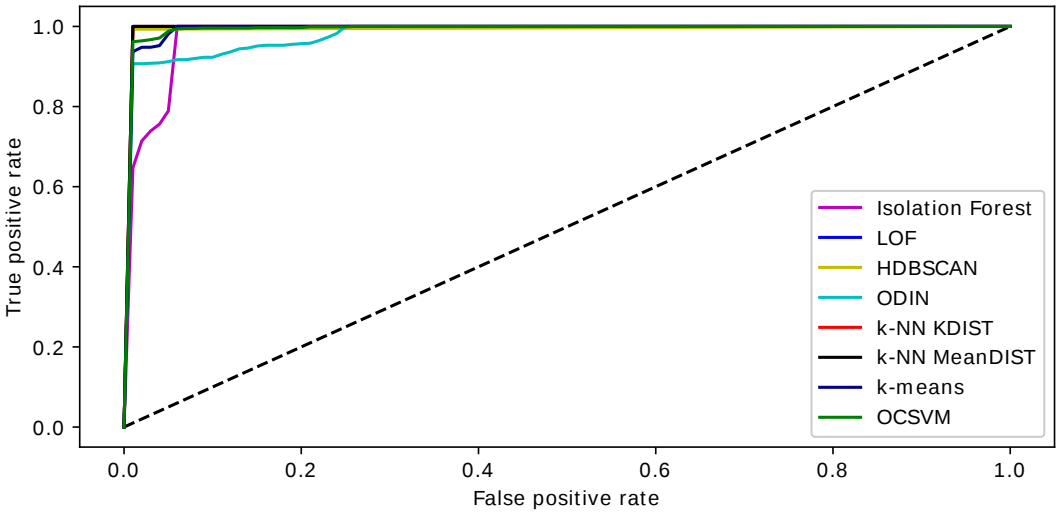


FIGURE 5.12 – Graphique ROC pour le logiciel légitime S2

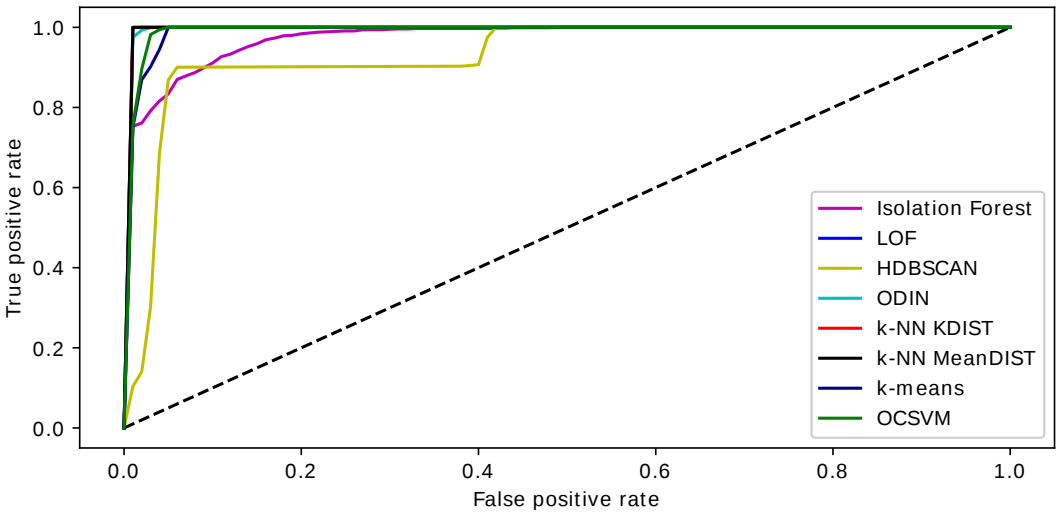


FIGURE 5.13 – Graphique ROC pour le logiciel légitime S3

heuristique HKF+ dans la suite du document. Nous utilisons également une autre heuristique appelée α-*med*. Cette heuristique s'appuie sur un ensemble de distances d et un paramètre $\alpha > 1$: le seuil $\tau_\alpha(d)$ est défini tel que $\tau_\alpha(d) = \alpha\,\text{median}(d)$. D'autres algorithmes comme HDBSCAN ont un ou des paramètres différents du seuil de détection, mais dont la finalité sur l'efficacité de l'algorithme est similaire.

Enfin, les algorithmes basés sur le voisinage (LOF, KDIST et MeanDIST et ODIN) ont besoin d'un paramètre k qui est le nombre de plus proches voisins à utiliser. Pour les algorithmes LOF, KDIST et MeanDIST, nous choisissons la valeur 100 ($\sqrt{n}$ avec n la taille de la population étudiée). Pour l'algorithme ODIN, nous choisissons 1 000 ($10\sqrt{n}$). La calibration se base sur une expérimentation avec 1% d'attaquants. Le seuil est choisi de manière à avoir le meilleur recall possible tout

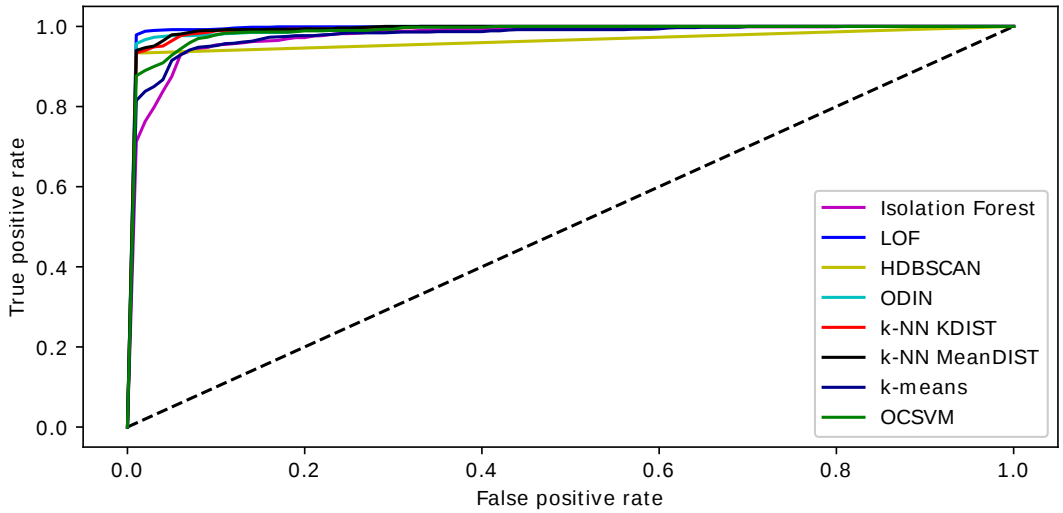


FIGURE 5.14 – Graphique ROC pour le logiciel légitime S4

en ayant une précision d'au moins 90% (pour rappel, la définition de ces mesures a été donnée dans la sous-section 3.2) ; s'il était impossible d'avoir une précision dépassant 90%, nous choisissions le paramètre qui maximisait la précision.

5.2.2.3 Efficacité de la détection

Une fois cette étape calibrage réalisée pour chacun des 4 logiciels avec 1% d'objets compromis, nous avons testé notre détecteur sur chaque logiciel en utilisant systématiquement le calibrage réalisé pour un autre logiciel. Ainsi, ces expérimentations nous permettent d'évaluer la robustesse de cette phase de calibrage, face aux mises à jour par exemple. Si les performances de la détection sont très bonnes pour un logiciel en utilisant le calibrage réalisé pour un autre logiciel, nous avons de bonnes raisons de penser que la détection restera performante sur un logiciel lorsque celui-ci sera mis à jour, sans que l'étape de calibration soit réalisée pour cette nouvelle mise à jour (ce que nous voulons éviter justement dans notre approche).

Nous avons donc testé notre détecteur lorsque 0%, 0,1%, 0,5%, 1%, 2% et 5% des objets sont compromis, en prenant des traces représentant les différents types de compromissions que nous avons considérée, réparties de façon équiprobable. Les résultats sont présentés dans les tableaux 5.1 et 5.2. Dans ce tableau et les suivants, TPR se réfère au taux de vrais positifs et FPR au taux de faux positifs. Nous n'avons pas inclus les résultats pour l'algorithme KDIST car ils sont quasiment identiques aux résultats de l'algorithme MeanDIST. On peut tout d'abord remarquer que le taux de faux positifs est très faible dans la plupart des cas (inférieur à 0,20%). Les cas ou le taux de faux positifs est supérieur à 1% correspondent toujours à l'utilisation de *k-means* ou *Isolation forest*. Concernant le taux de vrais positifs, ces deux mêmes algorithmes ainsi que *OCSVM* obtiennent les moins bons résultats, toujours inférieurs à 80% voire même inférieur à 50%, même en présence de très peu d'attaques. En dehors de ces cas, le taux de vrais positifs est satisfaisant globalement,

souvent autour des 80% ou 90% de détection avec un cas à 100% de détection.

