Titre: Title:	SERENIoT : politiques de sécurité collaboratives pour maisons connectées
Auteur: Author:	Corentin Thomasset
Date:	2020
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Thomasset, C. (2020). SERENIoT : politiques de sécurité collaboratives pour maisons connectées [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/5322/

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

SERENIoT : Politiques de sécurité collaboratives pour maisons connectées

CORENTIN THOMASSET
Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie informatique

Juillet 2020

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

SERENIoT : Politiques de sécurité collaboratives pour maisons connectées

présenté par **Corentin THOMASSET**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Alejandro QUINTERO, président
José FERNANDEZ, membre et directeur de recherche
David BARRERA, membre et codirecteur de recherche
Jeremy CLARK, membre et codirecteur de recherche
Nora BOULAHIA CUPPENS, membre

DÉDICACE

À ma famille et à tous ceux qui me sont chers, merci pour votre soutien.

REMERCIEMENTS

Je souhaite remercier toutes les personnes m'ayant permis de mener à bien mon projet de recherche. Tout d'abord les Pr. David Barrera et Jeremy Clark pour l'expertise qu'ils m'ont apportée dans leurs domaines respectifs et pour le temps qu'ils m'ont accordé pour faire avancer mon projet. Je tiens à remercier particulièrement Pr. David Barrera pour la confiance et l'autonomie qu'il m'a accordée au cours de ces deux années. Je le remercie aussi pour m'avoir fait découvrir le monde des conférences académiques et de la recherche.

Je tiens aussi à remercier toutes les personnes impliquées dans la rédaction de mon mémoire et dans les tâches administratives tout au long de ma maîtrise à Polytechnique Montréal. Le Pr. José M. Fernandez et Militza Jean pour leur assistance pour les démarches administratives. Je remercie aussi Marielba Urdaneta Velasquez pour le temps passé à m'aider à relire mon mémoire.

Je souhaite remercier mes collègues du laboratoire pour les bons moments passés ensemble, notamment François Labrèche, Jean-Yves Ouattara, Antonio Domingez et Mikaela Ngamboe.

Je remercie aussi Rose Hirigoyen et Adem Aber Aouni pour avoir rejoint mon groupe de projet en INF8808 et pour m'avoir aider à développer les graphiques D3.js utilisés dans l'interface web de SERENIoT.

Enfin je souhaite remercier mes parents pour le temps passé à relire mon mémoire.

RÉSUMÉ

Le domaine de la maison connectée est en plein essor. L'internet des objets (IoT) apporte de nouveaux services dans l'habitat pour répondre aux évolutions énergétiques, numériques et de bien-être. Cependant ces objets connectés peuvent créer des failles de sécurité dans les environnements domestiques mettant en péril la sécurité des utilisateurs et des réseaux.

Le contexte de l'environnement domestique impose un prix des équipements faible, une diversité de dispositifs très importante et des solutions de sécurité ne nécessitant pas d'action de l'utilisateur. L'ensemble de ces facteurs complique la mise en oeuvre de solutions de sécurité performantes et adaptées aux réseaux IoT domestiques.

La recherche menée porte sur le développement d'un système permettant une gestion autonome des politiques de sécurité pour sécuriser les installations IoT domestiques. Elle montre que les politiques de sécurité à appliquer pour sécuriser les dispositifs IoT domestiques peuvent être générées en observant leurs comportements au sein de différentes installations. L'objectif s'articule autour de la conception d'un système permettant la création et la mise en application de politiques de sécurité grâce à un système de filtrage réseau au niveau IP.

Aussi, après avoir validé la typologie générale des empreintes réseau des dispositifs IoT domestiques, nous avons développé SERENIoT (prononcé Serenity), un système autonome de génération et de mise en vigueur de politiques de sécurité pour réseaux IoT. SERENIoT surveille et filtre le trafic réseau des dispositifs IoT grâce à un dispositif appelé sentinelle qui s'intègre au sein des réseaux domestiques. Les sentinelles profilent le comportement des dispositifs IoT localement et se basent sur une blockchain pour déterminer si les comportements observés localement correspondent aux comportements observés globalement, par les autres sentinelles de SERENIoT. Les spécifications desquelles dérivent les politiques de sécurité sont le résultat d'un algorithme de consensus permettant d'identifier les comportements réseau observés par la majorité des participants du réseau. Les connexions réseau observées uniquement pour un dispositif sont ainsi bloquées tant qu'elles ne sont pas observées globalement, permettant de stopper les attaques ciblées et la propagation de botnets tels que Mirai.

Notre prototype et son concept ont été validés d'une part d'un point de vue fonctionnel par des tests sur un réseau IoT réel et d'autre part par simulation pour vérifier la compatibilité du système avec un panel d'IoT. Ainsi, nous avons validé lors d'une expérience avec des dispositifs réels que SERENIoT était capable de générer des politiques de sécurité de manière autonome et de les mettre en vigueur pour bloquer les connexions anormales. De plus, nous avons analysé le comportement de 53 dispositifs IoT issus d'une base de données publique

afin de caractériser la typologie des comportements de ces dispositifs. Nous avons alors pu vérifier que la majorité de ces dispositifs ont une empreinte réseau simple, constituée d'une dizaine de signatures de paquets, qui évolue peu au cours du temps. Enfin, nous avons validé le concept et son fonctionnement par différentes simulations à plus grande échelle pour vérifier la compatibilité et l'extensibilité du système. Celui-ci à mis correctement en place les politiques de sécurité en gérant correctement l'augmentation du nombre de sentinelles et de dispositifs IoT différents.

SERENIoT ouvre ainsi la voie à un nouveau type de systèmes de détections d'intrusions collaboratives ne nécessitant pas d'intervention humaine et adaptées à un usage domestique tel que les maisons connectées.

ABSTRACT

The recent development of the Internet of Things (IoT) brings new services to homes to improve energy efficiency and users' well-being. However, these new smart devices can create security holes in home environments, placing the security of users and networks at risk.

The design and deployment of effective security solutions tailored to home IoT networks is made difficult by the wide diversity of devices along with the constraints of equipment costs and impact on usability.

This thesis focuses on the development of a system allowing autonomous management of security policies to secure home IoT installations. The system demonstrates that the security policies for consumer IoT devices can be generated by observing their behavior within different installations. The goal was to design an autonomous system capable of generating and enforcing security policies by monitoring the network at IP level.

Thus, after validating the general typology of consumer IoT devices' network footprints, we have developed SERENIoT (pronounced Serenity), an autonomous system for generating and enforcing security policies for IoT networks. SERENIoT monitors and filters network traffic of IoT devices through a device called Sentinel which is integrated into home networks. Sentinels profile the behavior of IoT devices locally and use a blockchain to determine whether the behaviors observed locally match the behaviors observed globally, by the other Sentinels in the distributed SERENIoT network. Security policies are the result of a consensus algorithm identifying the network behaviors observed by the majority of network participants. Network connections that are unique to a device are blocked until they are observed by most nodes, preventing the spread of Mirai-style botnets.

We evaluate a prototype of the SERENIoT network through experiments on a real IoT network and on a simulated environment to ensure compatibility of the system with existing devices. We initially carried out a functional experiment with real devices, during which we validated that SERENIoT is capable of generating and enforcing security policies autonomously to block abnormal connections. We then analyzed the behavior of 53 IoT devices from a public dataset to categorize the different types of devices based. We observed that the majority of these devices have a network footprint composed of a dozen packet signatures and these signatures are mostly constant over time. Finally, we carried out various larger-scale simulations with these devices to test the compatibility and extensibility of the system. We observed that the system worked correctly with IoT devices with a simple and constant network footprint and that it scaled correctly to the increase in the number of Sentinels and devices.

SERENIoT is a first step toward a new type of collaborative intrusion detection systems that do not require human intervention and are therefore more likely to be deployed in smart homes.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

IoT	Internet des Objets (De l'anglais *Internet of Things*)
API	Interface de programmation (De l'anglais *Application programming interface*)
REST	De l'anglais *Representational State Transfer*
IDS	Systèmes de Détection d'Intrusion (De l'anglais *Intrusion Detection Systems*)
NIDS	Systèmes de Détection d'Intrusion Réseau (De l'anglais *Network Intrusion Detection Systems*)
IETF	*Internet Engineering Task Force*
IEC	Commission électrotechnique internationale (De l'anglais *International Electrotechnical Commission*)
MUD	De l'anglais *Manufacturer Usage Description*
LAN	Réseau local (De l'anglais *Local Area Network*)
WAN	Réseau étendu (De l'anglais *Wan Area Network*)
NAT	De l'anglais *Network Address Translation*
FAI	Fournisseur d'Accès Internet
OUI	De l'anglais *Organizationally Unique Identifier*
IP	De l'anglais *Internet Protocol*
TCP	De l'anglais *Transmission Control Protocol*
UDP	De l'anglais *User Datagram Protocol*
NTP	De l'anglais *Network Time Protocol Protocol*
DNS	De l'anglais *Domain Name System*
TLS	De l'anglais *Transport Layer Security*
FTP	De l'anglais *File Transfer Protocol*
PoW	De l'anglais *Proof of Work*
PoS	De l'anglais *Proof of Stake*
PoET	De l'anglais *Proof of Elapsed Time*
RRR	De l'anglais *Robust Round Robin*
MITM	Attaque de l'homme du milieu (De l'anglais *Main In The Middle*)
XSS	De l'anglais *Cross Site Scripting*
RAM	Mémoire vive (De l'anglais *Random-Access Memory*)
CPU	Processeur (De l'anglais *Central Processing Unit*)
JSON	De l'anglais *JavaScript Object Notation*

CHAPITRE 1 INTRODUCTION

L'Internet des Objets (IoT) est en plein essor. Ces nouveaux dispositifs intelligents facilitent la collecte d'informations et le développement de systèmes automatisés contrôlables à distance. Ils connectent le monde physique au monde virtuel et sont en passe de révolutionner nos villes et nos foyers en dotant les objets de notre quotidien de fonctionnalités intelligentes. Cette révolution soulève de nombreuses questions en termes de sécurité et de vie privée. Les dispositifs destinés au grand public sont en effet conçus pour maximiser les marges des constructeurs. Leurs interfaces, souvent rudimentaires compliquent les interactions avec l'utilisateur et le déploiement de mises à jour et de patchs de sécurité. Sans mises à jour, les logiciels qu'ils embarquent sont rapidement dépassés et peinent à se conformer aux dernières recommandations en termes de sécurité. De par leur nombre et leur diversité, ils mettent à l'épreuve les modèles de sécurité classiques de sécurisation des réseaux, souvent trop permissifs à leur égard [1]. L'absence d'expert pour maintenir les infrastructures de sécurité des réseaux domestiques rend en effet impossible la mise en place de politiques de sécurité spécifiques aux dispositifs et complique l'utilisation de pare-feu, de *Intrusion Detection Systems* (IDS) et de *Intrusion Prevention Systems* (IPS). De plus, aucune des différentes solutions proposées dans la littérature ne semble vraiment adaptée à l'usage domestique ou compatible avec un déploiement à grande échelle.

La recherche menée et présentée dans ce mémoire vise à développer et prototyper un nouveau concept permettant d'apporter des réponses à la problématique de sécurisation des dispositifs IoT en milieu domestique. Il n'existe en effet à ce jour aucune solution de sécurité adaptée aux dispositifs IoT grand publique et aux utilisateurs novices. L'objectif est de développer un système collaboratif de génération et de mise en vigueur de politiques de sécurité compatible avec la majorité des dispositifs IoT domestiques actuels et futurs permettant la sécurisation des réseaux IoT domestiques.

1. Une majorité des dispositifs connectés ont en effet un jeu de fonctionnalités très limité et les politiques de sécurité par défaut des réseaux domestiques autorisent souvent plus de connexions que nécessaire.

1.1 Définitions et concepts de base

1.1.1 Internet des objets

L'internet des objets est un terme générique de plus en plus utilisé pouvant désigner tout objet physique mis en réseau. L'internet des objets a de ce fait de nombreuses applications dans différents domaines et ne désigne pas forcément le même type de dispositifs en fonction du contexte. Ce mémoire traite de l'internet des objets domestique. Nous désignons donc par «internet des objets» les objets électroménagers et autres objets du quotidien dotés de fonctionnalités «intelligentes» tels que les «smart bulbs», «smart plug», «smart fridge», «smart cam», etc. Ces objets sont généralement contrôlables via une application mobile et disposent de fonctionnalités étendues par rapport aux objets similaires non connectés (par exemple une ampoule vs une ampoule connectée). Ces nouvelles fonctionnalités donnent plus de contrôle à l'utilisateur et lui permettent généralement d'améliorer son confort de vie.

1.1.2 Intrusions et politiques de sécurité

Une intrusion désigne un accès à une ressource non autorisée. Dans le contexte informatique, une intrusion désigne généralement un accès non autorisé à des données ou à un système entraînant sa prise de contrôle temporaire par un acteur malveillant. Les politiques de sécurité visent à protéger les systèmes informatiques contre notamment les intrusions et définissent un ensemble de «consignes de sécurité» à appliquer pour sécuriser le système. Concrètement une politique de sécurité peut définir une liste d'utilisateurs autorisés à utiliser un système ou une liste des connexions réseau autorisées pour un dispositif. Une fois la politique de sécurité définie, elle doit être mise en vigueur à l'aide de moyens technologiques tels que des pare-feu ou des systèmes de contrôle d'accès. Dans ce mémoire le terme «politique de sécurité» désigne une politique de sécurité réseau spécifiant les connexions réseau autorisées pour les dispositifs. Nous mettons en vigueur ces politiques de sécurité à l'aide d'un pare-feu.

1.1.3 Blockchain

Une blockchain est un registre répliqué et synchronisé entre différents participants d'un réseau pair à pair. Ce registre peut être utilisé pour stocker des transactions monétaires comme dans le cas du Bitcoin ou toutes autres données critiques nécessitant une structure de donnée décentralisée fiable. Contrairement aux bases de données classiques, une blockchain ne permet de stocker que quelques méga-octets de données qui ne peuvent pas être modifiées.

1.2 Éléments de la problématique

Les appareils IoT sont aujourd'hui omniprésents dans nos maisons [1]. Des ampoules aux appareils électroménagers, de plus en plus d'appareils embarquent des fonctionnalités intelligentes. Pour fonctionner, ces dispositifs nécessitent une connexion réseau permanente et sont souvent connectés à internet. Leur installation et configuration repose sur l'utilisateur final qui n'a pas forcément les connaissances nécessaires pour les configurer correctement. De plus, les dispositifs IoT grand public sont conçus pour être peu coûteux et ne sont majoritairement pas conformes aux dernières recommandations de sécurité. Mal sécurisés, ils constituent une menace à leurs utilisateurs et à leur environnement [2–5] et peuvent être exploités par des acteurs malintentionnés pour mener à bien des attaques de grande ampleur contre les infrastructures réseau d'internet [6].

La diversité logicielle et matérielle des dispositifs IoT ainsi que leur volume de déploiement complique la mise en place de systèmes de sécurité efficaces. La majorité des solutions de sécurité grand public utilisées dans les foyers sont en effet trop permissives et ne sont pas adaptées pour protéger efficacement les dispositifs IoT [7]. Contrairement aux ordinateurs personnels, tablettes et téléphones intelligents, les dispositifs IoT ont un jeu de fonctionnalités limité, qui ne varie pas ou peu à l'utilisation. Typiquement, les dispositifs IoT effectuent des relevés sur leur environnement et les envoient dans le cloud (par exemple un capteur d'humidité) et/ou attendent des commandes pour effectuer une tâche (par exemple un interrupteur WiFi). Les dispositifs IoT ne requièrent donc pas les mêmes privilèges réseau que les systèmes complexes tels que les ordinateurs [2] et nécessitent un filtrage spécifique à leurs fonctionnalités. Cependant, les dispositifs IoT sont généralement traités indifféremment des ordinateurs personnels et sont autorisés à se connecter à tous les hôtes sur internet. De ce fait, les dispositifs IoT compromis sont couramment utilisés par des *botnets* lors d'attaques de grande envergure pour attaquer des hôtes sur internet [6].

Afin de contrer ces attaques, l'utilisation de systèmes de sécurité classiques tels que les IDS par signatures ou par anomalies est possible. Cependant, ces systèmes de sécurité nécessitent une supervision par l'utilisateur pour monitorer les alertes et ajuster les paramètres et la logique de détection afin de minimiser le taux de faux positifs. De plus les IDS par signature ne sont efficace que pour détecter les attaques dont la signature est connue. Les IDS par anomalies nécessitent quant à eux une période d'apprentissage afin de profiler le comportement normal des périphériques. Cette période d'apprentissage doit être supervisée par des experts afin d'être sûr de ne pas profiler de comportements malveillants et doit être répétée à chaque

2. Un comportement normal pour un ordinateur personnel peut s'avérer malveillant dans le cas d'un appareil IoT (par exemple accéder à www.google.ca dans le cas d'un four connecté)

mise à jour des dispositifs.

Une approche alternative est d'utiliser un IDS par spécification afin de n'autoriser que le trafic réseau nécessaire au fonctionnement des dispositifs IoT. Cette méthode permet d'obtenir un taux de faux positifs (c.-à-d. bloquer du trafic légitime) très bas si les spécifications décrivent les comportements des dispositifs avec précision. Pour ce faire, la norme *Manufacturer Usage Description* (MUD) [8] standardise le langage permettant de décrire le comportement des dispositifs IoT afin de mettre en place des politiques de sécurité. Les fabricants peuvent ainsi rendre accessibles les spécifications comportementales de leurs dispositifs dans un format standardisé et interprétable directement par les systèmes de sécurité. Cependant cette solution dépend directement des fabricants et se heurte au problème du nombre et de la diversité des dispositifs IoT. Il existe un trop grand nombre de versions de dispositifs IoT uniques dont certains sont conçus et vendus sous différentes marques. De plus, la durée du support fournie par les fabricants est peu clair et il est de ce fait difficile de leur faire confiance pour fournir des spécifications fiables et à jour durant tout le cycle de vie de leurs produits. Alternativement, les spécifications comportementales des dispositifs IoT pourraient être fournies par un tiers de confiance analysant les différents dispositifs pour produire leurs spécifications. Cette solution semble aussi difficilement envisageable au vu du nombre de dispositifs IoT différents. Il n'est par ailleurs pas clair que les utilisateurs soient prêts à payer pour sécuriser leurs dispositifs IoT et les retours sur investissement d'un tel service seraient donc à évaluer.

Pour résoudre le problème de la création des spécifications comportementales des dispositifs IoT, nous proposons un système collaboratif de création et de mise en vigueur de politiques de sécurité pour dispositifs IoT. Notre système profile automatiquement le comportement des dispositifs IoT localement et se base sur un registre décentralisé pour déterminer si les comportements observés localement correspondent aux comportements observés globalement, par les autres participants du réseau. Les spécifications desquelles dérivent les politiques de sécurité sont le résultat d'un algorithme de consensus permettant d'identifier les comportements réseau observés par la majorité des participants du réseau. Les connexions réseau observées uniquement pour un dispositif sont ainsi bloquées tant qu'elles ne sont pas observées globalement, permettant de stopper la propagation des botnets tels que Mirai [6].

1.3 Objectifs de recherche

1.3.1 Objectif général

L'objectif de cette recherche est de répondre à la question suivante :

Est-il possible de déployer une solution autonome permettant la génération et la mise en vigueur de politiques de sécurité spécifiques pour réseaux IoT domestiques ?

1.3.2 Objectifs spécifiques

1. Identifier les enjeux de la sécurisation des réseaux IoT domestiques.
2. Valider qu'une majorité de dispositifs IoT domestiques ont une empreinte réseau simple et constante.
3. Concevoir un système permettant de générer et de mettre en vigueur des politiques de sécurité réseau pour dispositifs IoT domestiques sans intervention humaine.
4. Valider le fonctionnement du système par test sur un réseau IoT réel.
5. Valider la compatibilité du système avec différents dispositifs IoT par simulation.
6. Valider l'extensibilité du système par simulation.
7. Améliorer le système à l'aide des résultats expérimentaux.

1.3.3 Hypothèse

Les dispositifs IoT grand public ont une empreinte réseau simple et constante. Les politiques de sécurité à appliquer pour les sécuriser peuvent donc être apprises en observant les comportements des dispositifs sur différentes installations domestiques. Les comportements majoritaires sont considérés «normaux» et les comportements observés chez une minorité sont bloqués. Ainsi, la configuration des politiques de sécurité et la construction des spécifications fonctionnelles des dispositifs ne reposent plus sur l'utilisateur final ni sur les fabricants et le système apprend dynamiquement les politiques de sécurité à appliquer.

1.4 Portée de la recherche

La recherche présentée dans ce mémoire traite la problématique de la sécurisation des réseaux IoT domestiques. Nous étudions la création et la mise en application de politiques de sécurité grâce à un système de filtrage réseau des communications IP. Le spectre des dispositifs étudiés est ainsi centré les dispositifs IoT WiFi. Bien que les communications des dispositifs utilisant

des protocoles réseaux à courte portée tels que Bluetooth, Zigbee, Z-Wave, etc ne puissent pas être directement analysées, les communications issues des hubs domotiques auxquels ils sont connectés le sont (les hubs domotiques servent de pont entre les réseaux à courte porté et le réseau IP et permettent d'agréger plusieurs dispositifs Zigbee/Z-wave). Notre recherche se focalise aussi sur les dispositifs grand public que l'on retrouve au sein des maisons connectées ayant une empreinte réseau «simple» (composée d'une dizaine de signatures de paquets). De plus, nous nous intéressons uniquement aux connexions *Local Area Network (LAN) to Wide Area Network (WAN)* (entre le réseau local et internet).

1.5 Plan du mémoire

Le chapitre 2 présente le contexte de notre étude ainsi qu'une revue des solutions académiques et commerciales proposées pour sécuriser les réseaux IoT domestiques. Le chapitre 3 présente le système SERENIoT et la méthodologie mise en place afin de générer et de mettre en vigueur des politiques de sécurités spécifiques aux réseaux IoT domestiques. Ensuite, le chapitre 4 présente le prototype de SERENIoT développé ainsi que les expériences menées pour évaluer le système. Le chapitre 5 discute des résultats obtenus lors de l'évaluation du système et de l'approche utilisée par SERENIoT pour répondre à la problématique. Enfin, le chapitre 6 conclue sur les travaux effectués et donne des pistes d'améliorations pour de futures versions de SERENIoT.

CHAPITRE 2 CONTEXTE

Ce chapitre définit le contexte de notre étude et présente les solutions commerciales et académiques visant à sécuriser les réseaux IoT domestiques.

2.1 IoT domestique & domotique

L'internet des objets (IoT) désigne les objets physiques connectés à un réseau informatique. Ces objets établissent un lien entre le monde physique et les réseaux informatiques afin de faciliter l'automatisation, le contrôle à distance et la collecte de données. L'internet des objets a de ce fait de nombreuses applications dans différents domaines et ne désigne pas forcément le même type de dispositifs en fonction du contexte.

L'internet des objets domestique s'apparente à la domotique et désigne généralement l'ensemble de solutions technologiques visant à améliorer le confort de vie de l'utilisateur. Il caractérise les dispositifs tels que les ampoules connectées dont la fonctionnalité de base a été étendue avec des fonctionnalités d'automatisation et de contrôle à distance [1]. Ainsi, de plus en plus d'objets du quotidien et d'appareils électroménagers sont dotés de fonctionnalités intelligentes [9]. Ils nécessitent une connexion au réseau local permanente et sont généralement connectés à internet. Contrairement aux dispositifs IoT industriels, ces dispositifs ciblent le grand public et l'adoption de masse et tirent généralement les coûts de production vers le bas afin de maximiser les marges constructeurs tout en restant dans une gamme de prix accessibles par les particuliers.

2.1.1 Enjeux de l'internet des objets

Plusieurs rapports [9,10] montrent que l'internet des objets est un marché en pleine explosion. Au Canada, Statista [11] estime que le marché de la maison connecté atteindra 3449 millions de dollars canadiens d'ici 2023, soit un taux de croissance annuel entre 2020 et 2023 de 18,2%. Mondialement, le nombre de dispositifs IoT en service en 2025 devrait atteindre 21,5 milliards contre 7 milliards en 2018 selon une étude du cabinet IoT Analytics [12].

Dans cet environnement de la maison connectée qui offre de nouveaux services liés au confort, au bien être, mais aussi à l'efficacité énergétique des bâtiments, à la santé et la sécurité,

1. Cela ne désigne donc pas les ordinateurs personnels. Cependant, les consoles de jeux et téléphones intelligents sont régulièrement associés à l'internet des objets bien que ces dispositifs s'apparentent plus à un ordinateur personnel qu'à une ampoule connectée en termes de fonctionnalités.

les IoT apportent une réelle plus-value. Cependant, ceux-ci peuvent exposer les utilisateurs à différents types de risques impactant leur vie privée, leur sécurité physique ou celle des infrastructures. Par exemple, ceci peut se concrétiser par la prise de contrôle d'une caméra par un acteur malveillant, mais aussi par une perturbation du réseau électrique suite au détournement de climatiseurs connectés [13].

La fiabilité et la protection des données des objets connectés sont ainsi des enjeux importants, car ceux-ci sont affectés à des usages impactant directement l'humain comme par exemple les équipements dans le domaine de la santé pour l'aide à la personne à domicile.

La démultiplication des dispositifs IoT et leur utilisation accrue dans les années à venir constituent un défi important au niveau de la sécurité. Aussi il primordial que des systèmes de protection dédiés aux IoT et adaptés au grand public soient mis en place.

2.1.2 Typologie des dispositifs

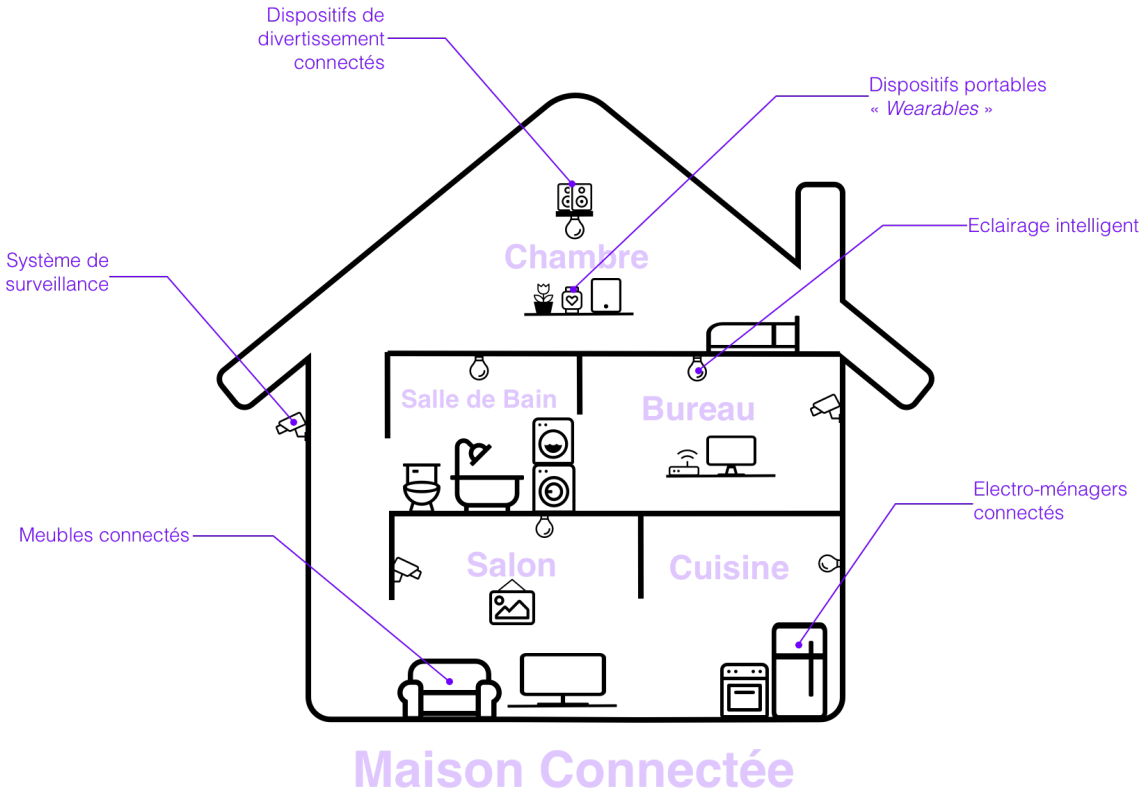


Figure 2.1 Exemple de maison connectée

L'internet des objets permet aux utilisateurs de pouvoir contrôler la teinte de l'éclairage dans sa maison pièce par pièce, de monitorer la température et la qualité de l'air, de consulter

à distance les stocks de nourriture dans le réfrigérateur, de recevoir des alertes lorsque la cuisson d'un plat est terminée dans le four, etc. Une des applications les plus communes est la caméra connectée, souvent utilisée au chevet des jeunes enfants pour pouvoir les surveiller à distance. La figure 2.1 montre un exemple de maison connectée avec différents dispositifs IoT. Dans un rapport de 2018 [14], Underwriters Laboratories classent les différents dispositifs IoT domestiques en 6 catégories :

- **Les dispositifs pour le divertissement** sont les dispositifs les plus présents au sein des maisons connectées en Amérique du Nord et en Europe de l'Ouest [1]. Parmi ces dispositifs on retrouve les dispositifs de streaming audio et vidéo et les *smart tv*. Ces appareils multimédias connectés permettent l'accès sans fil au contenu multimédia disponible sur internet. Cette catégorie comprend aussi les consoles de jeux qui se rapprochent de plus en plus des ordinateurs personnels en termes de puissance de calcul et de fonctionnalités (par exemple le *Xbox One system software* est basé sur Windows 10).
- **Les dispositifs pour la surveillance et la sécurité** sont populaires en Asie du Sud et en Asie du Sud-est [1]. Ces dispositifs visent à renforcer la sécurité de la maison connectée. Cette catégorie comprend les caméras connectées, mais aussi les serrures connectées et les détecteurs de monoxyde de carbone connectés.
- **Les dispositifs de gestion d'énergie et des ressources** visent à améliorer l'efficacité énergétique des maisons connectées en ajustant les consommations d'électricité, d'eau et de gaz. Cette catégorie comprend les compteurs connectés, les thermostats intelligents ainsi que les appareils ménagers connectés.
- **Dispositifs d'aide à la gestion des ménages et d'automatisation de la maison.** Par exemple les assistants vocaux, les robots cuisine, les stores et volets connectés, etc. Ces dispositifs permettent de faciliter les tâches ménagères.
- **Les dispositifs d'éclairage intelligents** contribuent à l'amélioration du confort de l'utilisateur et de l'efficacité énergétique de la maison. Cette catégorie comprend les ampoules connectées et les systèmes d'éclairage.
- **Les dispositifs pour la santé et le fitness** sont de plus en plus présent au sein des ménages et contribuent au bien être des utilisateurs en monitorant leurs indicateurs de santé vitaux. Ils permettent par exemple de détecter les chutes et les mouvements brusques. Ces dispositifs portatifs s'apparentent par exemple aux montres connectées qui embarquent de plus en plus de fonctionnalités orientées sur la santé de l'utilisateur (par exemple l'Apple Watch embarque un détecteur de chutes permettant de mettre l'utilisateur en contact avec les services d'urgence [15]).

Cette grande variété de dispositifs IoT se retrouve aussi au niveau matériel et logiciel. Ainsi,

les technologies des dispositifs IoT sont encore peu standardisées et les fabricants se tournent la majorité du temps vers des solutions propriétaires. Bien que de nombreux dispositifs IoT utilisent des versions modifiées de distributions Linux *open source* (à l'image des contrôleurs Vera [16] basés sur OpenWRT [2] avec une surcouche constructeur), peu de fabricants rendent disponible leur code ce qui complique l'analyse et la compréhension du fonctionnement de ces dispositifs. De plus tous les dispositifs IoT ne sont pas égaux en termes de puissance de calcul, d'alimentation électrique et de stockage ce qui complique une uniformisation matérielle et logicielle et pousse souvent les constructeurs à se tourner vers des solutions propriétaires optimisées spécialement pour leurs dispositifs.

2.1.3 Topologie des maisons connectées

La figure 2.2 montre une topologie classique de réseau de maison connectée.

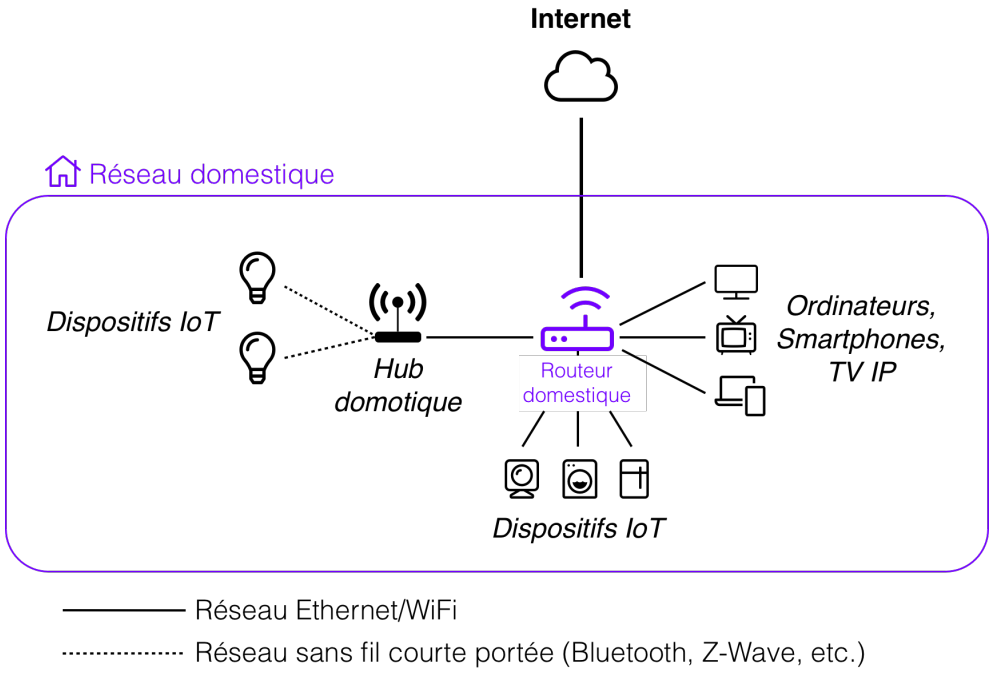


Figure 2.2 Topologie des maisons connectées

Le routeur domestique mis à disposition par le Fournisseur d'Accès Internet (FAI) est le point central du réseau. La majorité des dispositifs du réseau local sont connectés directement au routeur soit par le biais d'un câble Ethernet soit par WiFi. Parmi ces dispositifs, on retrouve classiquement les ordinateurs, les smartphones et les tablettes ainsi que les dispositifs

2. OpenWRT est une distribution Linux pour matériel embarqué, typiquement utilisée sur les routeurs. [17]

IoT Ethernet/WiFi. Les dispositifs IoT utilisant des protocoles à courte distance tels que Bluetooth ou Zigbee peuvent être connectés à un hub domotique. Le hub domotique est connecté au réseau Ethernet domestique et sert d'intermédiaire entre les dispositifs qui lui sont connectés et le reste du réseau. Dans la majorité des cas le réseau possède un unique point d'accès internet via le routeur domestique qui met en place un *Network Address Translation* (NAT). Les connexions initiées par les dispositifs à l'extérieur du réseau sont alors bloquées par le NAT [3].