Pour la suite de l'analyse des résultats, nous nous sommes ainsi focalisés sur les résultats des algorithmes les plus efficaces : ODIN, LOF, MeanDIST α-*med* et MeanDIST HKF+. En effet, pour ces algorithmes, le taux de faux positifs est toujours inférieur à 0,25%, et le taux de détection très bon dans la plupart des cas. Enfin, nous pouvons noter une faiblesse dans la façon dont nous utilisons les algorithmes de détection de valeurs aberrantes : le taux de détection est médiocre lorsqu'il y a trop d'attaques simultanées (à partir de 5% des objets compromis). Cela vient du fait que lorsque les attaques sont trop nombreuses, elles forment de petits groupes cohérents plus difficiles à repérer par les algorithmes. De plus, à partir de 5% d'objets compromis, la standardisation robuste peut être moins fiable. Cependant, l'intérêt de la détection de valeurs aberrantes est de lever des alertes au plus tôt, et dans l'idéal bien avant 5% d'objets compromis. Il est de toutes façons très peu probable que 5% d'objets soient compromis exactement au même moment, faisant varier brutalement de 0% à 5% le pourcentage d'objets compromis. Ce pourcentage va varier progressivement entre 0% et 5%, et notre système de détection sera en capacité de lever une alerte dès le premier pourcent d'objets compromis.

Les précédentes expériences ont été faites avec une calibration réalisée avec un logiciel qui n'était pas celui évalué. Nous avons également étudié la stabilité de la phase de calibrage des algorithmes en relançant ces expériences en utilisant cette fois les paramètres optimisés et en analysant la différence de performances. Nous comparons les taux de faux positifs (FPR) et les taux de vrais positifs (TPR). Pour chacun des algorithmes, la différence de taux de faux positifs est très faible, 0,12% pour la plus élevée. Concernant les taux de vrais positifs, les résultats sont alors plus dépendants des algorithmes. La différence est en moyenne inférieure à 4% pour LOF et MeanDIST α-*med*, 8,00% pour ODIN, et 16,71% pour MeanDIST HKF+. Ces résultats sont à mettre en relation avec les taux de détection présentés dans le tableau 5.2. Ces résultats indiquent que LOF et MeanDIST α-*med* sont les algorithmes les plus stables au regard du paramétrage. MeanDIST HKF+ est en revanche beaucoup plus variable. Ce qui peut représenter un désavantage non négligeable dans le cas de notre approche.

Les résultats du tableau 5.1 sont présentés de manière globale, sans distinguer les différentes attaques. Il est ainsi possible que certaines attaques soient plus difficilement détectables avec notre approche. Pour vérifier cela, nous avons étudié l'efficacité de détection des algorithmes en fonction du type d'attaque étudié. Les tableaux 5.3 et 5.4 donnent les résultats de détection pour chacune des attaques lorsqu'il y a 1% d'objets compromis. Nous avons, comme précisé précédemment, étudié uniquement les résultats pour les algorithmes les plus efficaces. Cette étude et ce tableau nous montrent que la détection est excellente pour la plupart des charges malveillantes et la plupart des algorithmes. Les charges malveillantes qui se distinguent sont la charge P7, c'est-à-dire les victimes de brute force SSH et la charge P4, c'est-à-dire les objets victimes d'une phase de découverte et d'analyse par des attaquants ayant réussi à s'y connecter. Cependant, la charge P4 est globa-

TABLE 5.1 – Sensibilité de détection en fonction du taux de compromission des objets, en utilisant le calibrage fait avec un logiciel différent, et pour 0%, 0,1% et 0,5% d'attaques. Rouge : TPR ≤50%. Orange : TPR ≤80% or FPR ≥1%.

		0%	*0.1%*		*0.5%*	
Software	Algorithm	FPR	FPR	TPR	FPR	TPR
S1	ODIN	0.23%	0.18%	72.08%	0.15%	79.82%
	LOF	0.18%	0.09%	78.13%	0.18%	89.43%
	MeanDIST α-med	0.07%	0.07%	77.08%	0.07%	90.32%
	MeanDIST HKF+	0.23%	0.22%	75.42%	0.20%	90.92%
	Isolation Forest	0.79%	0.72%	40.0%	0.34%	53.38%
	k-means	1.61%	0.07%	77.08%	1.58%	64.27%
	OCSVM	0.06%	0.00%	40.63%	0.00%	44.21%
S2	ODIN	0.10%	0.08%	79.38%	0.02%	89.03%
	LOF	0.07%	0.07%	87.50%	0.07%	96.07%
	MeanDIST α-med	0.06%	0.06%	87.08%	0.05%	98.64%
	MeanDIST HKF+	0.10%	0.10%	87.71%	0.10%	97.78%
	Isolation Forest	1.16%	1.01%	39.38%	0.73%	52.72%
	k-means	1.63%	1.64%	42.92%	1.56%	76.39%
	OCSVM	0.03%	0.36%	42.92%	0.00%	41.15%
S3	ODIN	0.03%	0.09%	81.67%	0.02%	93.69%
	LOF	0.04%	0.03%	84.17%	0.02%	94.24%
	MeanDIST α-med	1.02%	0.89%	72.71%	0.50%	100%
	MeanDIST HKF+	0.00%	0.01%	74.79%	0.00%	83.38%
	Isolation Forest	0.42%	0.31%	23.30%	0.09%	73.23%
	k-means	1.60%	1.51%	70.24%	1.22%	78.56%
	OCSVM	0.00%	*no alert*		*no alert*	
S4	ODIN	0.07%	0.07%	80.83%	0.06%	87.88%
	LOF	0.10%	0.10%	80.63%	0.10%	89.18%
	MeanDIST α-med	0.03%	0.03%	68.33%	0.03%	77.60%
	MeanDIST HKF+	0.09%	0.08%	72.50%	0.06%	79.30%
	Isolation Forest	0.08%	0.04%	23.96%	0.02%	49.16%
	k-means	0.00%	0.00%	47.08%	0.00%	53.81%
	OCSVM	0.03%	0.63%	49.17%	0.00%	46.77%

lement difficile à repérer, car peu impactante sur l'activité de l'objet, et la charge P7 est difficile à repérer uniquement dans les cas des logiciels légitimes surveillés S1 et S4. Cela peut s'expliquer par le fait que ces deux logiciels légitimes utilisent eux-mêmes les communications de manière assez intensives. Ainsi les demandes de connexion SSH par attaque de type force brute sont masquées par l'activité des communications légitimes. Enfin le tableau nous montre également que les résultats de LOF et de MeanDIST α-*med* sont globalement meilleurs et plus stables que ceux des deux autres algorithmes.

5.2.2.4 Analyse des performances et de passage à l'échelle

L'efficacité de détection de notre approche est bien sûr fondamentale, mais il est également important d'évaluer les ressources requises pour effectuer notre détection. Pour quantifier cet usage des ressources, nous allons dans cette sous-section étudier la quantité des données manipulées et la complexité en temps au pire cas

TABLE 5.2 – Sensibilité de détection face au taux de compromission des objets, en utilisant le calibrage fait avec un logiciel différent, et pour 1%, 2%, et 3% d'attaques. Rouge : TPR ≤50%. Orange : TPR ≤80% or FPR ≥1%.

		1%		*2%*		*5%*	
Software	Algorithm	FPR	TPR	FPR	TPR	FPR	TPR
S1	ODIN	0.10%	81.05%	0.06%	45.00%	0.03%	8.54%
	LOF	0.18%	88.78%	0.19%	86.33%	0.18%	33.37%
	MeanDIST α-med	0.07%	89.78%	0.07%	89.42%	0.06%	60.81%
	MeanDIST HKF+	0.15%	88.46%	0.13%	86.56%	0.02%	34.66%
	Isolation Forest	0.10%	58.18%	0.00%	55.81%	0.00%	28.72%
	k-means	1.55%	64.94%	1.50%	64.95%	1.50%	65.34%
	OCSVM	0.00%	36.15%	0.00%	7.80%	0.00%	1.82%
S2	ODIN	0.01%	77.48%	0.01%	44.01%	0.01%	8.46%
	LOF	0.07%	94.28%	0.07%	93.05%	0.08%	49.45%
	MeanDIST α-med	0.05%	98.98%	0.05%	98.59%	0.06%	78.28%
	MeanDIST HKF+	0.09%	96.79%	0.08%	88.27%	0.00%	55.29%
	Isolation Forest	0.18%	56.09%	0.01%	53.02%	0.00%	37.33%
	k-means	1.52%	76.06%	1.46%	75.70%	1.42%	75.79%
	OCSVM	0.00%	7.35%	0.00%	6.32%	0.00%	1.72%
S3	ODIN	0.01%	83.64%	0.01%	45.81%	0.00%	14.37%
	LOF	0.02%	90.64%	0.01%	70.22%	0.01%	42.33%
	MeanDIST α-med	0.43%	98.90%	0.28%	98.71%	0.27%	79.81%
	MeanDIST HKF+	0.00%	77.49%	0.00%	75.49%	0.00%	45.21%
	Isolation Forest	0.02%	74.98%	0.00%	75.00%	0.00%	56.20%
	k-means	1.08%	78.04%	0.94%	76.62%	0.91%	75.58%
	OCSVM	*no alert*		*no alert*		*no alert*	
S4	ODIN	0.07%	86.86%	0.06%	49.92%	0.04%	6.30%
	LOF	0.10%	90.62%	0.10%	89.91%	0.09%	49.44%
	MeanDIST α-med	0.03%	78.19%	0.03%	76.32%	0.03%	45.97%
	MeanDIST HKF+	0.05%	75.77%	0.04%	73.91%	0.01%	33.31%
	Isolation Forest	0.00%	51.33%	0.00%	52.73%	0.00%	43.05%
	k-means	0.00%	53.60%	0.00%	53.76%	0.00%	53.54%
	OCSVM	0.00%	35.38%	0.00%	6.25%	0.00%	1.16%

des algorithmes de détection.

Nous discutons ici de ces différentes ressources. Tout d'abord, en ce qui concerne la quantité de données stockées et envoyées sur le réseau, chaque fichier de trace contient Δ_s/Δ_p valeurs par compteur pour un total de $|C|(\Delta_s/\Delta_p)$ valeurs au total. En considérant le cas où les compteurs font 64 bits, un encodage binaire entraînerait une taille de fichier de $8|C|(\Delta_s/\Delta_p)$ octets. Ainsi, pour une population de $|D|$ objets, le serveur reçoit $8|C||D|(\Delta_s/\Delta_p)$ octets à analyser tous les Δ_s. Dans notre expérience en particulier, nous avons : $|C| = 7$,$|D| = 10,000$, $\Delta_p = 5s$ et $\Delta_s = 1,800s$. Ainsi le serveur reçoit et analyse près de 200MB toutes les 30 minutes dans le cas de nos expériences. Cette quantité de données est tout à fait raisonnable si on considère le nombre élevé d'objets considérés.

En ce qui concerne les durées d'exécution, tous les algorithmes de détection de valeurs aberrantes que nous avons testés produisent leurs résultats en moins de 15 secondes, sachant que la machine que nous avons utilisée pour le serveur est un

TABLE 5.3 – efficacité de détection avec 1% d'objets compromis par charge malveillante. Rouge : TPR ≤50%. Orange : TPR ≤80%.

Logiciel	Algorithme	Precision	FPR	TPR P1	TPR P2	TPR P3
S1	ODIN	88.59%	0.10%	100%	100%	100%
	LOF	83.06%	0.18%	100%	100%	100%
	MeanDIST α-med	93.07%	0.07%	100%	100%	100%
	MeanDIST HKF+	87.01%	0.15%	100%	100%	100%
S2	ODIN	99.52%	0.01%	99.80%	99.74%	100%
	LOF	93.06%	0.07%	100%	100%	100%
	MeanDIST α-med	95.13%	0.05%	100%	100%	100%
	MeanDIST HKF+	91.80%	0.09%	100%	100%	100%
S3	ODIN	98.29%	0.01%	99.29%	66.67%	100%
	LOF	98.29%	0.02%	99.29%	100%	100%
	MeanDIST α-med	77.06%	0.43%	99.33%	100%	100%
	MeanDIST HKF+	100%	0%	0.56%	100%	100%
S4	ODIN	92.80%	0.07%	98.61%	100%	100%
	LOF	89.77%	0.10%	100%	100%	100%
	MeanDIST α-med	95.89%	0.03%	100%	100%	100%
	MeanDIST HKF+	95.16%	0.05%	93.05%	100%	100%

TABLE 5.4 – efficacité de détection avec 1% d'objets compromis par charge malveillante. Rouge : TPR ≤50%. Orange : TPR ≤80%.

Logiciel	Algorithme	TPR P4	TPR P5	TPR P6	TPR P7	TPR P8
S1	ODIN	14.50%	100%	100%	33.93%	100%
	LOF	79.14%	100%	100%	31.06%	100%
	MeanDIST α-med	87.28%	100%	100%	30.92%	100%
	MeanDIST HKF+	76.40%	100%	100%	31.28%	100%
S2	ODIN	10.52%	100%	99.09%	100%	99.70%
	LOF	54.21%	100%	100%	100%	100%
	MeanDIST α-med	91.83%	100%	100%	100%	100%
	MeanDIST HKF+	74.92%	100%	99.39%	100%	100%
S3	ODIN	63.09%	66.67%	99.27%	92.63%	81.53%
	LOF	25.85%	100%	100%	100%	100%
	MeanDIST α-med	91.83%	100%	100%	100%	100%
	MeanDIST HKF+	19.93%	100%	99.39%	100%	100%
S4	ODIN	53.61%	93.75%	100%	48.88%	100%
	LOF	96.18%	100%	100%	28.78%	100%
	MeanDIST α-med	36.58%	63.09%	100%	25.87%	100%
	MeanDIST HKF+	34.15%	53.01%	99.79%	26.15%	100%

ordinateur portable doté d'un processeur Intel Core i7-3687U CPU @ 3.3GHz avec 8Go de RAM. Ces algorithmes travaillent sur des caractéristiques qu'il faut d'abord déterminer. Les phases d'extraction et de standardisation de ces caractéristiques ont demandé au maximum 5 minutes de calcul (à l'aide d'un code python qui peut être optimisé) dans nos expérimentations et en sauvegardant les valeurs intermédiaires (évolution et valeurs non-standardisées). Cependant, il est important de souligner que ces algorithmes sont effectués en réalité sur des serveurs qui peuvent être bien plus puissants que la machine que nous avons utilisée pour les expérimentations. On peut donc s'attendre à des temps de détection très satisfaisants au final.

On peut estimer l'efficacité temporelle des algorithmes de détection de valeurs aberrantes vis-à-vis de la mise à l'échelle des données en utilisant leur complexité temporelle. Nous avons ainsi indiqué cette complexité temporelle dans le tableau 5.5. OCSVM a de loin la plus grande complexité : traiter dix fois plus de données prend environ mille fois plus de temps. Ceci signifie qu'OCSVM ne pourrait probablement pas traiter 100 000 données en trente minutes. À part OCSVM, on peut s'attendre à ce que tous les autres algorithmes s'exécutent en moins de 5 minutes avec 100 000 données en entrées. Notons également que *Isolation Forest* et *k-means* ont les meilleurs complexités et pourraient probablement s'exécuter en moins de 30 minutes avec 10 000 000 données. En effet, le temps d'exécution de ces deux algorithmes est proportionnel à la taille de la population étudiée n. On remarque aussi que les algorithmes basés sur les plus proches voisins, que nous avons identifiés comme étant les plus performants, ont une complexité temporelle tout à fait raisonnable. Ainsi en passant la population de 10 000 à 100 000, nous aboutissons à des calculs près de 30 fois plus longs (en se rappelant que nous utilisant l'heuristique $k \propto \sqrt{n}$). Enfin HDBSCAN présente une complexité intéressante mais pour des résultats de détection insatisfaisants.

TABLE 5.5 – Complexités des différents algorithmes. n le nombre de traces, k le nombre de plus proche voisins.

ODIN	LOF	MeanDIST	KDIST
$\mathcal{O}(kn\log(n))$	$\mathcal{O}(kn\log(n))$	$\mathcal{O}(kn\log(n))$	$\mathcal{O}(kn\log(n))$
Isolation Forest	k-means	OCSVM	HDBSCAN
$\mathcal{O}(n)$	$\mathcal{O}(kn)$	$\mathcal{O}(n^3)$	$\mathcal{O}(n\log(n))$

5.2.3 Conclusion et discussions

Grâce à l'analyse de nos résultats, nous pouvons maintenant répondre explicitement aux questions qui nous ont mené à l'élaboration de ces expérimentations.

- *Quels sont les algorithmes les plus efficaces ?*
  Nos expérimentations montrent que les algorithmes ODIN, LOF et MeanDIST semblent les plus efficaces compte tenu de nos données. Ce sont eux qui fournissent la meilleure détection avec un faible taux de faux positifs.
- *Quel est l'impact de la proportion d'attaque μ sur l'efficacité des algorithmes ?*
  Nos expérimentations montrent que l'efficacité de détection des algorithmes se dégrade significativement lorsque plus de 5% d'objets sont compromis. Mais comme expliqué précédemment, notre approche se base sur le fait qu'une telle quantité d'objets ne peut être compromise instantanément. Nous utilisons des algorithmes de détection de valeurs aberrantes et mettons en forme nos données en prenant ce facteur en compte. Donc ce résultat était attendu. La détection doit se faire au plus tôt et elle est très efficace dès le premier pourcent d'objets corrompus.

- *Comment un algorithme calibré avec un logiciel performe sur un autre logiciel ?* Tous nos résultats ont été obtenus avec des algorithmes optimisés au niveau des paramètres pour détecter des valeurs aberrantes d'un logiciel différent du logiciel surveillé dans nos expériences. Les résultats obtenus sont probants. De plus, nous avons fait une étude de stabilité en comparant les résultats de détection lorsque les algorithmes sont optimisés avec des valeurs du logiciel surveillé final et avec un logiciel différent. Nous avons ainsi déterminé les algorithmes les plus stables, c'est-à-dire les algorithmes qui sont le moins affectés dans leur détection par des potentielles mises à jour. Les meilleurs algorithmes sont ainsi ODIN, LOF et MeanDIST α-*med*.
- *Quel est l'impact de la mise à l'échelle sur de tels algorithmes ?* Concernant la mise à l'échelle, nous avons constaté que tous les algorithmes sont utilisables avec les 10 000 données, et s'exécutent avec un temps tout à fait acceptable au vu du contexte. Avec une étude de complexité et en gardant la même configuration, nous sommes également confiants dans l'idée que nous pouvons aisément augmenter à 100 000 le nombre de traces et donc d'objets à analyser.

Cependant, il reste encore des points de discussion importants concernant certains aspects de notre approche.