2.1.4 Risques de sécurité liés à la maison connectée

Un rapport de 2019 par Avast sur la sécurité des maisons connectées [18] estime que 40.8% des maisons connectées dans le monde possèdent au moins un dispositif vulnérable. Ce rapport estime aussi que parmi les dispositifs vulnérables, 31.8% le sont à cause d'une vulnérabilité non patchées dans leur logiciel et 69.2% le sont à cause d'identifiants trop faibles.

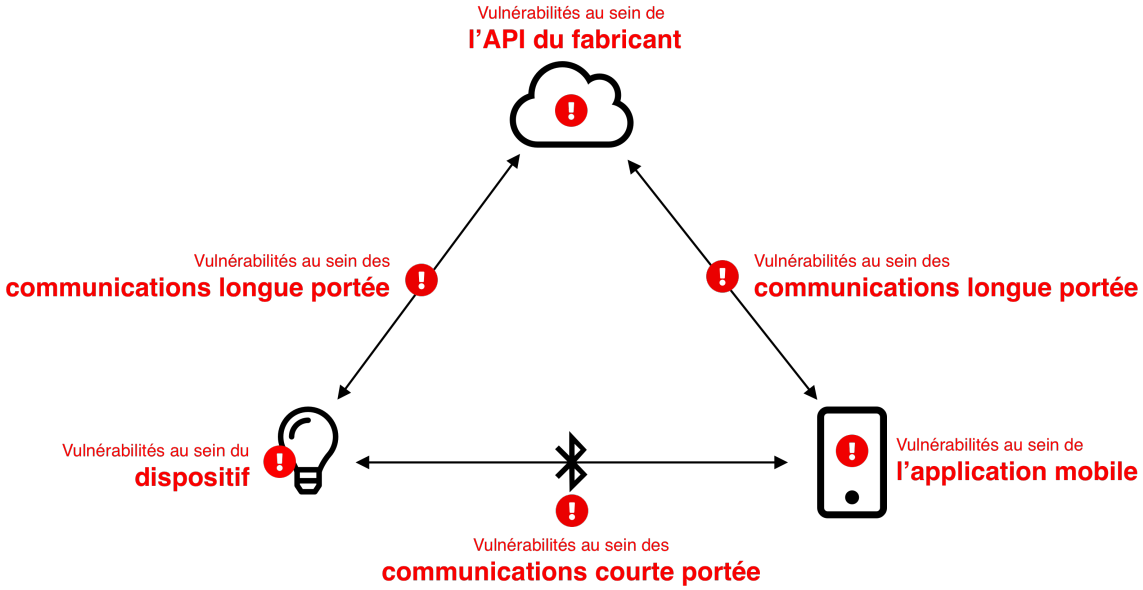


Figure 2.3 Points vulnérables de l'écosystème IoT

Ainsi, comme le soulignent Alrawi et al. dans leur étude sur la sécurité des maisons connectées de 2019 [2], l'écosystème de l'internet des objets est vulnérable à différents niveaux. La figure 2.3 montre les différents points vulnérables de l'écosystème IoT identifiés dans leur étude.

3. Cependant, des redirections de ports peuvent être mises en place pour autoriser certaines connexions entrantes

- **Au niveau des dispositifs.** Différents vecteurs d'attaques sont possibles sur les dispositifs IoT : L'exploitation de vulnérabilités du logiciel, l'exploitation d'identifiants faibles et l'exploitation de la configuration par défaut. Ces vecteurs d'attaques sont régulièrement utilisés par des acteurs malveillants pour compromettre des dispositifs IoT, à l'image du botnet Mirai [6] qui exploitait l'utilisation des identifiants par défaut pour prendre le contrôle de caméras connectées.
- **Au niveau du service *cloud*.** Comme le montre l'analyse de sécurité de la poupée Hello Barbie [19], les *Application programming interface* (API) des fabricants des dispositifs IoT souffrent des mêmes vulnérabilités que les applications web classiques (Vulnérabilités *Cross Site Scripting* (XSS), mauvaise gestion de l'authentification, aucune protection contre le *brute force* des identifiants, etc).
- **Au niveau de l'application mobile.** L'analyse de sécurité du dispositif *August Smart Lock* par Max [20] montre que certaines applications mobiles utilisées pour contrôler les dispositifs IoT contiennent par exemple des informations sensibles *hard codées* qu'il est possible d'extraire et qui mettent en danger la sécurité des dispositifs.
- **Au niveau des communications.** D'après Alrawi et al. [2], les communications de nombreux dispositifs IoT ne sont pas suffisamment sécurisées et sont vulnérables à des attaques de type *Man In The Middle* (MITM). Certaines implémentations des librairies des chiffrements SSL/TLS sont aussi vulnérables.

Les maisons connectées sont de ce fait exposées à trois types de risques comme le souligne la figure 2.4 :

- **Les risques impactant la vie privée des utilisateurs.** Un attaquant peut être en mesure d'intercepter ou d'accéder à des données mettant en danger la vie privée des utilisateurs. Il peut par exemple avoir accès au flux vidéo d'une caméra connectée, à l'activité sportive d'une personne via un traqueur d'activité, a ses habitudes de consommation et de vie via les logs des assistants vocaux, etc.
- **Les risques impactant la sécurité physique des utilisateurs.** Certains dispositifs IoT peuvent effectuer des actions dans le monde physique. Les serrures connectées peuvent par exemple déverrouiller des portes, les machines à laver connectées peuvent contrôler une arrivée d'eau, etc. Les attaques peuvent donc aussi avoir des répercussions dans le monde réel. Par exemple en 2019, Ray et al. ont montré qu'il était possible de déverrouiller les serrures connectées de certains hôtels sans avoir de clef [4].
- **Les risques impactant la sécurité des infrastructures informatiques.** Les dispositifs IoT peuvent aussi être l'objet d'attaques visant à détourner leur comportement pour héberger des services illégaux [21] ou pour servir lors d'attaques de plus grande envergure [5]. Un exemple connu d'attaque utilisant des dispositifs IoT et visant des

infrastructures informatique est le botnet Mirai [6] qui ciblait les caméras connectées afin d'effectuer des attaques de déni de service contre des entreprises telles que Dyn et OVH.

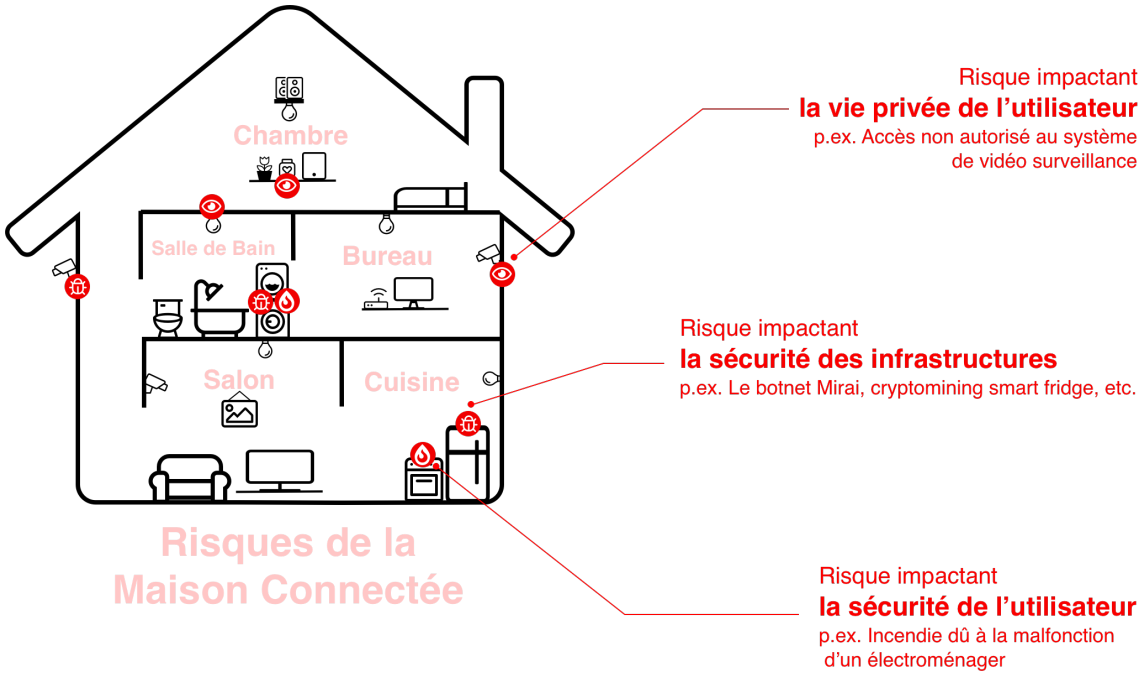


Figure 2.4 Risques liés à la présence d'IoT au sein des ménages

Les vulnérabilités présentent dans les dispositifs IoT mettent en péril la sécurité de l'intégralité du réseau domestique. Il est de ce fait primordial de mettre en place des systèmes de surveillance spécifiques aux dispositifs IoT afin d'assurer la sécurité des maisons connectées. Les différentes attaques de *botnets* tels que Mirai [6] ont en effet montré que les mesures de sécurité actuellement utilisées par les ménages ne sont pas efficaces pour sécuriser les dispositifs IoT.

2.1.5 Solutions de sécurisation des réseaux IoT domestiques

Une solution pour sécuriser les réseaux IoT souvent évoquée dans la littérature est la mise en place d'un système de détection d'intrusions réseau [22–24]. Cependant, peu d'articles proposent une solution pour la génération autonome des politiques de sécurité. Les *Network Intrusion Detection Systems* (NIDS) surveillent le trafic réseau à la recherche de signatures d'attaques connues. Les paquets correspondant à des signatures d'attaques sont alors bloqués et le reste du trafic est autorisé. Bien que cette solution soit adaptée pour les réseaux professionnels disposant d'équipes de sécurité dédiées, elle ne l'est pour les environnements

domestiques. Cette solution nécessite en effet la présence d'experts afin de la paramétrer, de la maintenir et de la monitorer. La liste des signatures d'attaques doit aussi constamment être maintenue à jour afin de détecter les dernières attaques et les alertes générées doivent être analysées afin de s'assurer qu'il ne s'agit pas de faux positifs.

Les NIDS peuvent aussi être augmentés avec du *machine learning* afin de détecter des attaques inconnues. Cependant, l'utilisation de machine learning introduit une certaine incertitude, les faux positifs pouvant être exploités par des attaquants [25]. Ce taux d'incertitude est aussi présent lors de l'identification des modèles de dispositifs [26]. Les similarités comportementales de certains modèles de dispositifs compliquent en effet l'identification des dispositifs à partir de leur trafic réseau et la détermination des politiques de sécurité à appliquer.

Une approche alternative est de créer une liste blanche des comportements autorisés pour les différents dispositifs en se basant sur les spécifications décrivant les comportements nominaux des différents dispositifs. Cette approche est souvent référée comme la détection d'intrusion par spécifications. La police de sécurité utilisée pour sécuriser le réseau décrit alors la liste des comportements autorisés. Cette police peut être fournie par les fabricants directement ou par un tiers de confiance comme proposé par l'*Internet Engineering Task Force* (IETF) dans la RFC8520 [8]. Alternativement, la police peut être générée en observant le comportement des dispositifs localement [7].

Une autre approche est de diviser les dispositifs IoT en deux catégories : les contrôleurs et les non-contrôleurs [27]. Les non-contrôleurs sont alors restreints pour ne pouvoir communiquer uniquement avec l'API cloud et ne sont pas autoriser à communiquer avec d'autres dispositifs, contrairement aux contrôleurs.

Enfin, certaines solutions proposent d'embarquer des capacités de détection d'anomalies au sein des dispositifs [28]. Le firmware des dispositifs est alors mis à jour pour embarquer un agent de détection d'anomalies qui surveille le système et le protège contre de potentielles attaques. Cependant, cette approche nécessite la mise à jour du firmware de tous les dispositifs IoT existants et est de ce fait incompatible avec la majorité des dispositifs déployés, n'étant plus maintenus par les fabricants. De plus cette solution nécessite une forte coopération avec les fabricants afin d'être déployée.

Différentes solutions commerciales existent aussi. L'éditeur d'antivirus Bitdefender propose ainsi la Bitdefender Box [29], une solution se sécurisation tout en un pour les réseaux IoT domestiques. Cependant, les entreprises derrière ces solutions communiquent peu sur les méthodes utilisées pour construire les politiques de sécurité appliquées pour les dispositifs IoT et on peut donc s'interroger sur la compatibilité de ces solutions avec la grande variété de dispositifs disponibles. De plus les tarifs, souvent élevés de ces solutions peuvent être vu

comme un frein pour l'adoption à grande échelle par le grand publique. Enfin la durabilité dans le temps de ces solutions n'est pas garantie à l'image du Norton Core [30] qui n'est plus à la vente depuis le 31 janvier 2019 et qui cessera de fonctionner après le 31 janvier 2022 [31].

2.2 Systèmes de détection et de prévention d'intrusion

Les IDS sont des systèmes utilisés pour détecter les actions malveillantes sur les réseaux privés. Ils sont généralement opérés par des équipes de sécurité formées pour prendre des actions en cas d'alerte. Il existe trois différents types d'IDS [32] :

- **Les IDS par signatures** analysent le trafic réseau à la recherche d'attaques connues. Afin de détecter les attaques, les IDS par signatures se basent sur une liste de descriptions d'attaques (appelées signatures) établie par des experts. Cette méthode de détection est efficace uniquement contre les attaques connues déjà répertoriées.
- **Les IDS par spécifications** utilisent comme référence une liste de comportement normaux (appelées spécifications). Ils détectent tout comportement déviant des spécifications. Les spécifications sont définies par des experts.
- **Les IDS par anomalies** détectent les actions anormales. Pour ce faire, ils profilent le comportement normal sur le réseau pendant une période d'observation. À l'issue de cette période d'observation toute action déviante du profil est considérée anormale. Contrairement aux deux autres types, les IDS par anomalies nécessitent une période d'entraînement pour construire les profils et sont sujets aux faux positifs (actions normales qui n'ont pas été observées durant le profilage).

Les IDS commerciaux ne sont pas adaptés à une utilisation domestique, car ils nécessitent généralement une équipe d'expert pour opérer. Ainsi les listes de signatures d'attaques des IDS par signatures doivent être régulièrement mises à jour par les experts afin de pouvoir détecter les attaques les plus récentes. Les IDS par spécifications nécessitent quant à eux une mise à jour des spécifications par les experts à chaque modification du système protégé. Enfin, les IDS par anomalies nécessitent d'être supervisés par des experts durant la période d'observation. La période d'observation doit aussi être répétée à chaque modification du système surveillé afin de profiler son nouveau comportement.

2.3 Blockchain

Cette section introduit les concepts de base de la blockchain. Des explications plus en détail peuvent être trouvées dans l'étude de Ruoti et al. [33].

La blockchain est une technologie permettant la collaboration de plusieurs parties, sans relation de confiance, au sein d'un même système. La blockchain fournit en effet une gouvernance partagée où les participants collaborent pour décider ce qui est ajouté dans la chaîne et assure que le protocole est appliqué correctement par tous les participants.

Un aspect majeur des blockchains est leur vérifiabilité : les données stockées dans une blockchain sont le résultat de l'exécution d'un algorithme de consensus et ont été acceptées par l'ensemble des participants. Seules les données ayant été acceptées par le consensus sont ajoutées dans la chaîne qui ne contient ainsi que des données vérifiées. Lorsque les données ont été vérifiées par les participants du système, un nouveau bloc contenant ces données est créé et ajouté à la chaîne. Ce bloc possède une empreinte cryptographique qui le lie au bloc précédent, permettant à toutes les parties de vérifier la continuité de la chaîne de blocs.

Une autre propriété des blockchains est la prévention contre la perte de données. Grâce à la nature décentralisée des blockchains, leurs données sont répliquées chez tous les participants ce qui facilite la récupération en cas de perte de données. Un participant peut à tout moment demander une copie de la chaîne complète et vérifier la validité de son contenu. Grâce à ces propriétés, la technologie blockchain fournit un registre décentralisé inviolable pouvant être utilisé au-delà des cryptomonnaies, pour les applications nécessitant de la responsabilité, transparence et confiance dans les données. Les blockchains sont ainsi adaptées pour les applications ouvertes où des participants sans relations de confiances doivent s'entendre pour former un consensus.