5.2.3.1 La représentation du comportement du processeur

L'hypothèse de base que nous faisons avec notre approche est que les compteurs de performance reflètent le comportement du processeur, et donc indirectement le comportement du logiciel s'exécutant sur celui-ci. Cependant, les logiciels s'exécutant sur les processeurs peuvent être divers et complexes, et le comportement d'un processeur vis-à-vis d'un logiciel est très dépendant de ce processeur. Il est donc compliqué de reporter directement les résultats d'un processeur à l'autre. Nous pensons que ces faits représentent des faiblesses des autres recherches menées utilisant les compteurs dans le cadre de la sécurité. Nous insistons donc sur le fait que notre approche concerne bien des objets simples, pour que les compteurs et le comportement du processeur soient le plus directement liés au comportement du logiciel surveillé. De plus, tous les processeurs déployés dans notre approche sont identiques. Ainsi, il n'y a pas de différence d'interprétation des résultats entre les objets étudiés, et nous sommes donc sûrs que tous les compteurs ont bien été implémentés de la même façon. Comme ces compteurs sont implémentés différemment en fonction du processeur et que nous voulons une approche qui puisse être généralisée à d'autres processeurs, nous avons jugé plus opportun de mener des expérimentations sur divers profils d'application ainsi que différentes charges malveillantes, ainsi que sur le pourcentage d'objets compromis. Malgré tout, la représentation du comportement de l'objet reste partielle, puisque nous étudions uniquement le comportement du processeur sans considérer les différents autres composants matériels de l'objet. En particulier, les valeurs des compteurs de performance ne nous fournissent qu'une représentation indirecte de l'utilisation des ressources telles que le réseau

ou la RAM. Par exemple, nous savons que l'envoi de message va provoquer une exception du processeur. Aussi une utilisation intense du réseau se répercute sur le compteur d'exceptions, mais ce n'est pas le seul moyen de générer des exceptions, et par conséquent l'observation du compteur d'exceptions ne donne pas une mesure exacte de l'utilisation du réseau. Un moyen de palier à cette limitation serait d'ajouter des dispositifs matériels nous permettant d'obtenir plus d'informations sur l'objet, mais cela serait contradictoire avec l'objectif d'une solution ne nécessitant pas de modification matérielle.

5.2.3.2 L'efficacité des différents algorithmes de détection de valeurs aberrantes

Les conclusions sur les modèles d'apprentissage automatique que nous avons utilisés dépendent fondamentalement des hypothèses que nous avons identifiées au début de cette thèse. Ceci étant dit, nous pouvons émettre des pronostics sur l'extension de ces résultats à d'autres contextes. Les faibles performances de HDBSCAN et la complexité temporelle de OCSVM nous conduisent à conclure que ces deux algorithmes ne sont pas adaptés au problème de la détection de valeurs aberrantes dans une large population. L'algorithme k-means fonctionne correctement en connaissant le nombre de groupes légitimes (son paramètre k). Nous avons fait l'hypothèse (**H2**) d'une population légitime homogène, aussi avons-nous choisi $k = 1$. Pour rappel, cette hypothèse n'est pas aussi forte qu'il n'y paraît car il semble raisonnable d'imaginer que le serveur puisse connaître le numéro de version des logiciels tournant sur les objets et donc qu'il puisse regrouper les traces qu'il reçoit en des groupes au comportement homogène. En l'absence de cette hypothèse, l'algorithme k-means serait plus difficile à utiliser. Du fait même qu'ils s'appuient sur des informations locales, les algorithmes basés sur les plus proches voisins, qui ont les meilleures performances de détection, devraient conserver des performances satisfaisantes en présence de plusieurs clusters légitimes à condition qu'ils soient suffisamment grands. Néanmoins, de plus amples expériences seraient nécessaires pour étudier les performances de ces algorithmes dans une plus grande variété de situations, comme faire varier le nombre de profils légitimes, leur diversité, autoriser plusieurs comportements légitimes différents dans une même population.

5.3 Conclusion

Dans ce chapitre, nous avons décrit les expérimentations qui nous permis de développer et valider notre approche. Les résultats obtenus montrent globalement la pertinence de l'utilisation des compteurs matériels dans le cadre d'objets simples et identiques. L'analyse expérimentale a notamment permis d'identifier 3 algorithmes qui sont plus particulièrement efficaces dans ce contexte. Les résultats en terme d'efficacité de la détection, en terme de performances et de capacité de passage à l'échelle montrent la pertinence de l'approche pour un déploiement opérationnel. Il reste cependant à confirmer la validité de ces résultats dans un environnement

industriel complexe, dans différents scénarios.

Chapitre 6

Conclusion

6.1 Bilan

Dans cette thèse nous nous sommes intéressés à la détection d'intrusions dans le cas d'objets connectés. Nous avons pour cela dans un premier temps présenté les objets connectés, certaines de leurs particularités ainsi que les attaques dont ils sont aujourd'hui régulièrement la cible. Cependant le monde de l'Internet des objets est vaste, et nous avions, dans nos travaux, un objectif plus précis lié au contexte applicatif de cette thèse menée en partenariat avec EDF : la détection d'intrusions dans un contexte industriel. Nous avons donc présenté plus en détails ce domaine particulier de l'Internet des objets ainsi qu'un panorama des attaques ciblant ce type d'objets. Plus particulièrement, nous nous sommes intéressés à des objets déployés dans le contexte de la gestion de l'énergie avec des spécificités particulières que nous avons exposées. Nous avons ainsi décrit explicitement les objectifs et hypothèses qui ont guidé ces travaux. En particulier, nous souhaitions proposer une approche légère, ne nécessitant pas de modifications des logiciels ou des systèmes d'exploitation s'exécutant sur les objets, et qui puisse rester efficace lors de mises à jour des objets, autrement dit, dont la détection ne dépend pas du logiciel installé. Nous nous sommes donc données pour objectif de ne pas avoir à élaborer de modèle du comportement de l'application légitime pour pouvoir détecter des intrusions. Une hypothèse importante de notre travail également est que les objets connectés sont tous identiques d'un point de vue matériel et logiciel et qu'ils sont massivement déployés.

Afin de mieux comprendre le modèle de menaces spécifiques visant ces objets, nous avons tout d'abord effectué un travail préliminaire comprenant une étude EBIOS d'un objet connecté représentatif du contexte ciblé par nos travaux. Cette étude nous a ainsi permis de mettre en évidence un certain nombre de vulnérabilités et de scénarios de menaces qu'il est nécessaire de prendre en compte dans nos travaux.

Après cet état des lieux de la menace et des hypothèses des attaquants, nous avons proposé un état de l'art des solutions de la littérature qui nous semblaient de bonnes candidates pour notre problématique. Parmi ces travaux, la détection d'intrusions est la solution qui nous a semblé la plus pertinente et pour laquelle nous avons donc choisi de proposer une contribution. Cette contribution est ainsi détaillée dans le chapitre 4. Elle se base sur la collecte et l'analyse de compteurs de performances des processeurs, afin de détecter des valeurs aberrantes, caractérisant des objets corrompus au sein d'une flotte massive d'objets sains. Cette approche,

même si elle a essuyé quelques critiques dans des contextes différents du nôtre, nous a semblé pertinente dans notre contexte spécifique, où les objets similaires du point de vue logiciel et matériel sont massivement déployés de façon identique. Les principales étapes de notre approche sont présentées dans ce chapitre : collecte des compteurs, extraction de caractéristiques pertinentes, standardisation, application d'algorithmes de détection de valeurs aberrantes.

Enfin, dans le dernier chapitre de ce manuscrit, nous décrivons deux expérimentations qui ont été réalisées dans le but de valider notre approche. La première était une expérimentation plus légère et exploratoire que la seconde, dans laquelle nous voulions nous assurer que cette approche était bien pertinente, compte tenu des différentes critiques dont elle faisait l'objet par ailleurs. La seconde expérimentation a été menée à plus grande échelle, incluant toutes les étapes de notre approche, et nous a permis non seulement de confirmer la pertinence de notre approche, mais aussi de comparer différents algorithmes de détection de valeurs aberrantes afin de déterminer ceux qui donnaient les meilleurs résultats dans notre contexte.

Au final, nous avons pu proposer dans nos travaux une approche originale permettant d'effectuer la détection d'anomalies à distance pour une large flotte d'objets connectés identiques, qui est à la fois légère et qui ne nécessite pas de modélisation complexe des logiciels mis en œuvre sur les objets légitimes. De plus cette approche a été validée avec des expérimentations qui ont permis de faire des tests à l'échelle (simulation de 10 000 objets), avec différents profils de programmes légitimes et de charges malveillantes, tout en comparant plusieurs méthodes de détection de valeurs aberrantes.

6.2 Perspectives

Nous avons noté plusieurs points de notre approche qui pourraient mériter de plus amples recherches. Un des points importants est le choix des caractéristiques. En effet, nous avons choisi dans notre cas 7 statistiques descriptives pour représenter les séries temporelles des compteurs. Comme ces statistiques sont simples à calculer et suffisantes pour la détection, nous n'avons pas exploré d'autres possibilités pour le moment. Cependant il pourrait être intéressant d'essayer d'autres caractéristiques permettant de représenter les séries temporelles, qui pourraient mettre en lumière certains aspects qui ne sont pas exploités dans notre approche actuelle. Il faut ensuite bien sûr évaluer la complexité de calcul que ces nouvelles caractéristiques peuvent ajouter, car nous considérons un grand nombre d'objets déployés en même temps. On peut citer quelques exemples de caractéristiques qu'il serait intéressant de calculer à partir des traces : l'autocorrélation, les transformées de Fourier, la régression polynomiale, etc.

Un autre point intéressant de notre approche concerne l'utilisation de plusieurs algorithmes de détection de valeurs aberrantes. En effet, un type d'attaque peut être mieux détecté par un algorithme que par un autre, etc. Il existe également les algorithmes qui sont plus efficaces avec peu d'attaques (1% par exemple) et

d'autres qui sont plus robustes avec plus d'attaques (10% par exemple). Ce genre de techniques existent en apprentissage supervisé (sous l'appellation "apprentissage ensembliste") mais est bien moins développé pour l'apprentissage non-supervisé (sans apprentissage de modèle). Par exemple, nous pouvons remarquer que l'algorithme k-means a des performances qui paraissent indépendantes de la proportion d'attaques. Nous pourrions donc imaginer un détecteur hybride qui s'appuie sur un algorithme efficace avec peu d'attaques (LOF par exemple) et, quand il détecte beaucoup d'attaques (plus de 2% par exemple), utiliser également k-means. Il est également possible d'utiliser plusieurs algorithmes en parallèle et de procéder à un vote afin de tirer parti de la complémentarité de certaines approches. En revanche, ces méthodes augmentent significativement les calculs nécessaires à la détection et donc, si on les adopte, il est nécessaire de chercher un compromis entre l'efficacité de la détection et les performances requises pour cette détection.

Enfin, nous n'avons jusque là travaillé qu'avec des clichés instantanés de la population sans prendre en compte son évolution. Cet aspect temporel pourrait intervenir à plusieurs niveaux. Tout d'abord, peut-être que des attaques pourraient être détectées en observant l'évolution du comportement d'un objet au cours du temps. La technique de détection resterait la même, mais au lieu de travailler sur une population de différents objets à un même instant, on pourrait travailler avec une "population" composée d'un seul objet à différents instants (par exemple par pas de 30 mn). Cela pourrait permettre d'identifier des objets qui changent subitement de comportement. Une autre possibilité est de travailler avec des populations d'objets qui incluent plusieurs clichés dans une même grande population. Par exemple, si nous travaillons avec une population d'objets sur vingt pas de temps (c'est-à-dire que pour chaque objet, il y aurait dans la population vingt points lui correspondant), alors même si la population est à un instant t entièrement compromise, nous n'aurions pour cette population élargie que 5% d'attaque (19 clichés sans attaques et 1 cliché rempli d'attaques). Ceci est donc une manière supplémentaire de s'assurer d'une borne maximale sur la proportion d'attaque simultanée, ce qui lèverait la principale limitation de notre méthode.

Bibliographie

[Al-Karaki 2004] Jamal N Al-Karaki et Ahmed E Kamal. *Routing techniques in wireless sensor networks : a survey.* IEEE wireless communications, vol. 11, no. 6, pages 6–28, 2004. (Cité en page 6.)

[Ambrosin 2020] Moreno Ambrosin, Mauro Conti, Riccardo Lazzeretti, Md Masoom Rabbani et Silvio Ranise. *Collective Remote Attestation at the Internet of Things Scale : State-of-the-Art and Future Challenges.* IEEE Communications Surveys & Tutorials, vol. 22, no. 4, pages 2447–2461, 2020. (Cité en page 46.)

[Amer 2013] Mennatallah Amer, Markus Goldstein et Slim Abdennadher. *Enhancing one-class support vector machines for unsupervised anomaly detection.* Dans Proceedings of the ACM SIGKDD workshop on outlier detection and description, pages 8–15, 2013. (Cité en page 79.)

[Ara 2017] Anderson Ara, Francisco Louzada et Carlos AR Diniz. *Statistical monitoring of a web server for error rates : a bivariate time-series copula-based modeling approach.* Journal of Applied Statistics, vol. 44, no. 13, pages 2287–2300, 2017. (Cité en page 56.)

[Asokan 2015] Nadarajah Asokan, Ferdinand Brasser, Ahmad Ibrahim, Ahmad-Reza Sadeghi, Matthias Schunter, Gene Tsudik et Christian Wachsmann. *Seda : Scalable embedded device attestation.* Dans Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pages 964–975, 2015. (Cité en page 46.)

[Avizienis 1977] Algirdas Avizienis, Liming Chen*et al. On the implementation of N-version programming for software fault-tolerance during program execution.* 1977. (Cité en page 47.)

[Avizienis 2004] Algirdas Avizienis, J.-C Laprie, Brian Randell et Carl Landwehr. *Basic Concepts and Taxonomy of Dependable and Secure Computing.* Dependable and Secure Computing, IEEE Transactions on, vol. 1, pages 11 – 33, 02 2004. (Cité en page 9.)

[Ayadi 2017] Aya Ayadi, Oussama Ghorbel, Abdulfattah M Obeid et Mohamed Abid. *Outlier detection approaches for wireless sensor networks : A survey.* Computer Networks, vol. 129, pages 319–333, 2017. (Cité en page 60.)

[Bahador 2014] Mohammad Bagher Bahador, Mahdi Abadi et Asghar Tajoddin. *HPCMalHunter : Behavioral malware detection using hardware performance counters and singular value decomposition.* Dans 2014 4th International Conference on Computer and Knowledge Engineering (ICCKE), pages 703–708. IEEE, 2014. (Cité en page 65.)

[Bahador 2019] Mohammad Bagher Bahador, Mahdi Abadi et Asghar Tajoddin. *HLMD : a signature-based approach to hardware-level behavioral malware*

detection and classification. The Journal of Supercomputing, pages 1–32, 2019. (Cité en page 65.)

[Bauer 2017] Harald Bauer, Ondrej Burkacky et Christian Knochenhauer. *Security in the Internet of Things.* Semiconductor, McKinsey & Company : New York, NY, USA, 2017. (Cité en page 15.)

[Bourdon] Malcolm Bourdon, Eric Alata, Mohamed Kaaniche, Vincent Migliore, Vincent Nicomette et Youssef Laarouchi. *Anomaly detection using hardware performance counters on a large scale deployment.* (Cité en page 3.)

[Bourdon 2020] Malcolm Bourdon, Pierre-François Gimenez, Eric Alata, Mohamed Kaaniche, Vincent Migliore, Vincent Nicomette et Youssef Laarouchi. *Hardware-Performance-Counters-based anomaly detection in massively deployed smart industrial devices.* Dans 2020 IEEE 19th International Symposium on Network Computing and Applications (NCA), pages 1–8. IEEE, 2020. (Cité en page 3.)

[Breunig 2000] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng et Jörg Sander. *LOF : Identifying Density-Based Local Outliers.* SIGMOD Rec., vol. 29, no. 2, page 93–104, mai 2000. (Cité en page 78.)

[Cherepanov 2017] Aqnton Cherepanov et Robert Lipovsky. *Industroyer : Biggest threat to industrial control systems since Stuxnet.* WeLiveSecurity, ESET, vol. 12, 2017. (Cité en page 17.)

[Chiba 2019] Zouhair Chiba, Noreddine Abghour, Khalid Moussaid, Mohamed Rida*et al. Intelligent approach to build a Deep Neural Network based IDS for cloud environment using combination of machine learning algorithms.* Computers & Security, vol. 86, pages 291–317, 2019. (Cité en page 51.)

[Collberg 2012] Christian Collberg, Sam Martin, Jonathan Myers et Jasvir Nagra. *Distributed application tamper detection via continuous software updates.* Dans Proceedings of the 28th Annual Computer Security Applications Conference, pages 319–328, 2012. (Cité en page 48.)

[Damien 2020] Aliénor Damien, Pierre-François Gimenez, Nathalie Feyt, Vincent Nicomette, Mohamed Kaâniche et Eric Alata. *On-board Diagnosis : A First Step from Detection to Prevention of Intrusions on Avionics Applications.* Dans 2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE), pages 358–368. IEEE, 2020. (Cité en page 59.)

[Das 2019] Sanjeev Das, Jan Werner, Manos Antonakakis, Michalis Polychronakis et Fabian Monrose. *SoK : The challenges, pitfalls, and perils of using hardware performance counters for security.* Dans Proceedings of 40th IEEE Symposium on Security and Privacy, 2019. (Cité en page 69.)

[Demme 2013] John Demme, Matthew Maycock, Jared Schmitz, Adrian Tang, Adam Waksman, Simha Sethumadhavan et Salvatore Stolfo. *On the feasibility of online malware detection with performance counters.* ACM SIGARCH Computer Architecture News, vol. 41, no. 3, pages 559–570, 2013. (Cité en pages 65 et 66.)

[Di Pinto 2018] Alessandro Di Pinto, Younes Dragoni et Andrea Carcano. *TRITON : The first ICS cyber attack on safety instrument systems.* Dans Proc. Black Hat USA, pages 1–26, 2018. (Cité en page 17.)

[Eldefrawy 2012] Karim Eldefrawy, Gene Tsudik, Aurélien Francillon et Daniele Perito. *SMART : Secure and Minimal Architecture for (Establishing Dynamic) Root of Trust.* Dans Ndss, volume 12, pages 1–15, 2012. (Cité en page 46.)