2.3.1 Types de blockchain

Il existe deux différents types de blockchains répondant à différents besoins :

— **Les blockchains ouvertes**, telles que celle utilisée par le Bitcoin. Concrètement, une blockchain est un registre répliqué et synchronisé entre différents participants d'un réseau pair à pair. Ce registre peut être utilisé pour stocker des transactions monétaires comme dans le cas du Bitcoin ou toutes autres données critiques nécessitant une structure de données décentralisée fiable. Contrairement aux bases de données classiques, une blockchain permet de stocker quelques méga-octets de données qui ne peuvent pas être modifiées. La participation à une blockchain dépend du type de blockchain. Ces blockchains sont publiques et n'importe qui peut rejoindre le réseau

pour contribuer et consulter les données qu'elles contiennent. Ce type de blockchain est particulièrement utilisé pour les projets open sources publics. De par leur nature décentralisée, les blockchains ouvertes permettent de supprimer les intermédiaires de confiance.

— **Les blockchains fermées** sont gérées par une autorité centralisée qui assigne les rôles permettant de contribuer ou de consulter la blockchain. Les blockchains fermées peuvent être consultables par tous, mais nécessiter une autorisation particulière pour pouvoir contribuer. Ce type de blockchain est utilisé dans l'industrie lorsqu'un contrôle précis des accès et des rôles des différents acteurs interagissant avec la blockchain est requis.

2.3.2 Algorithme de consensus

L'algorithme de consensus utilisé par une blockchain est de ce fait un aspect primordial : il assure que la chaîne de blocs contenant les données est constamment synchronisée entre les participants de manière à ce qu'ils en possèdent tous une copie identique à tout moment. L'algorithme de consensus permet aussi d'empêcher la blockchain d'évoluer trop rapidement en introduisant un délai entre la création de nouveaux blocs. Il existe de nombreux algorithmes de consensus [34]. Les deux principaux sont *Proof of Work* (PoW) et *Proof of Stake* (PoS). Ils utilisent une approche probabiliste pour valider les blocs.

— **L'algorithme de PoW [35, 36]** est largement utilisé par les blockchains ouvertes telles que Bitcoin et Ethereum. Cet algorithme nécessite que les empreintes cryptographiques des blocs soient inférieures à une certaine valeur afin de sélectionner aléatoirement un participant pour créer un nouveau bloc et l'ajouter à la chaîne. Le poids de chaque participant dans le procédé de validation est alors déterminé par sa capacité à calculer un grand nombre d'empreintes cryptographiques rapidement. Cette approche est de ce fait très coûteuse en puissance de calcul et énergétiquement, car l'effort fourni par les participants n'a pas d'autre utilité que de délayer aléatoirement leur capacité à produire un bloc valide. Un autre inconvénient de cet algorithme est qu'un participant possédant une grande puissance de calcul peut dominer le réseau et prendre le contrôle de la blockchain.

— **L'algorithme de PoS [37]** sélectionne aléatoirement le participant pour créer un nouveau bloc parmi les participants ayant verrouillé un certain montant de cryptomonnaies. Le poids des participants dépend alors du montant de cryptomonnaies verrouillé. Contrairement à la PoW, la PoS ne se base pas sur les empreintes cryptographiques et est donc moins énergivore. D'autre part, une attaque par un participant est quasiment impossible, car celui-ci s'expose à la perte du montant de cryptomonnaies

verrouillé. Dans le cas des principales blockchains, ce montant équivaut à des dizaines de milliers de dollars. Un participant souhaitant prendre le contrôle de la blockchain devrait ainsi posséder 51% des cryptomonnaies de la blockchain. Ce principe réduit fortement la probabilité d'une attaque de ce type et est un avantage par rapport à la PoW où il est plus aisé de prendre le contrôle de 51% de la puissance de calcul du réseau. Cependant, PoS nécessite une cryptomonnaie intégrée à la blockchain pour fonctionner.

2.3.3 Systèmes de détection d'intrusion pour réseaux IoT basés sur la blockchain

Golomb et al. [28] proposent CIOTA, un système monitorant le comportement des dispositifs IoT au niveau de leur firmware afin de former un registre d'actions légitimes. Cette solution collecte ainsi des données des dispositifs sous forme d'Extensive Markov Model (EMM) afin de construire un modèle de leurs comportements. Le système utilise la blockchain pour créer un registre de ces modèles ainsi que pour informer un système de détection d'intrusion côté client. Bien que cette solution parait prometteuse, elle nécessite la modification du firmware des dispositifs pour déployer un agent EMM.

Mendez et al. [38] ont développé et évalué une preuve de concept basée sur la blockchain Ethereum afin de protéger les réseaux domestiques. Leurs *gatekeepers* filtrent le trafic à la bordure du réseau en se basant sur une liste blanche des actions autorisées. La liste blanche est obtenue en se basant sur les informations stockées dans un *smart contract* Ethereum. Cette approche souffre de différents problèmes : Ce dispositif ne résout pas le problème de la création de la politique de sécurité qui doit être manuellement stockée dans le *smart contract*. Ce concept repose aussi sur un système centralisé, car les *gatekeepers* interagissent avec un unique *smart contract* qui pourrait être remplacé par une base de données. Enfin cette solution n'est pas adaptée à un usage réel, car son déploiement dans la blockchain Ethereum publique nécessiterait le paiement de *fees* à chaque interaction des *gatekeepers* pour mettre à jour les données du *smart contract*.

2.3.4 Conclusion

Le nombre croissant de dispositifs IoT et leur adoption rapide posent différents problèmes de sécurité pour les réseaux des maisons connectées. Beaucoup de ces dispositifs sont en effet vulnérables et menacent la vie privée de leurs utilisateurs, leur sécurité physique ainsi que la sécurité des infrastructures informatiques. De plus, les solutions de sécurité actuellement déployées dans les maisons connectées ne sont pas adaptées à ce nouveau type de dispositifs. Les dispositifs IoT bénéficient de ce fait souvent de trop de privilèges et sont ainsi en mesure

de contacter tous les hôtes sur internet ce qui profite aux acteurs malveillants et aux *botnets*. Bien que de nombreuses solutions de sécurité existent, aucune ne semble être adaptée à ce problème. Les solutions classiques tels que les NIDS nécessitent en effet la plupart du temps une équipe d'experts pour les paramétrer et gérer les alertes de sécurité. Les solutions proposées dans la littérature académique souffrent quant à elles de problèmes de compatibilité avec les dispositifs existants ou ne sont pas adaptées pour utilisation à grande échelle par les ménages.

CHAPITRE 3 SERENIoT

Nous avons vu dans le chapitre 2 que la majorité des solutions de sécurité actuelles ne conviennent pas aux réseaux IoT domestiques car les utilisateurs ne sont pas en mesure de maintenir une infrastructure de sécurité complexe. En effet, les solutions exposées dans le chapitre précédent présentent plusieurs limites : d'une part, elles requièrent un paramétrage complexe et une maintenance régulière pour assurer leur bon fonctionnement, et d'autres part, elles ne sont compatibles qu'avec un nombre restreint de dispositifs ou sont difficilement déployables à grande échelle. Aussi, pour remédier à ces problématiques, nous avons développé SERENIoT, un système novateur basé sur une blockchain capable de construire des politiques de sécurité pour dispositifs IoT sans intervention humaine. Dans le présent chapitre nous détaillons le fonctionnement de notre système SERENIoT.

3.1 Vue d'ensemble de la solution

SERENIoT (prononcé *Serenity*) est un système de détection d'intrusion collaboratif pour réseau IoT domestique.

Principe global

SERENIoT surveille le trafic réseau des dispositifs IoT afin de détecter et de bloquer les paquets anormaux. Pour ce faire, un réseau WiFi dédié aux IoT est déployé et contrôlé à partir d'un dispositif appelé «sentinelle» qui assure le filtrage du trafic et sert d'interface avec le réseau LAN. La politique de sécurité appliquée par la sentinelle est stockée dans un registre décentralisé.

Architecture

La figure 3.1 montre la topologie d'un réseau domestique intégrant une sentinelle. La sentinelle est placée entre les dispositifs et le routeur du fournisseur d'accès internet. les sentinelles émettent un réseau WiFi auquel sont connectés les dispositifs IoT, permettant ainsi la surveillance et le filtrage du trafic réseau entre les dispositifs IoT et internet. Concrètement, ce réseau sans-fil établit un pont vers le réseau local domestique (LAN) afin que l'intégralité du trafic réseau entrant et sortant des sentinelles soit analysée. Les sentinelles ont été conçues pour être déployées sur les équipements réseau domestiques tels que les routeurs. Une configuration typique est de déployer une sentinelle par maison connectée.

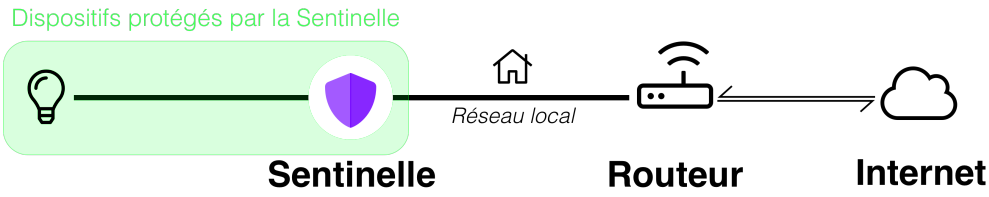


Figure 3.1 Topologie des réseaux domestiques SERENIoT

Procédé

L'identification des paquets anormaux par la sentinelle se base sur un registre décentralisé contenant la liste des signatures des paquets autorisés. Cette liste caractérise le comportement attendu des différents dispositifs IoT surveillés et correspond à la politique de sécurité appliquée. Les sentinelles collaborent pour déterminer le comportement réseau attendu des différents dispositifs IoT. Ainsi, les sentinelles qui composent le réseau SERENIoT bénéficient toujours implicitement des dernières mises à jour du registre contenant les spécifications de sécurité à appliquer, contrairement aux solutions de sécurité classiques telles que les systèmes de détection d'intrusion par signatures.

Concrètement, les sentinelles ne transfèrent que les paquets réseau dont la signature est listée dans le registre décentralisé. Ces listes de signatures sont appelées *listes blanches*. Chaque liste blanche spécifie les signatures des paquets réseau caractérisant le comportement d'un dispositif IoT, tel qu'observé par la majorité des sentinelles. Les listes blanches pour tous les différents dispositifs IoT sont stockées dans la blockchain SERENIoT. L'utilisation d'une blockchain permet à SERENIoT d'être totalement décentralisé et de pouvoir démarrer avec un petit nombre de sentinelles.

Avantages de la solution

Grâce à son architecture pair-à-pair et à l'utilisation d'un registre décentralisé, SERENIoT fonctionne de manière autonome et ne requiert aucune action de la part des utilisateurs. L'utilisation d'une blockchain permet au système d'être totalement indépendant des fabricants de dispositifs IoT et de ne pas reposer sur un tiers de confiance, ce qui favorise le support d'une grande variété de dispositifs IoT. Le recours à la blockchain est développé dans la section 3.5. SERENIoT est compatible avec la majorité des dispositifs IoT IP existants et ne nécessite aucune modification de leur hardware ou firmware.

Limites d'application

SERENIoT a été conçu spécifiquement pour les dispositifs IoT grand public pouvant se connecter à un réseau WiFi. Lors de notre évaluation, nous avons observé de bonnes performances avec les dispositifs ayant un jeu de fonctionnalité limité tels que les *smart bulb*, *smart plug*, *smart lock*, *smart thermostat*, etc. Ces dispositifs n'interagissent en effet qu'avec un nombre limité de services cloud via des API spécifiques connues. Leur empreinte réseau peut de ce fait être déterminée avec précision. D'après une étude de 2019 [1], ce type de dispositifs représente environ 41% des dispositifs déployés en nord Amérique et 28.4% des dispositifs en Europe de l'Ouest [1]. SERENIoT ne supporte pas les dispositifs ayant une empreinte réseau variant en fonction de leur utilisation, car chaque dispositif peut générer une empreinte réseau unique. La figure 3.2 montre le spectre des dispositifs supportés par SERENIoT. Du coté gauche, les dispositifs ont une empreinte réseau plus «simple» qui a tendance à être partagée entre les dispositifs d'un même modèle. Du coté droit, les dispositif ont une empreinte réseau unique déterminée par l'usage qu'en font leurs utilisateurs.

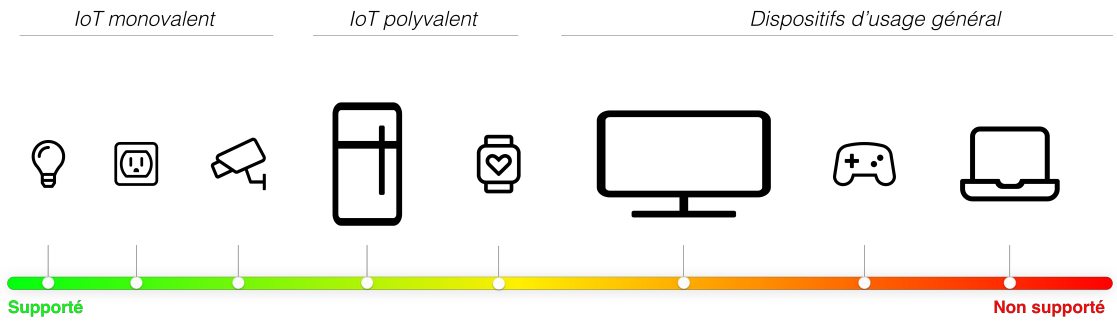


Figure 3.2 Dispositifs supportés par SERENIoT

1. Nous considérons ici tous les dispositifs de l'étude de Kumar et al. [1], à l'exception des consoles de jeux, des stations multimédias et des dispositifs de stockages de fichiers qui sont fonctionnellement aussi complexes que des ordinateurs personnels.

3.2 Threat model

SERENIoT protège les dispositifs IoT contre les attaques ayant pour objectifs de modifier leur comportement, telles que les attaques de botnet [6,39]. Plus spécifiquement, SERENIoT a été conçu pour protéger les dispositifs IoT dans les deux scénarios suivants :

- Un dispositif IoT a été compromis localement par un malware essayant de changer son comportement pour accomplir des actions malveillantes. Le vecteur d'attaque peut varier : le dispositif peut avoir été infecté par un autre dispositif sur le réseau local (par exemple par un ordinateur infecté), ou l'infection peut être le résultat d'une action physique sur le dispositif (par exemple un changement de carte mémoire). Dans cette situation, SERENIoT se comporte comme un pare-feu et bloque tout le trafic réseau issu du dispositif qui ne correspond pas aux spécifications établies dans la liste blanche.
- Un dispositif IoT est directement exposé sur internet. SERENIoT protège alors le dispositif contre les attaques en bloquant toutes les connexions entrantes ne correspondant pas aux spécifications. La plupart des dispositifs IoT ne reçoivent pas normalement de connexions entrantes depuis internet et SERENIoT se comporte alors comme un pare-feu, bloquant toutes les connexions entrantes.

Une fois qu'une liste blanche est établie pour un dispositif, un attaquant doit changer le comportement de plus de 50% des dispositifs IoT du même modèle afin de modifier les spécifications de SERENIoT et de permettre à l'attaque d'aboutir.

3.3 Architecture des sentinelles

Les sentinelles sont composées de 4 modules principaux :

- **Le filtre réseau (1)** qui est en charge d'appliquer la politique de sécurité en bloquant les paquets réseau qui ne sont pas autorisés.
- **Le module de signature des paquets (2)** qui sérialise les paquets IP en signatures textuelles.
- **Le module de politiques de sécurité (3)** qui liste les signatures de paquets autorisées
- **Le module blockchain (4)** qui permet de garder la politique de sécurité à jour en synchronisant le registre avec les autres sentinelles et en reportant toute signature de paquet inconnue.

La figure 3.3 représente l'architecture des sentinelles.

Les sentinelles sont placées entre les dispositifs IoT et le routeur d'accès internet de manière à pouvoir intercepter le trafic réseau entre les dispositifs et internet. Pour ce faire, les sentinelles

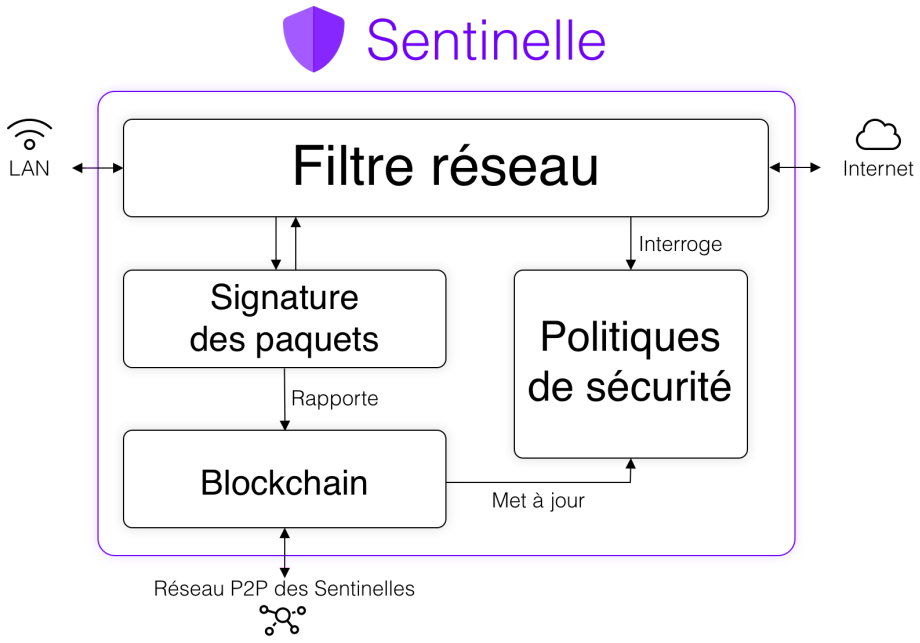


Figure 3.3 Architecture des sentinelles

embarquent un point d'accès WiFi et émettent un réseau sans fil dédié aux dispositifs IoT.