[ENISA 2017] EU ENISA. *Baseline Security Recommendations for IoT in the context of Critical Information Infrastructures*, 2017. (Cité en pages v, 7 et 8.)

[Ertoz 2004] Levent Ertoz, Eric Eilertson, Aleksandar Lazarevic, Pang-Ning Tan, Vipin Kumar, Jaideep Srivastava et Paul Dokas. *Minds-minnesota intrusion detection system.* Next generation data mining, pages 199–218, 2004. (Cité en page 60.)

[Forrest 1996] Stephanie Forrest, Steven A Hofmeyr, Anil Somayaji et Thomas A Longstaff. *A sense of self for unix processes.* Dans Proceedings 1996 IEEE Symposium on Security and Privacy, pages 120–128. IEEE, 1996. (Cité en page 59.)

[Gassend 2002] Blaise Gassend, Dwaine Clarke, Marten Van Dijk et Srinivas Devadas. *Silicon physical random functions.* Dans Proceedings of the 9th ACM Conference on Computer and Communications Security, pages 148–160, 2002. (Cité en page 43.)

[Halak 2018] Basel Halak. Physically unclonable functions : From basic design principles to advanced hardware security applications. Springer, 2018. (Cité en page 44.)

[Hautamaki 2004] Ville Hautamaki, Ismo Karkkainen et Pasi Franti. *Outlier detection using k-nearest neighbour graph.* Dans Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004., volume 3, pages 430–433. IEEE, 2004. (Cité en pages 79 et 99.)

[Holcomb 2007] Daniel E Holcomb, Wayne P Burleson, Kevin Fu*et al. Initial SRAM state as a fingerprint and source of true random numbers for RFID tags.* Dans Proceedings of the Conference on RFID Security, volume 7, page 01, 2007. (Cité en page 43.)

[Jackson 2013] Todd Jackson, Andrei Homescu, Stephen Crane, Per Larsen, Stefan Brunthaler et Michael Franz. *Diversifying the software stack using randomized NOP insertion.* Dans Moving Target Defense II, pages 151–173. Springer, 2013. (Cité en page 47.)

[Jangda 2015] Abhinav Jangda, Mohit Mishra et Bjorn De Sutter. *Adaptive just-in-time code diversification.* Dans Proceedings of the Second ACM Workshop on Moving Target Defense, pages 49–53, 2015. (Cité en page 48.)

[Koc 2012] Levent Koc, Thomas A Mazzuchi et Shahram Sarkani. *A network intrusion detection system based on a Hidden Naïve Bayes multiclass classifier.*

Expert Systems with Applications, vol. 39, no. 18, pages 13492–13500, 2012. (Cité en page 58.)

[Kolias 2017] Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou et Jeffrey Voas. *DDoS in the IoT : Mirai and other botnets.* Computer, vol. 50, no. 7, pages 80–84, 2017. (Cité en page 13.)

[Kong 2014] Joonho Kong, Farinaz Koushanfar, Praveen K Pendyala, Ahmad-Reza Sadeghi et Christian Wachsmann. *PUFatt : Embedded platform attestation based on novel processor-based PUFs.* Dans 2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC), pages 1–6. IEEE, 2014. (Cité en page 45.)

[Koroniotis 2019] Nickolaos Koroniotis, Nour Moustafa, Elena Sitnikova et Benjamin Turnbull. *Towards the development of realistic botnet dataset in the internet of things for network forensic analytics : Bot-iot dataset.* Future Generation Computer Systems, vol. 100, pages 779–796, 2019. (Cité en page 58.)

[Krishnamurthy 2019] Prashanth Krishnamurthy, Ramesh Karri et Farshad Khorrami. *Anomaly detection in real-time multi-threaded processes using hardware performance counters.* IEEE Transactions on Information Forensics and Security, vol. 15, pages 666–680, 2019. (Cité en pages 67 et 73.)

[Kumar 2007] Sailesh Kumar. *Survey of current network intrusion detection techniques.* Washington Univ. in St. Louis, pages 1–18, 2007. (Cité en page 51.)

[Langner 2011] Ralph Langner. *Stuxnet : Dissecting a cyberwarfare weapon.* IEEE Security & Privacy, vol. 9, no. 3, pages 49–51, 2011. (Cité en page 14.)

[Laprie 1996] Jean-Claude Laprie, Jean Arlat, Jean-Paul Blanquart, Alain Costes, Yves Crouzet, Yves Deswarte, Jean-Charles Fabre, Hubert Guillermain, Mohamed Kaâniche, Karama Kanoun, Corinne Mazet, David Powell, Christophe Rabéjac et Pascale Thévenod. Guide de la sûreté de fonctionnement. Cépaduès, Toulouse (France), 1996. OCLC : 35123685. (Cité en page 9.)

[Larsen 2014] Per Larsen, Andrei Homescu, Stefan Brunthaler et Michael Franz. *SoK : Automated software diversity.* Dans 2014 IEEE Symposium on Security and Privacy, pages 276–291. IEEE, 2014. (Cité en pages v, 47 et 49.)

[Lim 2005] Daihyun Lim, Jae W Lee, Blaise Gassend, G Edward Suh, Marten Van Dijk et Srinivas Devadas. *Extracting secret keys from integrated circuits.* IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 13, no. 10, pages 1200–1205, 2005. (Cité en page 43.)

[Liu 2008] Fei Tony Liu, Kai Ming Ting et Zhi-Hua Zhou. *Isolation forest.* Dans 2008 Eighth IEEE International Conference on Data Mining, pages 413–422. IEEE, 2008. (Cité en page 79.)

[Maes 2010] Roel Maes et Ingrid Verbauwhede. *Physically unclonable functions : A study on the state of the art and future research directions.* Dans Towards Hardware-Intrinsic Security, pages 3–37. Springer, 2010. (Cité en pages v, 43 et 44.)

[McInnes 2017a] Leland McInnes et John Healy. *Accelerated hierarchical density based clustering.* Dans 2017 IEEE International Conference on Data Mining Workshops (ICDMW), pages 33–42. IEEE, 2017. (Cité en page 81.)

[McInnes 2017b] Leland McInnes, John Healy et Steve Astels. *hdbscan : Hierarchical density based clustering.* Journal of Open Source Software, vol. 2, no. 11, page 205, 2017. (Cité en page 81.)

[McQuade 2018] MIKE McQuade. *The untold story of NotPetya, the most devastating cyberattack in history*, 2018. (Cité en page 17.)

[Miller 2015] Charlie Miller et Chris Valasek. *Remote exploitation of an unaltered passenger vehicle.* Black Hat USA, vol. 2015, page 91, 2015. (Cité en page 13.)

[Mucci 1999] Philip J Mucci, Shirley Browne, Christine Deane et George Ho. *PAPI : A portable interface to hardware performance counters.* Dans Proceedings of the department of defense HPCMP users group conference, volume 710. Citeseer, 1999. (Cité en page 64.)

[Nicomette 2011] V. Nicomette, M. Kaaniche, E. Alata et M. Herrb. *Set-up and deployment of a high-interaction honeypot : experiment and lessons learned.* Journal in computer virology, vol. 7, no. 2, page 143, 2011. (Cité en pages 96 et 97.)

[Pappas 2013] Vasilis Pappas, Michalis Polychronakis et Angelos D Keromytis. *Practical software diversification using in-place code randomization.* Dans Moving Target Defense II, pages 175–202. Springer, 2013. (Cité en page 48.)

[Pedregosa 2011] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg*et al. Scikit-learn : Machine learning in Python.* the Journal of machine Learning research, vol. 12, pages 2825–2830, 2011. (Cité en page 90.)

[Perito 2010] Daniele Perito et Gene Tsudik. *Secure code update for embedded devices via proofs of secure erasure.* Dans European Symposium on Research in Computer Security, pages 643–662. Springer, 2010. (Cité en pages 46 et 71.)

[Powell 2001] David Powell, Robert Stroud (editors, Sadie Creese (qinetiq, Yves Deswarte (laas cnrs, Klause Kursawe (ibm Zrl, Jean claude Laprie (laas cnrs, David Powell (laas cnrs et James Riordan (ibm Zrl. *Malicious- and accidental-fault tolerance for internet applications : Conceptual model and architecture.* Conceptual model and architecture, 2001. (Cité en page 11.)

[Prada 2019] Iván Prada, Francisco D Igual et Katzalin Olcoz. *Detecting time-fragmented cache attacks against AES using Performance Monitoring Counters.* arXiv preprint arXiv :1904.11268, 2019. (Cité en page 66.)

[Premaratne 2009] Upeka Premaratne, Charles Ling, Jagath Samarabandu et Tarlochan Sidhu. *Possibilistic decision trees for intrusion detection in iec61850*

automated substations. Dans 2009 International Conference on Industrial and Information Systems (ICIIS), pages 204–209. IEEE, 2009. (Cité en page 58.)

[Ramaswamy 2000] Sridhar Ramaswamy, Rajeev Rastogi et Kyuseok Shim. *Efficient algorithms for mining outliers from large data sets*. Dans Proceedings of the 2000 ACM SIGMOD international conference on Management of data, pages 427–438, 2000. (Cité en page 77.)

[Ravi 2020] Nagarathna Ravi et S Mercy Shalinie. *Semisupervised-Learning-Based Security to Detect and Mitigate Intrusions in IoT Network*. IEEE Internet of Things Journal, vol. 7, no. 11, pages 11041–11052, 2020. (Cité en page 59.)