3.4 Calcul des signatures de paquets

Principe

Les signatures de paquets permettent aux sentinelles de caractériser les paquets inspectés. Un algorithme de signature efficace doit de ce fait pouvoir répondre à différent critères :

- **Unicité :** Produire pour un même dispositif des signatures différentes pour les paquets issus de connexions différentes.
- **Consistance :** Produire des signatures identiques pour les paquets émanant de dispositifs d'un même modèle.

Contrairement aux ordinateurs personnels dont l'empreinte réseau varie d'une utilisation à l'autre en fonction des actions de l'utilisateur, les dispositifs IoT d'un même modèle se comportent de manière similaire [2] et produisent donc des empreintes réseau similaires. Nous avons vérifié cette hypothèse durant notre évaluation dans la section 4.2. Notre prototype se base sur des signatures de paquets au niveau IP et se focalise sur les champs qui demeurent constants pour les dispositifs d'un même modèle.

2. Les dispositifs IoT d'un même modèle se comportent souvent de manière similaire en termes de trafic et de destination des paquets

Procédé

Nous avons choisi d'utiliser une approche similaire à NetFlow [3] afin d'avoir un compromis entre unicité et consistance lors de la génération des signatures de paquets IP.

Les signatures utilisées par SERENIoT agrègent les séquences de paquets partageant les champs suivants :

— Protocole de la *payload* IP
— *Endpoint* (Nom de domaine ou adresse IP si le nom de domaine n'est pas disponible)
— Port utilisé par le service

Endpoint identifie l'hôte distant avec lequel le dispositif interagit. Afin de résoudre le potentiel nom de domaine, nous effectuons des requêtes *reverse DNS lookup*. Le port utilisé par le service identifie le service de la connexion (par exemple 80 pour HTTP). Le port utilisé par le service réfère généralement au numéro de port de l'hôte distant. Cependant, certains dispositifs IoT (par exemple les caméras) hébergent un service localement. Dans ce cas, le port utilisé par le service indique le numéro de port utilisé par le dispositif IoT. Afin de différencier les ports de services locaux et distants, nous ajoutons au numéro de port une lettre pour identifier la direction : L pour local, R pour disant. Les paquets d'une même connexion partagent une même signature qui est utilisée par les sentinelles pour identifier les paquets autorisés et interdits. Les signatures sont calculées de la manière suivante :

$$Signature = H(protocole, endpoint, port\ du\ service)$$

Les signatures n'incluent pas d'identifiants spécifiques aux dispositifs tels que le *Media Access Control Organizationally Unique Identifier* (OUI). Tous les dispositifs d'un même modèle ne partagent en effet pas forcément le même OUI car de nombreux fabricants possèdent plus d'un OUI. Deux dispositifs identiques avec deux OUI différents seraient assignés à deux chaînes différentes ce qui réduirait la sécurité des deux chaînes. Un code malveillant exécuté sur un dispositif IoT peut être capable de modifier le OUI du dispositif. Notre choix de signature de paquets permet de différencier les paquets vers des hôtes non fiables de ceux allant vers l'API des fabricants. Il permet aussi de différencier les paquets envoyés par les dispositifs protégés de ceux envoyés par un hôte distant dans le cas des réseaux IPv6 ou des réseaux sans NAT où les dispositifs sont directement exposés sur internet. SERENIoT n'identifie pas précisément les dispositifs. Les dispositifs produisant les mêmes signatures de paquets sont groupés ensemble et le système assume qu'ils sont du même modèle. Les dispositifs sont caractérisés par les signatures générées par les paquets qu'ils émettent et les modèles des

3. CISCO NetFlow est une technologie permettant d'analyser les flux réseau [40].

dispositifs sont identifiés par leur empreinte réseau en calculant le haché de la liste triée de leurs signatures de paquets.

Limites

Il est à noter que cette méthode de calcul de signatures ne permet pas l'analyse du trafic entre les dispositifs du réseau local. En effet, les adresses IP assignées aux dispositifs sur les réseaux locaux varient d'un réseau à l'autre et ne sont jamais constantes entre les dispositifs d'un même modèle. Ainsi lorsque deux dispositifs du réseau local communiquent, il n'est pas possible d'identifier un *Endpoint* constant afin de calculer une signature commune à tous les dispositifs du modèle. Cette restriction pourrait être contournée en utilisant une autre méthode de calcul des signatures réseau ne prenant pas en compte l'adresse IP de l'hôte avec lequel le dispositif communique. Cependant, nous ne sommes pas certains qu'un tel algorithme de calcul de signature soit en mesure de différencier les connexions vers des hôtes malveillants.

Notre approche se base sur des champs des paquets IP qui ont des valeurs similaires pour tous les dispositifs d'un même modèle. Notre méthode pourrait être complétée en analysant plus en détail le contenu des *payloads* IP, cependant notre approche s'est montrée suffisante pour mener à bien nos expériences et pour implémenter un prototype de système de détection d'intrusion.

3.5 La blockchain SERENIoT

La blockchain est la clef du mécanisme collaboratif de génération des politiques de sécurité de SERENIoT. La blockchain permet la validation des signatures de paquets listés dans les listes blanches par un consensus décentralisé. Cette approche garantit la robustesse du système et la validité des politiques de sécurité en assurant qu'aucune signature malveillante ne soit inscrite dans une liste blanche. Ceci, sous réserve que la majorité des sentinelles observe des comportements légitimes sur leurs dispositifs locaux.

Nous avons choisi d'utiliser une blockchain afin de satisfaire les trois exigences suivantes :

- Avoir un fonctionnement indépendant de SERENIoT. La blockchain permet de répartir l'hébergement des données sur tous les utilisateurs. Cela permet d'être indépendant de tout système tiers.
- Déployer à moindre coût un système hautement disponible fournissant des mises à jour de politiques de sécurité en temps réel.
- Rendre le système accessible sans restriction d'utilisation à certaines marques de dis-

positifs.

Nous avons développé notre propre blockchain afin d'implémenter SERENIoT. En effet, l'utilisation d'une blockchain de cryptomonnaie aurait impliqué une utilisation payante du système. De même, l'utilisation d'une blockchain fermée aurait nécessité une entité racine pour gérer le système ce qui ne permet pas d'avoir un système indépendant et libre d'accès. La blockchain publique créée pour SERENIoT se base sur la technologie de la blockchain du Bitcoin, la plus accessible et la plus rapide à mettre en oeuvre pour un prototype. Elle permet de choisir les fonctionnalités répondant le mieux à notre besoin tout en évitant des problèmes de compatibilités potentiels lors de l'ajustement d'un framework à notre cas d'utilisation.

3.5.1 Registre décentralisé

Le registre de SERENIoT liste les signatures de paquets reportés par les sentinelles. Le registre se base sur un mécanisme d'horodatage décentralisé, chaînant les blocs de données. Ce mécanisme assure que les blocs ne peuvent pas être réarrangés ni modifiés sans invalider tous les blocs suivants dans la chaîne. Lorsque la blockchain grandit, les sentinelles convergent sur la plus longue chaîne de blocs (celle ayant le plus de blocs). Dans notre implémentation, les blocs stockent la liste des signatures de paquets réseau reportés par les sentinelles, contrairement à la racine de l'arbre de Merkle des transactions dans le cas du Bitcoin. La liste complète des signatures est en effet nécessaire pour construire la liste blanche. La figure 3.4 représente la structure des blocs utilisée dans SERENIoT. L'en-tête des blocs est composé de 6 champs et le corps du bloc contient la liste des signatures reportées par la sentinelle. Les en-têtes des blocs contiennent les champs suivants :

— **Le haché du bloc précédent.** Ce haché permet de chaîner le bloc à son bloc parent dans la chaîne. Les hachés des blocs sont des identifiants uniques qui correspondent à l'empreinte numérique des champs des entêtes des blocs. Chaque haché de bloc est ainsi calculé en se basant sur le haché du bloc parent ce qui garantit que la chaîne ne peut pas être brisée sans avoir à recalculer tous les blocs suivants dans la chaîne. Le premier bloc dans la chaîne, aussi appelé bloc genèse, n'a pas de bloc parent et a donc le champ du haché du bloc parent vide.

— **L'adresse de la sentinelle.** L'adresse est un identifiant unique calculé au démarrage à l'aide de l'algorithme utilisé pour générer les adresses Bitcoin. Cela permet d'obtenir une adresse unique par sentinelle. Les adresses des sentinelles sont utilisées pour identifier les sentinelles au sein du système.

— **L'index du bloc.** L'index correspond au numéro du bloc dans la chaîne. Le premier bloc de la chaîne a l'index 0. Deux blocs peuvent avoir le même index dans le cas où la chaîne fork (c.-à-d. la chaîne est divisée en plusieurs branches). Cependant lorsque le

fork est résorbé, seule la plus longue chaîne est conservée et les blocs possèdent alors un unique index.

— **L' identifiant de la chaîne.** L'identifiant de la chaîne indique la blockchain à laquelle appartient le bloc. Cet identifiant est utilisé pour le multi-chain tel que décrit dans la section 3.5.3.

— **Le timestamp.** Le timestamp correspond à la date à laquelle le bloc a été créé.

— **Le nounce.** Le nounce est utilisé lors du processus de *proof of work* afin de faire varier le haché du bloc jusqu'à obtenir un haché valide.

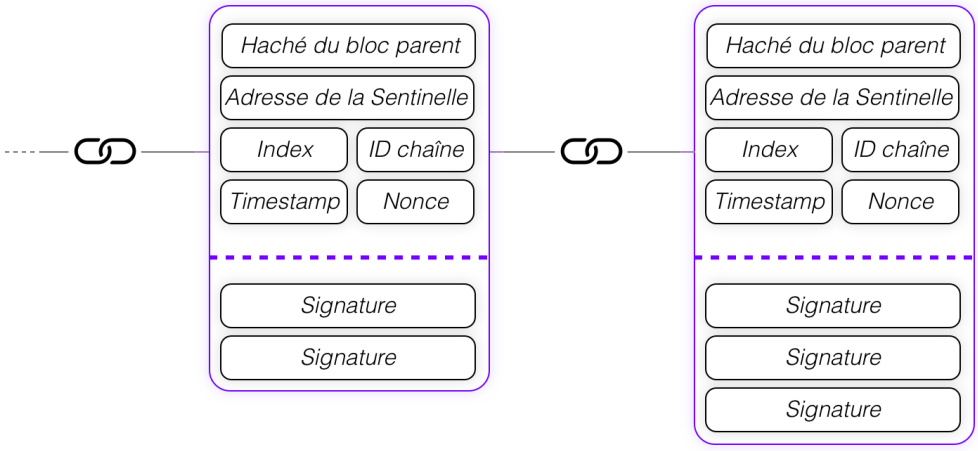


Figure 3.4 Structure des blocks SERENIoT

Afin de reporter de nouvelles signatures au système, les sentinelles ajoutent de nouveaux blocs à la blockchain. Pour ce faire, les sentinelles considèrent seulement les blocs contenant des signatures de paquets déjà observés. Les sentinelles évitent ainsi d'ajouter des blocs aux chaînes incluant des signatures inconnues. Les signatures inconnues peuvent en effet être malveillantes ou refléter un comportement jusque là inconnu pour un dispositif. Ainsi la chaîne grandissant le plus vite contient toujours les signatures de paquets les plus communément observés par une majorité des sentinelles. Ce mécanisme est décrit en détail dans la section 3.5.4. La figure 3.5 représente une blockchain pour un modèle de dispositif. Les sentinelles ajoutent des signatures de paquets dans des blocs. Lorsque la blockchain grandit, seules les signatures listées dans la plus grande chaîne sont considérées de confiance.

La liste blanche est constituée de l'ensemble des signatures de paquets autorisés, contenues dans les blocs confirmés depuis l'initialisation de la chaîne. La liste blanche peut contenir des signatures de paquets autorisant des comportements n'étant plus utilisés par les dispositifs (par exemple dans le cas de la suppression d'une fonctionnalité au cours d'une mise à jour).

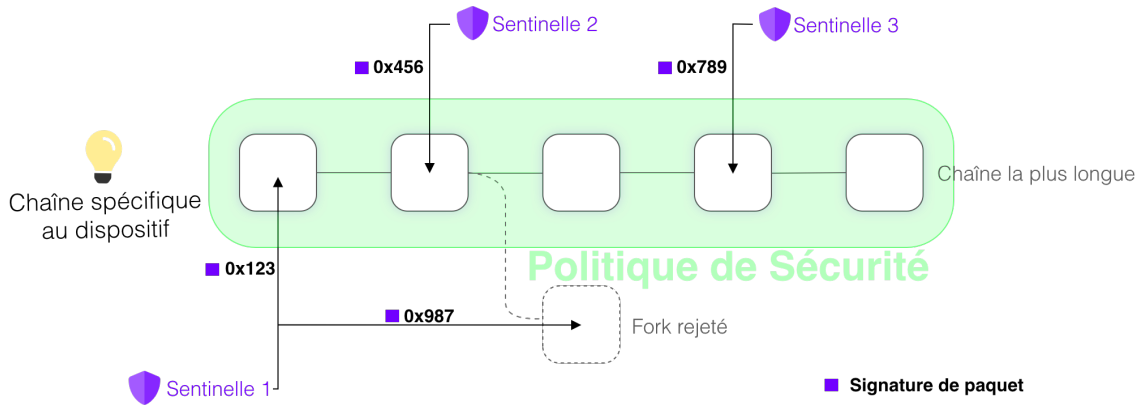


Figure 3.5 Blockchain pour un dispositif

La suppression des signatures obsolètes n'a pas été considérée dans cette recherche mais pourrait être explorée lors de travaux futurs.

3.5.2 Consensus

L'algorithme de consensus utilisé par SERENIoT assure que la liste blanche est synchronisée en permanence entre toutes les sentinelles afin qu'elles convergent des copies identiques de la blockchain. Il assure aussi que la chaîne grandissant le plus vite regroupe le plus de sentinelles et permet de se prémunir contre les attaques Sybil [41].

De nombreux algorithmes de consensus ont été développés au cours des dernières années pour les besoins des différentes blockchains publiques. Nous en avons présenté certains dans la section 2.3. Afin de faciliter le développement d'un prototype, nous avons choisi d'utilisé l'algorithme de *proof of work* [42] (PoW) lors de la conception de SERENIoT. L'algorithme de *proof of work* est largement utilisé par les blockchains ouvertes telles que Bitcoin et Ethereum. Cet algorithme nécessite que les empreintes cryptographiques des blocs soient plus petites qu'une certaine valeur afin de sélectionner aléatoirement un participant pour créer un nouveau block et l'ajouter à la chaîne. Le poids de chaque participant dans le procédé de génération de blocs est alors déterminé par sa capacité à calculer un grand nombre d'empreintes cryptographiques rapidement. Nous discutons des algorithmes alternatifs dans la section 5.

Avec l'algorithme de *proof of work*, la sélection aléatoire d'une sentinelle est basée sur la rapidité de résolution de puzzles informatiques. La sentinelle sélectionnée est celle qui résout le puzzle le plus rapidement. Elle ajoute alors son block à la blockchain. Cette opération est réitérée pour construire la blockchain. Chaque bloc ajouté à la chaîne augmente l'effort

nécessaire pour réécrire la plus longue chaîne. En effet, la réécriture d'un bloc précédent dans la chaîne nécessite de recalculer la *proof of work* pour tous les blocs le suivant. Ainsi la plus longue chaîne regroupera toujours le plus de sentinelles tant que la puissance de calcul est équilibrée entre les différentes sentinelles du réseau.

3.5.3 Support de plusieurs dispositifs

La solution décrite précédemment avec une unique blockchain fonctionne pour un seul modèle de dispositif IoT. En effet, les sentinelles ne peuvent pas se prononcer sur les blocs contenant des signatures pour des dispositifs inconnus et chaque modèle de dispositif nécessite donc une blockchain/liste blanche spécifique. Pour résoudre ce problème, SERENIoT utilise un mécanisme de *multichain* permettant aux sentinelles de souscrire aux listes blanches concernant les dispositifs qu'elles protègent. Ainsi, chaque modèle de dispositif dispose d'une blockchain séparée pour traquer l'évolution de son comportement. Lorsqu'un nouveau dispositif est connecté à une sentinelle, la sentinelle le profile et l'assigne à la blockchain groupant les dispositifs du même modèle. Pour profiler les dispositifs, les sentinelles observent le comportement des dispositifs pendant une courte période de profilage. Lorsque la phase de profilage est terminée, les sentinelles calculent l'identifiant des listes blanches correspondant aux dispositifs. Les identifiants de liste blanche sont déterminés en calculant l'empreinte numérique de la liste triée des signatures enregistrées durant la période de profilage. SERENIoT n'identifie pas précisément les dispositifs. Les dispositifs produisant les mêmes signatures sont groupés et le système assume qu'ils sont du même modèle. Ainsi les dispositifs sont caractérisés par leurs signatures, les empreintes des différents modèles sont réalisées en calculant le haché de la liste de leurs signatures.

Pour supporter différents modèles de dispositifs, la blockchain de SERENIoT est composée d'une chaîne de contrôle et des chaînes spécifiques aux modèles de dispositifs (appelées aussi listes blanches). Il y a une chaîne spécifique par modèle protégé par les sentinelles. Dans notre implémentation, la chaîne de contrôle stocke les en têtes des blocs valides issus des listes blanches. La chaîne de contrôle utilise l'algorithme de consensus *proof of work*. Les chaînes spécifiques n'ont pas d'algorithme de consensus particulier et s'appuient sur le *proof of work* de la chaîne de contrôle. La figure 3.7 montre l'architecture multichain de SERENIoT. Les empreintes numériques des en-têtes des blocs des chaînes spécifiques aux dispositifs sont listées et validées dans la chaîne de contrôle.

La chaîne de contrôle améliore la robustesse du système en obligeant toutes les sentinelles à contribuer à une même chaîne principale en parallèle des chaînes spécifiques aux dispositifs. Il est alors plus difficile pour un acteur malveillant de cibler une chaîne peu populaire pour

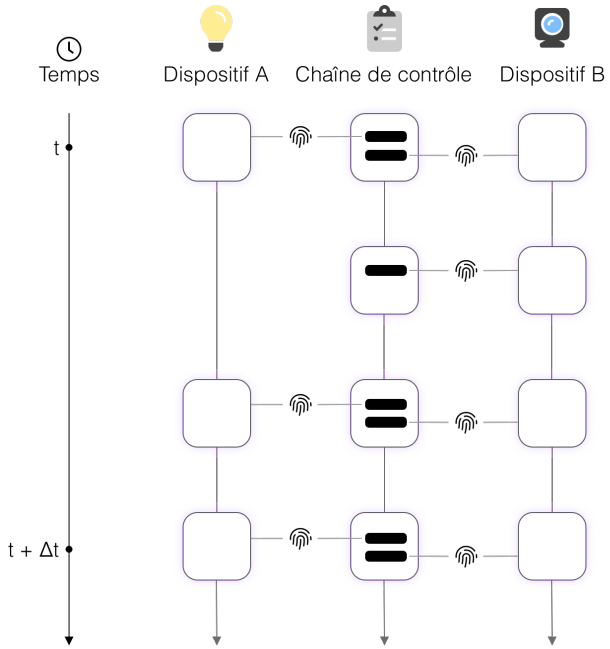


Figure 3.6 L'architecture «Multichain» de SERENIoT

la réécrire. En effet, tous les blocs des chaînes spécifiques aux dispositifs sont validés dans la chaîne de contrôle qui regroupe toutes les sentinelles du réseau.

3.5.4 Workflow des sentinelles

Les sentinelles maintiennent et mettent à jour les listes blanches et se comportent comme des noeuds de la blockchain. Les sentinelles souscrivent uniquement aux listes blanches correspondant aux dispositifs qu'elles surveillent sur leur réseau local. Le processus des sentinelles peut être résumé comme suit :

1. La sentinelle surveille les dispositifs qui lui sont connectés et enregistre les nouvelles signatures de paquets dans des blocs candidats. Un bloc candidat est créé pour chaque liste blanche souscrite. Si un dispositif est inactif ou qu'aucune nouvelle signature n'est enregistrée, les sentinelles produisent un bloc vide.
2. La sentinelle calcule les hachés des en têtes des blocs candidats et les ajoute dans le bloc candidat pour la chaîne de contrôle.
3. Les sentinelles essayent de résoudre la *proof of work* pour le bloc candidat pour la chaîne de contrôle.
4. La première sentinelle à résoudre la *proof of work* est sélectionnée pour ajouter ses blocs aux listes blanches correspondantes. Pour ce faire, la sentinelle diffuse le bloc

pour la chaîne de contrôle ainsi que ses blocs pour les listes blanches aux autres sentinelles.

5. Les sentinelles acceptent toujours les blocs de contrôle. Les sentinelles acceptent les blocs de liste blanche si elles ont souscrit à la même liste blanche et si elles reconnaissent les signatures de paquets qu'ils contiennent. Lorsqu'une sentinelle accepte un bloc, elle travaille pour étendre la chaîne à la suite de ce bloc. Un bloc de liste blanche est valide uniquement si le haché de son en-tête est listé dans un bloc de contrôle. Les sentinelles convergent toujours sur la plus longue chaîne et les *forks* sont résolus lorsqu'une branche devient plus longue que les autres.

Les sentinelles produisent des blocs vides lorsque les comportements de leurs dispositifs n'évoluent pas afin d'indiquer qu'aucune nouvelle signature n'a été enregistrée. Si la majorité des sentinelles produisent des blocs vides, aucune signature ne sera ajoutée aux blockchains correspondantes.

3.5.5 Incitatifs

La blockchain SERENIoT a été conçue pour fonctionner sans cryptomonnaie intégrée. Ainsi, les sentinelles qui contribuent au système en fournissant de la puissance de calcul ne peuvent pas être récompensées avec de la cryptomonnaie. Afin d'encourager les sentinelles à rester actives et à contribuer au système, les sentinelles inactives sont isolées par leurs voisins et ne reçoivent pas les dernières mises à jour des listes blanches. Pour signaler leur activité et leur contribution au réseau, les sentinelles utilisent un mécanisme inspiré des *mining pool*[4] et diffusent les solutions partielles pour la *proof of work* qu'elles essayent de résoudre. Cela permet de prouver à leurs voisins qu'elles sont actives et qu'elles contribuent au réseau.

3.6 Détection des évolutions comportementales

SERENIoT a été conçu pour protéger les dispositifs IoT tels que les ampoules, prises et caméras connectées, ayant un nombre limité de fonctionnalités. Ces dispositifs établissent en général un nombre limité de connexions réseau et il est alors possible de caractériser leur comportement nominal en observant le trafic réseau d'un grand nombre de ces dispositifs. Les sentinelles se basent ainsi sur le comportement le plus couramment observé pour les dispositifs d'un même modèle afin de créer une spécification décrivant le comportement autorisé

4. Les *mining pools* sont des groupes de mineurs qui mettent en commun leurs ressources de calcul afin d'augmenter leurs chances de résoudre les puzzles cryptographiques de la *proof of work*. Ce mécanisme est courant pour les cryptomonnaies telles que le Bitcoin ou l'Ethereum car il permet aux mineurs d'obtenir une source de revenus constante.

pour ces dispositifs. Les systèmes de détection d'intrusion par spécifications émettent des alertes lorsqu'un comportement ne correspond pas aux spécifications. Le trafic réseau est alors autorisé ou bloqué, sans notion de confiance ou de probabilité d'attaque, contrairement aux systèmes de détection d'intrusion par anomalies. Les IDS par spécifications diffèrent aussi des IDS par signature pour lesquels les experts définissent une liste de signatures des différentes attaques connues. SERENIoT cherche à définir une liste des comportements légitimes afin de bloquer le trafic réseau inconnu.

Pour ce faire, les sentinelles se basent sur les listes blanches afin de répertorier les signatures des paquets réseau correspondants au comportement «normal» des dispositifs. Lorsqu'un paquet d'un dispositif IoT est intercepté par une sentinelle, cette dernière calcule la signature de ce paquet et vérifie si la signature apparaît dans la liste blanche associée au dispositif. Si la signature apparaît dans la liste blanche, le paquet est transmis. Sinon, le paquet est bloqué par la sentinelle qui reporte la signature au système (c.-à-d. la sentinelle ajoute la signature dans le bloc candidat pour la liste blanche correspondante). Cette signature sera éventuellement ajoutée à la liste blanche si elle est reportée par une majorité de sentinelles.

Ce mécanisme permet aux sentinelles de détecter et de bloquer les comportements anormaux observés seulement par une petite proportion de sentinelles.

3.6.1 Mise à jour des polices de sécurité

Lorsque le comportement d'un dispositif change, les autres sentinelles du réseau confirment ou non si elles ont aussi observé le changement. Le changement de comportement peut être le résultat d'une mise à jour ou d'une attaque. Pour décider si le nouveau comportement est légitime, les sentinelles se basent sur les observations de la majorité : si le changement a été observé par une majorité de sentinelles, il est considéré comme légitime et est ajouté à la liste blanche. Sinon il est considéré comme anormal et est bloqué. Cette logique repose sur l'idée que si une majorité de dispositifs d'un même modèle se comportent de manière similaire, ce comportement est le comportement «attendu» pour ces dispositifs. Le comportement «attendu» peut cependant être anormal. Dans ce cas, les dispositifs doivent être considérés comme non fiables et des actions doivent être prises contre le fabricant. Par exemple en janvier 2020, Google a révoqué l'accès du fabricant Xiaomi à l'écosystème Google Home Hub car des utilisateurs avaient accès aux flux vidéo des caméras d'autres utilisateurs [43].

3.6.2 Audits et transparence

La nature ouverte et décentralisée de SERENIoT introduit une nouvelle source de données pouvant être utile aux experts en sécurité pour auditer en temps réel l'évolution des comportements des dispositifs IoT à travers le monde. Ces données peuvent ainsi permettre de faire le suivi les menaces grandissantes, les taux d'adoption des mises à jour, etc. Par exemple, il est possible de surveiller la propagation d'un botnet en regardant les fork rejetés. Les implications en termes de vie privée sont discutées dans la section 5.

3.7 Connexion des dispositifs

Les dispositifs sont profilés lors de leur connexion aux sentinelles. Cette phase de profilage est nécessaire et permet aux sentinelles d'identifier les dispositifs afin de pouvoir mettre en vigueur le filtrage réseau. Le processus de connexion des dispositifs est détaillé ci-dessous :

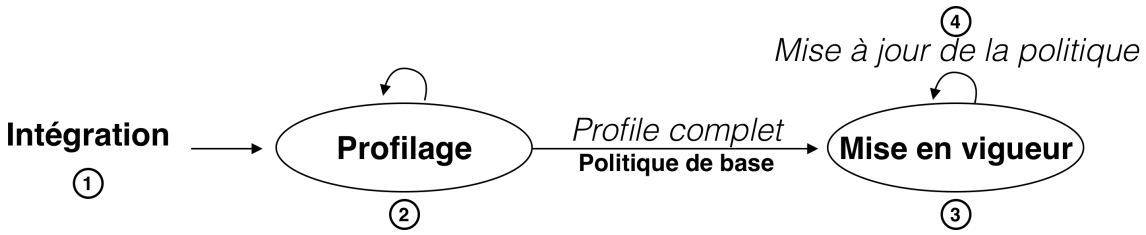


Figure 3.7 Flowchart de connexion des dispositifs

(1) **Intégration :** L'utilisateur achète un nouveau dispositif IoT et le connecte à la sentinelle via un réseau WiFi dédié diffusé par la sentinelle.

(2) **Profilage :** La sentinelle autorise l'intégralité du trafic réseau du dispositif et calcule les signatures des paquets réseaux échangés par le dispositif. Les signatures enregistrées durant la phase de profilage permettent de caractériser le dispositif et de l'assigner à la chaîne regroupant tous les dispositifs ayant une empreinte réseau similaire. Cette phase de profilage nécessite environ une minute pour la plupart des dispositifs testés durant notre évaluation (section 4.2). Si la blockchain correspondante n'existe pas (c.-à-d. aucun dispositif avec une empreinte réseau similaire n'est connecté à une sentinelle), une nouvelle blockchain spécifique pour ce type de dispositif est initialisée.

(3) **Mise en vigueur :** La sentinelle télécharge la blockchain à laquelle le dispositif a été assigné afin de construire la liste blanche correspondante. Le filtrage des paquets réseau est mis en vigueur et les paquets ne correspondant pas à la politique de sécurité sont bloqués. La sentinelle reporte aussi les nouvelles signatures à la liste blanche.

(4) **Changement de comportement / Mise à jour de la politique de sécurité :** Lorsque le comportement du dispositif évolue, les sentinelles votent afin de déterminer si ces changements doivent être incorporés dans la liste blanche en se basant sur les observations de la majorité. Dans le cas où une sentinelle contribue seule à une blockchain, elle peut décider des paquets à ajouter dans la liste blanche. Cela peut être problématique si la première sentinelle à contribuer à cette blockchain est malveillante et incorpore des paquets anormaux dans la liste blanche. Ceci est discuté dans la section 4.2.4.

3.8 Conclusion

Dans ce chapitre nous avons présenté SERENIoT, un système conçu pour répondre à la problématique de sécurisation des réseaux IoT domestiques en proposant une solution autonome et hautement compatible. SERENIoT ne nécessite en effet aucune intervention de la part de l'utilisateur pour fonctionner et est compatible avec les dispositifs IoT domestiques actuels et futurs quelle que soit leur marque. Les politiques de sécurités générées par SERENIoT mettent en vigueur un filtrage plus strict et plus adapté aux dispositifs IoT que les règles autorisant l'intégralité du trafic sortant, typiquement utilisée par les pare-feu domestiques pour sécuriser les ordinateurs personnels. Son architecture décentralisée basée sur une blockchain facilite son déploiement à grande échelle et lui permet de fonctionner sans nécessiter l'implication des fabricants ni de tiers de confiance. Bien que certains aspects comme l'utilisation de l'algorithme de *proof of work* soient expérimentaux et ne sont pas destinés à une utilisation commerciale, SERENIoT ouvre la voie à un nouveau type de systèmes de détection d'intrusion collaboratifs et introduits une nouvelle méthode de génération de politiques de sécurité basée sur la blockchain. SERENIoT construit les politiques de sécurité à appliquer pour sécuriser les dispositifs IoT en se basant sur la collaboration des sentinelles déployées au sein des différents réseaux domestiques. En agrégeant les observations d'une multitude de sentinelles SERENIoT détermine les comportements «normaux» des dispositifs et se base sur ces spécifications pour bloquer les actions anormales. Dans le chapitre 4, nous présentons les détails de l'implémentation du prototype de SERENIoT ainsi que les différentes expériences menées pour l'évaluer.

CHAPITRE 4 PROTOTYPE ET ÉVALUATION

Le chapitre 3 détaille les principes sur lesquels s'appuie SERENIoT et présente la stratégie mise en place pour générer et mettre en vigueur des politiques de sécurité pour les dispositifs IoT. Dans ce chapitre, nous allons expliciter l'implémentation du prototype de SERENIoT ainsi que les différentes expériences menées pour évaluer le système et valider nos hypothèses de recherche.

4.1 Détails de l'implémentation

Le prototype de SERENIoT se compose de trois logiciels distincts :

1. **Le logiciel des sentinelles** qui correspond à l'agent SERENIoT exécuté sur les sentinelles. Le logiciel des sentinelles implémente les principes décrits dans le chapitre 3 et gère l'interception et l'analyse des paquets issus des dispositifs IoT ainsi que la génération et la mise en vigueur des politiques de sécurité.
2. **Le simulateur** s'intègre au logiciel des sentinelles et permet de simuler des dispositifs IoT en se basant sur des captures réseau de dispositifs réels. Le simulateur rend possible la simulation d'un réseau SERENIoT avec un grand nombre de sentinelles et de dispositifs IoT.
3. **L'interface web** permet de visualiser en temps réel l'état du système et présente des graphiques permettant d'évaluer son fonctionnement. Les trois logiciels qui composent le prototype ont intégralement été développés par nos soins afin d'évaluer notre solution.

La figure 4.1 présente un schéma de l'architecture logicielle globale du prototype et de ses dépendances. Le prototype s'appuie sur le paquet Debian RaspAP [44] pour établir le point d'accès WiFi sur lequel les dispositifs IoT sont connectés. Pour intercepter les paquets issus des dispositifs, le logiciel des sentinelles se base sur *NetFilter*. *NetFilter* est un framework mis à disposition par le noyau Linux permettant d'implémenter un pare-feu. Lorsqu'un paquet est reçu par le système d'exploitation, ce dernier est placé dans une queue grâce à une règle iptables [1] spécifique. Les sentinelles communiquent entre elles via WebRTC [2] et Websockets [3]. WebRTC et Websockets sont couramment utilisés dans les applications web temps réels et

1. Logiciel permettant de spécifier des règles de filtrage du trafic au niveau du système d'exploitation [45].
2. WebRTC est un framework web permettant des communications pair à pair en temps réel [46].
3. Websockets est un protocole réseau de couche 7 permettant une communication en temps réel full duplex par-dessus une connexion TCP [47].