[Rousseeuw 1993] Peter J Rousseeuw et Christophe Croux. *Alternatives to the median absolute deviation*. Journal of the American Statistical association, vol. 88, no. 424, pages 1273–1283, 1993. (Cité en page 74.)

[Sekar 2002] Ramasubramanian Sekar, Ajay Gupta, James Frullo, Tushar Shanbhag, Abhishek Tiwari, Henglin Yang et Sheng Zhou. *Specification-based anomaly detection : a new approach for detecting network intrusions*. Dans Proceedings of the 9th ACM conference on Computer and communications security, pages 265–274, 2002. (Cité en page 55.)

[Seshadri 2004] Arvind Seshadri, Adrian Perrig, Leendert Van Doorn et Pradeep Khosla. *SWATT : Software-based attestation for embedded devices*. Dans IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004, pages 272–282. IEEE, 2004. (Cité en page 46.)

[Smaha 1988] Stephen E Smaha*et al.* *Haystack : An intrusion detection system*. Dans Fourth Aerospace Computer Security Applications Conference, volume 44. Orlando, FL, USA, 1988. (Cité en pages 51 et 56.)

[Steiner 2016] Rodrigo Vieira Steiner et Emil Lupu. *Attestation in wireless sensor networks : A survey*. ACM Computing Surveys (CSUR), vol. 49, no. 3, pages 1–31, 2016. (Cité en pages v et 45.)

[Su 2006] Zhendong Su et Gary Wassermann. *The essence of command injection attacks in web applications*. Acm Sigplan Notices, vol. 41, no. 1, pages 372–382, 2006. (Cité en page 56.)

[Tang 2014] Adrian Tang, Simha Sethumadhavan et Salvatore J Stolfo. *Unsupervised anomaly-based malware detection using hardware features*. Dans International Workshop on Recent Advances in Intrusion Detection, pages 109–129. Springer, 2014. (Cité en page 66.)

[Wang 2015] Xueyang Wang, Charalambos Konstantinou, Michail Maniatakos et Ramesh Karri. *Confirm : Detecting firmware modifications in embedded systems using hardware performance counters*. Dans 2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pages 544–551. IEEE, 2015. (Cité en page 66.)

[Weaver 2013a] Vincent M Weaver. *Linux perf_event features and overhead.* Dans The 2nd International Workshop on Performance Analysis of Workload Optimized Systems, FastPath, volume 13, page 5, 2013. (Cité en page 64.)

[Weaver 2013b] Vincent M Weaver, Dan Terpstra et Shirley Moore. *Non-determinism and overcount on modern hardware performance counter implementations.* Dans 2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), pages 215–224. IEEE, 2013. (Cité en page 68.)

[Whitehouse 2007] Ollie Whitehouse. *An analysis of address space layout randomization on Windows Vista.* Symantec advanced threat research, pages 1–14, 2007. (Cité en page 48.)

[Zaidi 2015] Kamran Zaidi, Milos B Milojevic, Veselin Rakocevic, Arumugam Nallanathan et Muttukrishnan Rajarajan. *Host-based intrusion detection for vanets : a statistical approach to rogue node detection.* IEEE transactions on vehicular technology, vol. 65, no. 8, pages 6703–6714, 2015. (Cité en page 57.)

[Zarpelão 2017] Bruno Bogaz Zarpelão, Rodrigo Sanches Miani, Cláudio Toshio Kawakani et Sean Carlisto de Alvarenga. *A survey of intrusion detection in Internet of Things.* Journal of Network and Computer Applications, vol. 84, pages 25–37, 2017. (Cité en page 50.)

[Zhang 2006] Jiong Zhang et Mohammad Zulkernine. *Anomaly based network intrusion detection with unsupervised outlier detection.* Dans 2006 IEEE International Conference on Communications, volume 5, pages 2388–2393. IEEE, 2006. (Cité en page 60.)

[Zhou 2018] Boyou Zhou, Anmol Gupta, Rasoul Jahanshahi, Manuel Egele et Ajay Joshi. *Hardware performance counters can detect malware : Myth or fact ?* Dans Proceedings of the 2018 on Asia Conference on Computer and Communications Security, pages 457–468. ACM, 2018. (Cité en page 68.)

Abstract :

Today, the deployment of the Internet of Things (IoT) technologies is growing in all application areas. Unfortunately, the massive and fast deployment of IoT devices raises major security issues. For example, the Mirai attack took control of thousands of smart devices and used them in massive DDoS attacks. In this thesis, the environment that we envision is more controlled and restricted than usual IoT, we consider industrial deployment of devices. Even in these specific environments, some recent attacks caused serious damage, such as the Stuxnet attack targeting nuclear power plants.

This thesis studies the security of these devices, specifically for energy providers massively deploying devices that manage distributed resources or equipment. These devices may be deployed to efficiently manage the energy of smart factories or monitor the infrastructure of smart grids. By design, they generally exhibit a homogeneous behavior, with a similar software and hardware architecture, and are deployed in a large geographical area such as a region or a country. The approach described in this thesis allows us to detect compromised devices among a massive population of similar devices by analyzing the processor's Hardware Performance Counters. This analysis is based on existing outlier detection algorithms, with low computational and network traffic overhead, without modeling the behavior of the applications running on the devices. This approach is then easily adaptable to devices' updates. Moreover, as we only rely on the counters' analysis, we can detect anomalies of various origins, such as a compromised OS.

We performed thorough experiments based on two test platforms with various software profiles and attack payloads for the practical validation of our approach. The results obtained show that a high detection efficiency can be achieved, with low overhead and low execution time.

Résumé :

Les objets connectés, qui forment aujourd'hui l'Internet des objets ou IoT, sont de plus en plus nombreux et offrent une grande variété de fonctionnalités. Ils font cependant face à de nombreuses attaques, comme la célèbre attaque dénommée Mirai, qui a permis à des attaquants de prendre le contrôle de centaines de milliers d'objets connectés, notamment dans le domaine qui nous intéresse plus particulièrement dans cette thèse : le domaine industriel. Même dans ces environnements souvent restreints et très contrôlés, les objets connectés ont fait face à des attaques parfois dévastatrices, comme la fameuse attaque Stuxnet visant des centrales nucléaires.

Cette thèse se consacre à la protection de ces objets connectés, plus spécifiquement dans le domaine de l'énergie, pour la surveillance ou la maintenance du réseau électrique par exemple, ou encore pour l'optimisation des nouveaux moyens de consommation et de production locaux. Les objets envisagés sont identiques et déployés massivement à travers une région ou un pays. Plus spécifiquement, cette thèse porte sur la détection d'intrusions visant de tels objets connectés, en se basant sur l'analyse des déviations de comportement de ces objets. Nous proposons une solution basée sur l'observation et l'analyse de certains évènements internes au processeur via des compteurs de performances matérielles. Contrairement aux approches classiques se basant sur des détections faites au niveau des comportements logiciels, notre approche se base uniquement sur l'analyse de compteurs matériels et permet de se prémunir des malveillances qui pourraient cibler les systèmes d'exploitation par exemple. De plus, cette solution ne nécessite ni l'implication d'un expert ni de phase d'apprentissage automatique de modèle et s'adapte donc facilement aux potentielles mises à jour des objets. L'approche repose sur des algorithmes de détection de valeurs aberrantes sur des caractéristiques calculées à partir des valeurs de ces compteurs.

Ces travaux incluent également la description d'expérimentations permettant de valider notre approche. Ces expérimentations ont été effectuées sur deux plateformes composées d'objets connectés représentatifs d'un environnement industriel réel et elles montrent que notre approche est pertinente vis-a-vis d'un panel varié d'attaques.

Mots clés : IoT, Internet des Objets, Sécurité, Compteur matériel de performance, Détection d'anomalies

Annexe A

Analyse EBIOS

A.1 Métriques utilisées

Table A.1 – Métrique du critère de disponibilité

Disponibilité	
Besoin	Description
1	Le bien peut être indisponible plus de 48h
2	Le bien doit être disponible dans les 48h
3	Le bien doit être disponible dans les 12h
4	Le bien doit être disponible dans les 2h

Table A.2 – Métrique du critère d'intégrité

Intégrité	
Besoin	Description
1	Non-gênant, pas besoin d'intégrité
2	Détectable, l'altération doit être identifiée
3	Maîtrisé, l'altération doit être identifiée et corrigée
4	Intègre, les données doivent être intègres

Table A.3 – Métrique du critère de confidentialité

Confidentialité	
Besoin	Description
1	Public, pas besoin de confidentialité
2	Réservé, accessible en lecture par un groupe de personnes ou entités bien identifiées
3	Privé, accessible en lecture par une seul personne ou entité bien identifiée

Table A.4 – Métrique du critère de gravité

Gravité	
Niveau	Description
1	Négligeable, pas d'impact constaté
2	Limitée, impact minime
3	Importante, impact sérieux mais les dégâts restent réparables
4	Critique, impact grave, dégâts difficilement ou pas réparable

Table A.5 – Métrique du critère de vraisemblance

Vraisemblance	
Niveau	Description
1	Minime, ne devrait pas se produire
2	Significative, pourrait se produire
3	Forte, devrait se produire
4	Maximale, va se produire

A.2 Biens essentiels

Bien essentiel 01 : Consultation et paramétrage de l'objet depuis les serveurs

Critère de confidentialité

Cet évènement redouté correspond à la divulgation des ordres envoyés du serveur à l'objet.

- Besoin de sécurité : Privé, ces informations peuvent révélés un certain comportement de l'utilisateur et doivent être chiffrées (niveau 2).
- Sources de menace : FAI, personne extérieur ou intérieur malveillante ou objet connecté malveillant procédant à une attaque réseau.
- Impacts : Image de marque et possiblement révélation du comportement de l'utilisateur.
- Gravité : Importante.

Critère d'intégrité

Cet évènement redouté correspond au changement des ordres du serveur.

- Besoin de sécurité : Détectable, les ordres ne doivent pas pouvoir être changés par une tiers personnes mais n'ont pas de contraintes temporelles fortes.
- Sources de menace : FAI, personnes extérieur ou intérieur malveillante procédant à une attaque réseau, objet connecté malveillant.
- Impacts : Image de marque.
- Gravité : Limitée.

Critère de disponibilité

Cet évènement redouté correspond à l'impossibilité pour le serveur d'envoyer des données à l'objet.

- Besoin de sécurité : Doit être disponible dans les 48h, les ordres du serveur sont importants mais pas primordiaux au bon fonctionnement de l'objet.
- Sources de menace : FAI, personnes extérieur ou intérieur malveillante procédant à une attaque réseau, objet connecté malveillant.
- Impacts : Image de marque
- Gravité : Limitée

Bien essentiel 02 : Contrôle des modes par l'utilisateur

Critère de confidentialité

Cet évènement redouté correspond à la divulgation des mode utilisée (communications et automatismes)

- Besoin de sécurité : Privé, seul l'utilisateur doit savoir quand il veut activé ou désactivé les différents modes
- Sources de menace : Personne intérieur malveillant, objet connecté malveillant
- Impacts : Image de marque, révélation du comportement de l'utilisateur
- Gravité : Importante

Critère d'intégrité

Cet évènement redouté correspond au changement de mode par une personne autre que l'utilisateur

- Besoin de sécurité : Détectable, en cas de changement, l'utilisateur doit être prévenu.
- Sources de menace : Personne intérieur malveillante, objet connecté malveillant.
- Impacts : Image de marque et possible impact financier pour le client (en fonction de l'objet précisément défini, ici on considère qu'il peut y avoir un impact financier) .
- Gravité : Importante.

Critère de disponibilité

Cet évènement redouté correspond à l'impossibilité de changer de mode

- Besoin de sécurité : Disponible dans les 12h, on considère ici que le client accepte une telle indisponibilité de 12h.
- Sources de menace : Personne extérieur ou intérieur malveillante„ objet connecté malveillant
- Impacts : Image de marque
- Gravité : Limitée

Bien essentiel 03 : Données du client (liés à la configuration et aux données récoltés)

Critère de confidentialité

Cet évènement redouté correspond à la divulgation des données du client

- Besoin de sécurité : Réservé, seul le fournisseur de service et l'utilisateur doivent avoir accès à ces données.
- Sources de menace : Personne extérieur ou intérieur malveillante, objet connecté malveillant.
- Impacts : Image de marque, infraction au règlement de la CNIL sur les données à caractère personnel.
- Gravité : Critique.

Critère d'intégrité
Cet évènement redouté correspond au changement des données du client, que ce soit lorsque celles-ci remontent vers les serveurs distants ou lorsqu'elles sont saisies via l'interface de configuration.

- Besoin de sécurité : Détectable, pour éviter que des données modifiées par un tiers ne soient pris en compte.
- Sources de menace : Personne intérieur ou extérieur malveillante, objet connecté malveillant.
- Impacts : Image de marque.
- Gravité : Importante.

Critère de disponibilité
Cet évènement redouté correspond à la consultation ou au paramétrage des données du client

- Besoin de sécurité : Disponible dans les 48h, ces données ne devrait pas être changer régulièrement.
- Sources de menace : Personne extérieur ou intérieur malveillante, objet connecté malveillant
- Impacts : Image de marque
- Gravité : Limitée

Bien essentiel 04 : Données d'authentification auprès des serveurs
Critère de confidentialité
Cet évènement redouté correspond à la divulgation des données d'authentification auprès des serveurs distants

- Besoin de sécurité : Privé, seul le fournisseur de service devrait avoir accès à ces données.
- Sources de menace : Personne extérieure ou intérieure malveillante, objet connecté malveillant.
- Impacts : Image de marque, divulgation d'informations critique pouvant menées à d'autres attaques sur le fournisseur de service.
- Gravité : Importante

Critère d'intégrité
Cet évènement redouté correspond au changement des données d'authentification auprès des serveurs, que ce soit sur l'objet ou lors des échanges avec les serveurs.

- Besoin de sécurité : Intègre
- Sources de menace : Personne extérieure ou intérieure malveillante, objet connecté malveillant
- Impacts : Image de marque, impossibilité de s'authentifier auprès du serveur et donc de communiquer avec lui
- Gravité : Importante (occasionnera une gêne pour l'utilisateur mais pas un arrêt complet).

Critère de disponibilité
Cet évènement redouté correspond à l'inaccessibilité de ces données d'authentification (par exemple elles ont été effacées)

- Besoin de sécurité : Disponible dans les 48h, la connexion avec le serveur n'est pas primordiale au bon fonctionnement de l'objet
- Sources de menace : Personne intérieure ou extérieure malveillante, objet connecté malveillant.
- Impacts : Image de marque, impossibilité de s'authentifier auprès du serveur et donc de communiquer avec lui
- Gravité : Importante (occasionnera une gêne pour l'utilisateur mais pas un arrêt complet).

Bien essentiel 05 : Données d'authentification au réseau local
Critère de confidentialité
Cet évènement redouté correspond à la divulgation des données d'authentification au réseau local (Wi-Fi par exemple)

- Besoin de sécurité : Privé, seul l'utilisateur doit avoir accès à ces données.
- Sources de menace : Personne extérieure ou intérieure malveillante, objet connecté malveillant.
- Impacts : Image de marque, infraction au règlement de la CNIL sur les données à caractère personnel, intrusion de l'attaquant sur le réseau local.
- Gravité : Critique

Critère d'intégrité
Cet évènement redouté correspond au changement des données d'authentification, que ce soit sur l'objet ou lors de communications avec le routeur du client.

- Besoin de sécurité : Maîtrisé.
- Sources de menace : Personne extérieure ou intérieure malveillante, objet connecté malveillant.
- Impacts : image de marque

- Gravité : Limitée

Critère de disponibilité
Cet évènement redouté correspond à l'impossibilité de changer ces données d'authentification local.

- Besoin de sécurité : Disponible dans les 12h, a priori ces données ne devrait pas changer souvent après la première configuration.
- Sources de menace : Personne extérieure ou intérieur malveillante, objet connecté malveillant.
- Impacts : Image de marque
- Gravité : Importante, l'objet pourrait avoir besoin de s'authentifier sur le réseau pour fonctionner

Bien essentiel 06 : Logiciel
Critère de confidentialité
Cet évènement redouté correspond à la divulgation du code et des algorithmes utilisé par l'objet.

- Besoin de sécurité : Réservé, seul le constructeur et le fournisseur de service doivent avoir accès à ces algorithmes et au code.
- Sources de menace : Personnes extérieur ou intérieur malveillante, objet connecté malveillant
- Impacts : Vol des algorithmes utilisés, image de marque.
- Gravité : Importante

Critère d'intégrité
Cet évènement redouté correspond au changement du code ou des algorithmes utilisés.

- Besoin de sécurité : Intègre
- Sources de menace : Personne extérieure ou intérieure malveillante, objet connecté malveillant
- Impacts : Image de marque, détournement des fonctionnalités de l'objet.
- Gravité : Critique.

Critère de disponibilité
Cet évènement redouté correspond à l'indisponibilité du fonctionnement de l'objet.

- Besoin de sécurité : Disponible dans les 2h.
- Sources de menace : Personnes extérieur ou intérieur malveillante, objet connecté malveillant
- Impacts : Image de marque.
- Gravité : Importante.

Bien essentiel 07 : Mise à jour du logiciel

Critère de confidentialité

Cet évènement redouté correspond à la divulgation du processus de mise à jour.

- Besoin de sécurité : Privé, seul le fournisseur de service devrait avoir accès aux informations sur la mise à jour.
- Sources de menace : Personne extérieure ou intérieure malveillante, objet connecté malveillant
- Impacts : Image de marque
- Gravité : Importante

Critère d'intégrité

Cet évènement redouté correspond au changement de la mise à jour du logiciel et donc potentiellement du fonctionnement de l'objet.

- Besoin de sécurité : Maîtrisé.
- Sources de menace : Personne extérieure ou intérieure malveillante, objet connecté malveillant.
- Impacts : Image de marque.
- Gravité : Critique, il peut y avoir des mise à jour corrigeant certaines vulnérabilités par exemple.

Critère de disponibilité

Cet évènement redouté correspond à l'impossibilité de mettre à jour l'objet.

- Besoin de sécurité : Disponible dans les 48h.
- Sources de menace : Personne extérieure ou intérieure malveillante, objet connecté malveillant.
- Impacts : Image de marque.
- Gravité : Critique, il peut y avoir des mise à jour corrigeant certaines vulnérabilités par exemple.