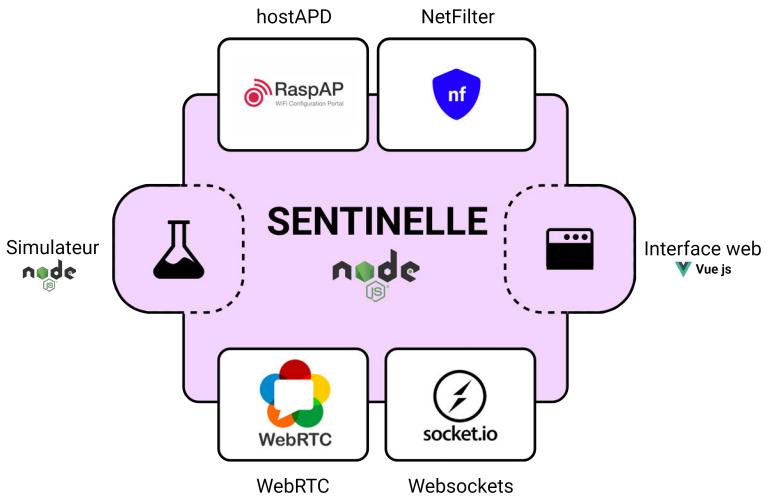


Figure 4.1 L'architecture logicielle du prototype

sont dotés de fonctionnalités de *NAT traversal* particulièrement pratiques dans le contexte des réseaux privés domestiques. Socket.io est une librairie node.js permettant d'utiliser facilement les Websockets. Socket.io est aussi utilisée pour permettre à l'interface web de communiquer avec le logiciel des sentinelles.

4.1.1 sentinelles

Nous avons développé un prototype fonctionnel de sentinelle afin d'évaluer notre système et de valider son fonctionnement. Le prototype de sentinelle est développé en Node.js, un environnement d'exécution Javascript *cross-plateform* et *opensource* conçu pour exécuter des applications web asynchrones. L'écosystème Node.js dispose d'une large communauté et de nombreux paquets pouvant être utilisés lors du développement. Il permet ainsi d'implémenter rapidement un prototype sans avoir à ré-implémenter l'intégralité de la pile logicielle. De plus, le caractère asynchrone de Node.js facilite le développement de logiciels temps réels.

Nous avons conçu le logiciel des sentinelles de manière modulaire afin que chaque module soit indépendant et puisse être mis à jour sans nécessiter de modifications des autres composants. Cette architecture modulaire permet de prioriser le développement des modules clefs et facilite les tests en permettant de tester chaque module indépendamment. L'architecture logicielle des sentinelles est basée sur l'architecture des sentinelles décrite dans la section 3.3. Chaque composant des sentinelles est implémenté par un des modules logiciels suivants :

— *firewall.js* est le module qui gère la mise en vigueur des politiques de sécurité. Il se base sur la librairie *NetFilter* pour intercepter les paquets réseaux issus des dispositifs

IoT.

— *fingerprint.js* est le module permettant de calculer les signatures de paquets.
— *blockchain.js* gère les blockchains et est chargé de maintenir à jour les registres décentralisés. La logique des blockchains a été implémentée de zéro. *block.js* et *consensus.js* décrivent respectivement la logique de création et de validation des blocs.
— *networking.js* est le module permettant la communication entre les sentinelles.
— *identity.js* génère l'adresse unique de la sentinelle au démarrage du logiciel. L'algorithme utilisé pour générer l'adresse de la sentinelle est identique à celui utilisé pour calculer les adresses Bitcoin.
— *settings.js* regroupe les paramètres du logiciel comme par exemple le port d'écoute des communications entre sentinelles et le chemin d'accès du stockage des blocs des blockchains.
— *ui.js* est le module permettant la communication avec l'interface web.
— *index.js* est le point d'entrée du logiciel des sentinelles. Il est chargé d'initialiser les différents modules pour démarrer le logiciel.

Le fonctionnement des modules clefs chargés du pare-feu, de la communication entre les sentinelles et de la gestion du registre décentralisé est détaillé ci-dessous :

Module pare-feu (*firewall.js*)

Ce module est en charge de la mise en vigueur des politiques de sécurité. Concrètement, ce module établit le lien entre la librairie *NetFilter* qui gère la capture des paquets au niveau du système d'exploitation et le logiciel de la sentinelle. Lorsqu'un paquet est intercepté par *NetFilter*, le pare-feu calcule sa signature grâce au module *fingerprint.js* puis consulte le registre décentralisé par l'intermédiaire du module *blockchain.js*. Le pare-feu transmet ensuite le verdict à *NetFilter* qui transmet ou bloque le paquet. Afin de pouvoir créer les règles nécessaires dans *iptable* et de pouvoir interagir avec *NetFilter*, le logiciel des sentinelles requiert les droits super-utilisateurs (*root*).

Module réseau (*networking.js*)

Les sentinelles sont connectées via un réseau pair à pair. Le module *networking.js* gère les connexions WebRTC et la transmission de données entre les sentinelles. Il met en place des mécanismes afin d'optimiser le temps de transmissions des blocs ainsi que la charge du réseau. Un mécanisme permet par exemple d'empêcher les retransmissions inutiles afin d'éviter les *tempêtes de broadcast* tout en s'assurant que toutes les sentinelles du réseau reçoivent les dernières mises à jour des blockchains. Le module *networking.js* gère aussi la découverte des pairs.

Cette fonctionnalité permet de déterminer d'une part les sentinelles à contacter initialement pour constituer le voisinage d'une sentinelle et d'autre part de détecter lorsque qu'un nouveau dispositif IoT est connecté. Le voisinage des sentinelles est en effet en constante évolution et s'adapte en fonction des dispositifs connectés aux sentinelles. Les sentinelles cherchent ainsi à établir des liens avec les autres sentinelles protégeant des dispositifs similaires afin d'optimiser la transmission des blocs pour les blockchains spécifiques aux dispositifs. Pour se connecter à un nouveau voisin, les sentinelles établissent dans un premier temps une connexion Websocket afin de négocier les paramètres de la connexion WebRTC.

Module blockchain (*blockchain.js, block.js & consensus.js*)

Le module blockchain maintient le registre décentralisé dans lequel sont stockées les politiques de sécurité propres aux dispositifs. Il s'assure que les données sont synchronisées entre les sentinelles et que seules les signatures de paquets reportées par la majorité des sentinelles sont inscrites dans le registre. Les sentinelles possèdent une instance de blockchain par modèle de dispositif protégé. Les blocs des blockchains sont stockés dans des fichiers JavaScript Object Notation (JSON) dans le répertoire «.blocks». Pour le besoin des simulations nous avons implémenté deux algorithmes de consensus : *proof of work* et *proof of work simulé*. L'algorithme de *proof of work simulé* délai aléatoirement la production de blocs des sentinelles sans consommer de puissance de calcul afin d'imiter le délai introduit par l'algorithme de *proof of work*.

4.1.2 Simulateur

Le simulateur s'intègre au logiciel des sentinelles et permet de simuler des dispositifs IoT lors des expériences à grandes échelles. Les dispositifs IoT sont simulés à partir de captures réseau réalisées avec des dispositifs réels. Pour ce faire, les paquets générant des signatures uniques sont extraits des captures réseau pour chaque dispositif. Ces paquets sont enregistrés dans des fichiers «profiles» caractérisant les empreintes réseau des dispositifs. Le simulateur charge alors le profile des différents dispositifs à simuler et rejoue les paquets dans l'ordre, à des intervalles aléatoires. L'utilisation d'intervalles aléatoires permet de reproduire l'aspect imprévisible du comportement des dispositifs IoT. Les interactions de l'utilisateur sont en effet imprévisibles et peuvent intervenir à tout moment. Par exemple, dans le cas d'une ampoule connectée, il est possible que l'utilisateur coupe l'alimentation électrique du dispositif à partir d'un interrupteur, ce qui a pour effet d'interrompre momentanément toutes les communications pour ce dispositif. Le simulateur permet aussi de simuler des attaques sur les dispositifs en incorporant des paquets anormaux dans les empreintes réseau des dispositifs.

Le simulateur s'intègre directement dans le logiciel des sentinelles et transmet les paquets IP au module du pare-feu. Lorsqu'il est utilisé, le simulateur remplace donc la librairie *NetFilter* qui est habituellement chargée de l'interception des paquets. Dans le cas de notre simulation, le verdict donné par le pare-feu pour les paquets n'est pas pris en compte par le simulateur.

4.1.3 Interface web

Nous avons développé une interface web afin d'avoir une représentation graphique de l'état du système. L'interface a la forme d'un tableau de bord et permet de suivre en temps réel l'évolution des blockchains, du réseau de sentinelles et des politiques de sécurité. L'interface permet ainsi de s'assurer du bon déroulement des simulations ainsi que de détecter de potentiels dysfonctionnements et d'évaluer le succès ou l'échec des expériences. Cette interface est donc principalement destinée aux chercheurs travaillant sur le projet SERENIoT mais peut aussi être utilisée lors de démonstrations pour illustrer le fonctionnement de SERENIoT. L'interface web a été développée avec le framework vue js [48]. Certains graphiques tels que les graps réseau et le graph de voisinage ont été réalisés avec D3.js durant le cours d'INF8808. L'interface web est composée de trois vues distinctes :

La vue réseau (figure 4.2) donne des informations sur le réseau de sentinelles. Le graphique réseau est le composant central de cette vue. Il représente les différentes sentinelles ainsi que leurs interconnexions. Il permet de visualiser l'état du réseau et d'identifier de potentiels groupements isolés de sentinelles. Les sentinelles sont représentées par des points, et leurs connexions par des traits entre les points. Au survol d'une sentinelle, un menu contextuel avec des informations sur la sentinelle est affiché. Les différentes connexions de la sentinelle survolée sont aussi mises en relief. Chaque sentinelle est identifiée visuellement par une *identicon* unique calculée à partir de son adresse. Les *identicons* permettent de retrouver et d'identifier facilement les sentinelles dans les différents graphiques. La colonne de gauche de la vue affiche un récapitulatif de l'état du réseau ainsi que la liste des sentinelles connectées. La colonne de droite affiche un diagramme à bandes des blockchains les plus actives en termes de pourcentage de sentinelles participantes (100% signifie que toutes les sentinelles du réseau participent à cette blockchain) ainsi que la liste des blockchains actives.

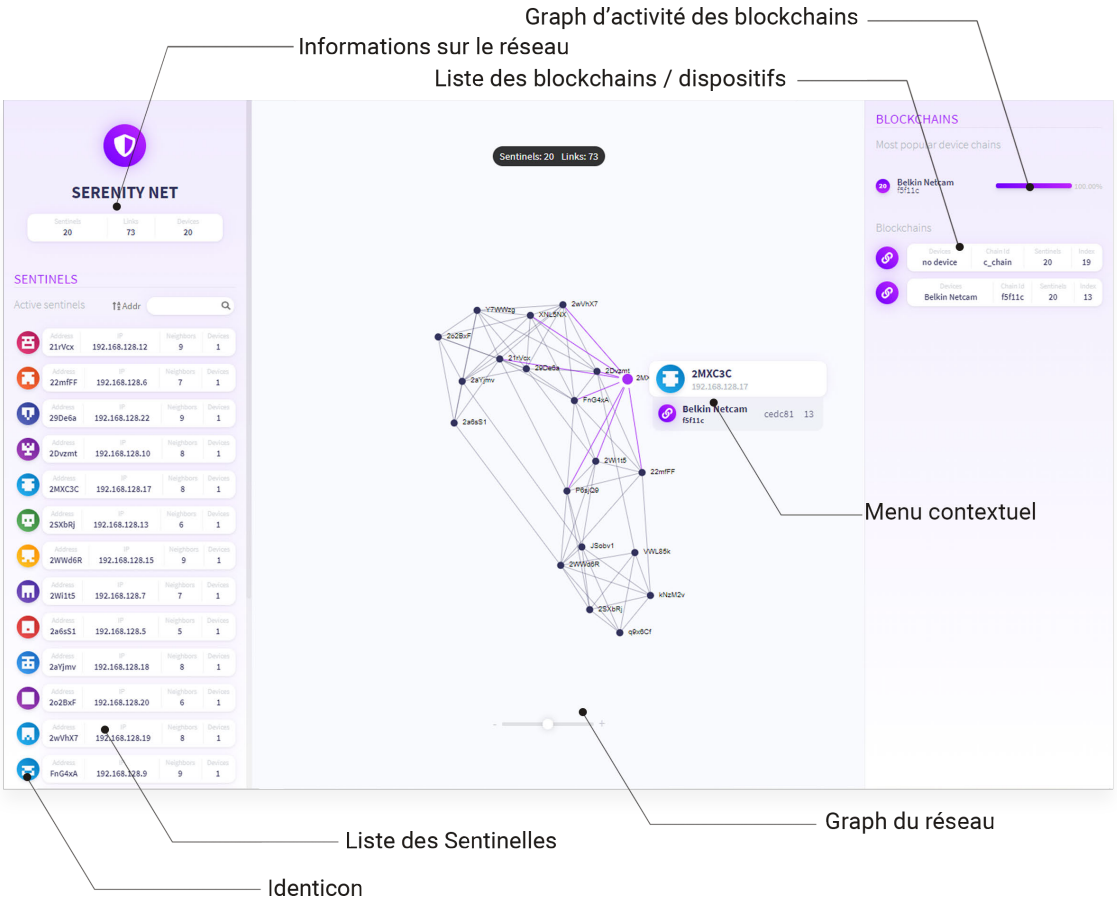
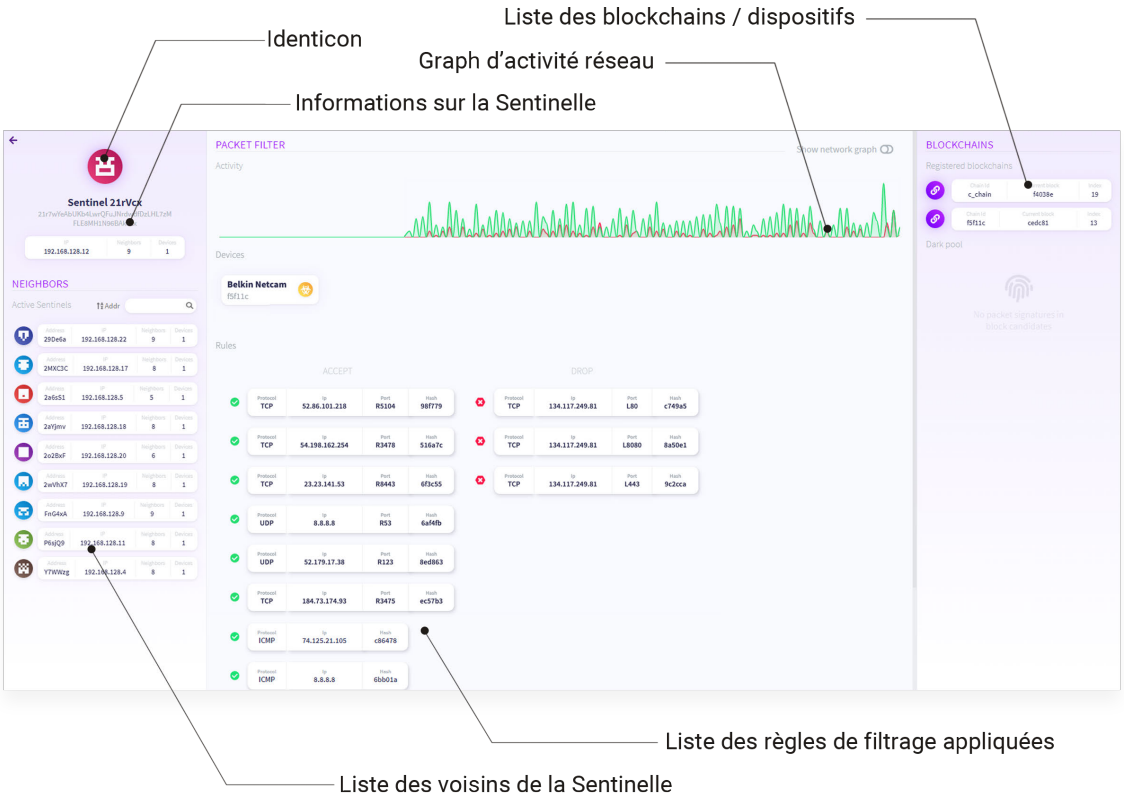


Figure 4.2 Vue réseau

La vue sentinelle (figure 4.3) permet d'obtenir des informations sur l'état du pare-feu, les dispositifs IoT connectés et le voisinage d'une sentinelle. La vue sentinelle est composée de deux onglets. Le premier permet à l'utilisateur de visualiser en temps réel l'état du pare-feu ainsi que le trafic réseau autorisé et bloqué. Il est composé d'une *sparkline* représentant le nombre de paquets autorisés et bloqués en temps réels, suivi de la liste des dispositifs et de leurs états (Autorisé, en quarantaine ou bloqué). On retrouve ensuite la liste des règles de filtrage réseau appliquées. Cet onglet permet d'identifier les paquets bloqués et les potentiels dispositifs compromis/ne fonctionnant pas correctement. En cliquant sur *show network chart* dans le panneau central en haut à droite, l'utilisateur peut afficher l'onglet du graphique de voisinage. Le graphique de voisinage représente le voisinage réseau de sentinelle. La sentinelle est représentée au centre par son *identicon*. Le premier cercle de connexions (en violet) représente les différents dispositifs IoT connectés à la sentinelle. Chaque dispositif est représenté par un point. Le second cercle (en violet estompé) représente les voisins de la sentinelle ainsi que leurs interconnexions. Chaque voisin est représenté par un point. Enfin le dernier cercle

représente les voisins des voisins de la sentinelle. Ce graphique représente le réseau du point de vue d'une sentinelle ce qui permet d'évaluer facilement le nombre de sauts nécessaires pour joindre les différentes sentinelles ainsi que l'intégration de la sentinelle au réseau. Il permet aussi de visualiser les dispositifs qui lui sont connectés. La colonne de gauche de la vue sentinelle affiche des informations sur la sentinelle (*Identicon*, Adresse, Adresse IP), suivi de la liste de ses voisins. La colonne de droite affiche la liste des blockchains auxquelles participe la sentinelle (une blockchain correspond à un modèle de dispositif IoT).

Enfin, **la vue blockchain** (figure 4.4) donne des informations sur les différentes blockchains. Le composant principal de la vue blockchain est un graphique de la chaîne de blocs. Il permet de visualiser l'évolution de la blockchain en temps réel, les différents blocs ainsi que la répartition du consensus. L'utilisateur peut ainsi observer la blockchain grandir et vérifier que la plus longue chaîne regroupe la majorité des sentinelles. Les signatures de paquets contenues dans les blocs sont listées à droite des blocs. En cliquant sur *show network chart* dans le panneau central en haut à droite, l'utilisateur peut afficher le graphique réseau de la blockchain. Le graphique réseau de la blockchain représente les différentes sentinelles participant à la blockchain ainsi que leurs différentes interconnexions. Chaque sentinelle est représentée par un ovale dans lequel est inscrit l'index courant de la sentinelle. La visualisation des index des différentes sentinelles permet de visualiser si la blockchain est correctement synchronisée entre toutes les sentinelles participantes. Les interconnexions entre les sentinelles sont représentées par des traits ce qui permet d'identifier rapidement si une sentinelle est isolée et désynchronisée. La colonne de gauche de la vue blockchain affiche des informations sur la blockchain, suivi d'un diagramme à bandes représentant la répartition des blocs minés parmi les sentinelles. On retrouve ensuite la liste des sentinelles participants à cette blockchain. La colonne de gauche affiche une *sparkline* représentant l'activité de la blockchain (en termes de blocs minés par minute), suivi de la liste des signatures inscrites dans la blockchain et des signatures présentes dans les blocs candidats.



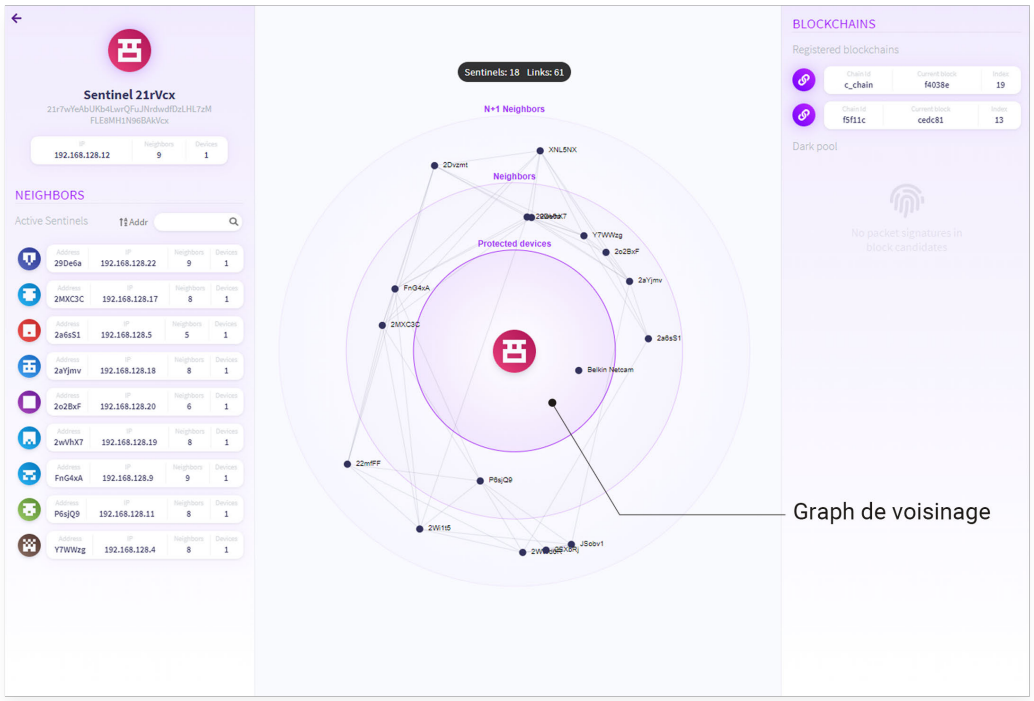


Figure 4.3 Vue sentinelle

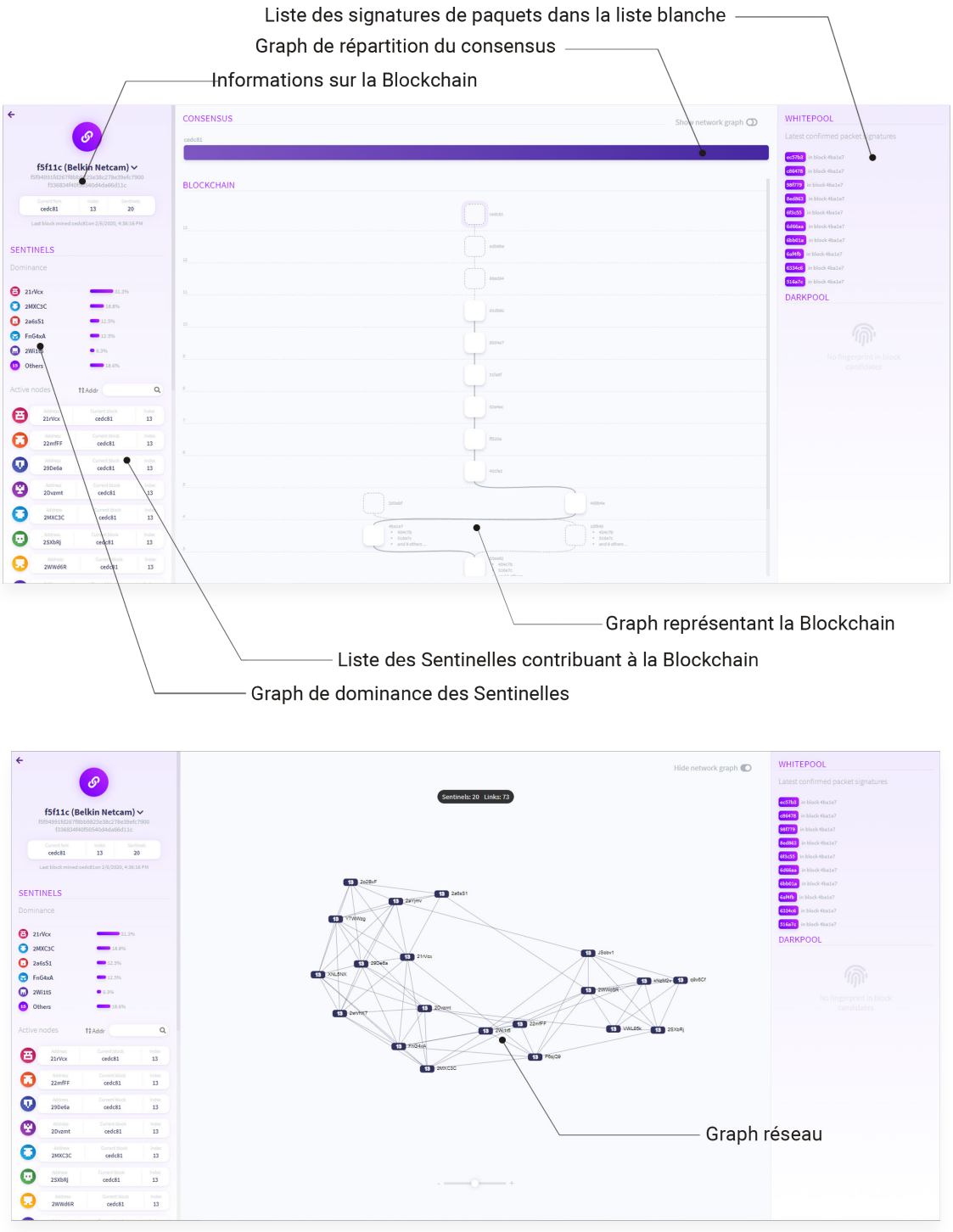


Figure 4.4 Vue blockchain

4.1.4 Limites du prototype

Nous avons observé des instabilités au niveau du module réseau pour certaines topologies. Ces instabilités sont dues à un bug dans l'implémentation de la gestion du *multichain* et de la demande de blocs. Les sentinelles affectées par ce bug sont dans l'incapacité de récupérer des blocs auprès de leurs voisins. Elles sont alors désynchronisées et ne sont plus en mesure de maintenir à jour leurs blockchains. Lors de nos expériences, ce bug a affecté une minorité de sentinelles mais n'a pas empêché la majorité du réseau de fonctionner correctement.

La simplicité du simulateur peut aussi interroger. Cependant, notre système se base sur l'empreinte réseau des dispositifs pour construire les politiques de sécurité. L'ordre et le délai réel entre les paquets ne rentrent donc très peu en compte lors de la génération des politiques de sécurité des dispositifs.

Enfin, le prototype a été conçu afin d'évaluer le système SERENIoT et son concept en laboratoire et n'est de ce fait pas adapté pour un déploiement commercial à grande échelle.

4.2 Évaluation de SERENIoT

Nous avons mis en place deux bancs de tests afin d'évaluer SERENIoT. Le premier est constitué de dispositifs IoT réels et permet de réaliser des expériences à petite échelle afin d'évaluer le fonctionnement du système avec des dispositifs réels. Le second banc de test est entièrement simulé et permet de réaliser des tests à plus grande échelle avec des sentinelles et des dispositifs simulés afin de tester la scalabilité et la robustesse du système.

4.2.1 Test sur sentinelles et dispositifs réels

Le banc de test avec des dispositifs réels a tenté de reproduire la topologie des réseaux domestiques. La photo 4.5 présente le banc de test utilisé pour cette expérience. Comme illustré par la figure 4.6, les sentinelles ont été déployées sur des Raspberry PI 3b+ et placées derrière le routeur d'accès à internet. Elles étaient ainsi en mesure d'intercepter le trafic entre les dispositifs IoT et internet. Chaque sentinelle représentait un réseau domestique différent. Cependant, pour des raisons pratiques nous n'utilisions qu'un seul routeur pour ce banc de test (bien qu'en réalité, chaque sentinelle serait connectée à un son propre routeur). Les dispositifs IoT utilisés pour ce banc de test étaient des *LIFX Smart Bulb mini*. Une ampoule *LIFX Smart Bulb mini* a été connectée à chaque sentinelle. Nous avons choisi ce modèle de dispositifs pour leur prix bon marché et pour leur popularité. Les dispositifs LIFX sont effet largement répandus et correspondent aux types de dispositifs que l'on retrouve couramment

au sein des réseaux domestiques [1].

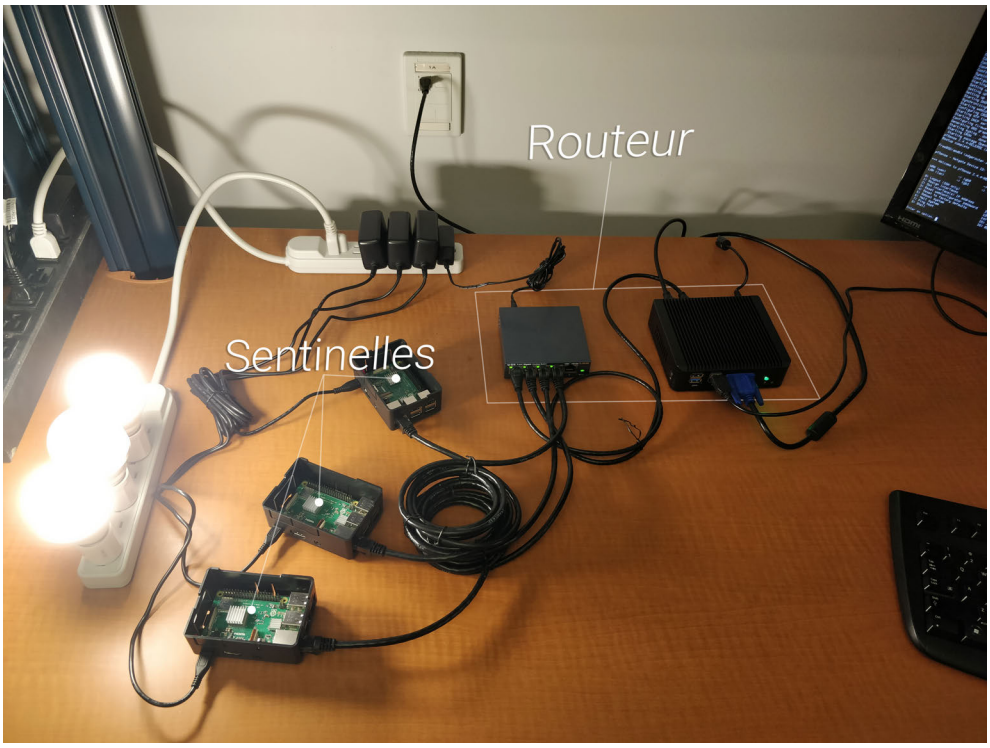


Figure 4.5 Photo du banc de test

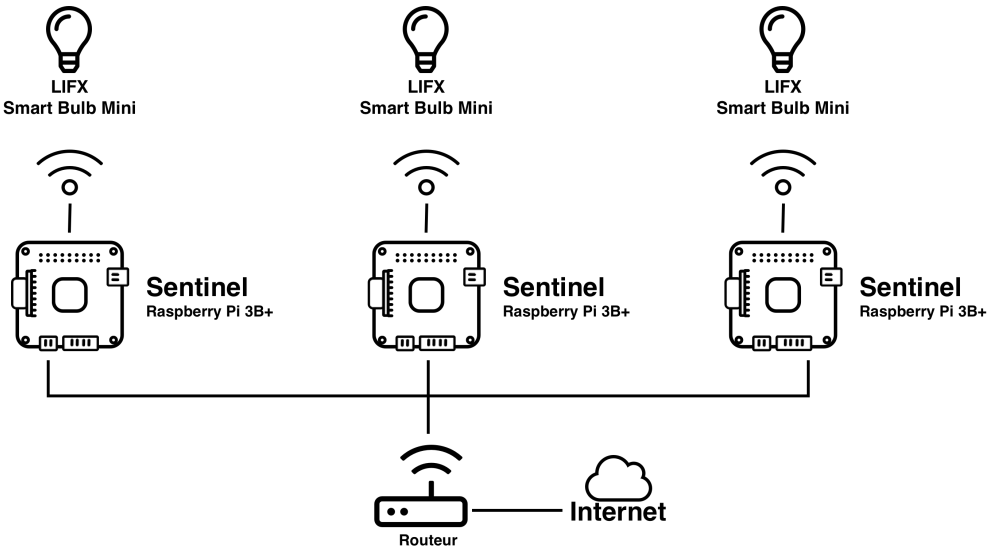


Figure 4.6 Topologie du banc de test utilisé lors de l'expérience avec des dispositifs réels

L'expérience en conditions réelles a été réalisée avec les moyens à notre disposition nous permettant seulement de mettre en oeuvre 3 sentinelles et 3 dispositifs. Malgré ces limitations, il nous a semblé judicieux de tester notre système en conditions réelles afin de vérifier la capacité des sentinelles à converger sur une politique de sécurité pour des dispositifs réels dont le comportement varie selon des variables qui ne peuvent pas être prisent en compte lors de simulations.

Méthodologie

1. Initialisation des sentinelles sans aucune connaissance préalable des dispositifs et avec des politiques de sécurité vides.
2. Mise en interaction des dispositifs avec l'application mobile du fabricant afin de générer du trafic réseau «légitime»
3. Test des politiques de sécurité en introduisant un dispositif perturbateur dans le réseau d'une sentinelle afin de vérifier que seul le trafic «légitime» est autorisé

Objectif de l'expérience

Valider le fonctionnement du système par test sur un réseau IoT réel.

Critères de succès

— Convergence des sentinelles pour construire et mettre en vigueur une politique de sécurité partagée pour les ampoules *LIFX Smart Bulb mini*.
— Lors d'interactions de l'utilisateur avec les dispositifs via l'application, seul le trafic correspondant aux dispositifs IoT est transmis.

Résultats

Le tableau 4.1 liste les signatures de paquets enregistrées pour les ampoules *LIFX Smart Bulb mini*. Cette liste de signatures correspond à la politique de sécurité appliquée par les sentinelles lors de l'expérience. On constate que seul le trafic NTP vers les serveurs de Google et le trafic TCP vers l'API de LIFX était autorisé.

Tableau 4.1 Signatures de paquets enregistrées pour les ampoules *LIFX Smart Bulb mini*

Payload du paquet IP	**Signature du paquet**	**Description**
UDP time1.google.com R123	0cca40...aed4d4	NTP
TCP 104.198.46.246 R56700	4e2b3d...2a4474	LIFX API

Le diagramme de convergence (figure 4.7) représente l'évolution du nombre de signatures bloquées dans la politique de sécurité au cours de l'expérience. La courbe rouge représente le nombre moyen de signatures bloquées par les sentinelles (si trois sentinelles bloquent trois signatures, le nombre moyen de signatures bloquées est un). La courbe verte représente le nombre moyen de signatures dans la politique de sécurité. La courbe noire représente le nombre moyen de paquets interceptés par les sentinelles.

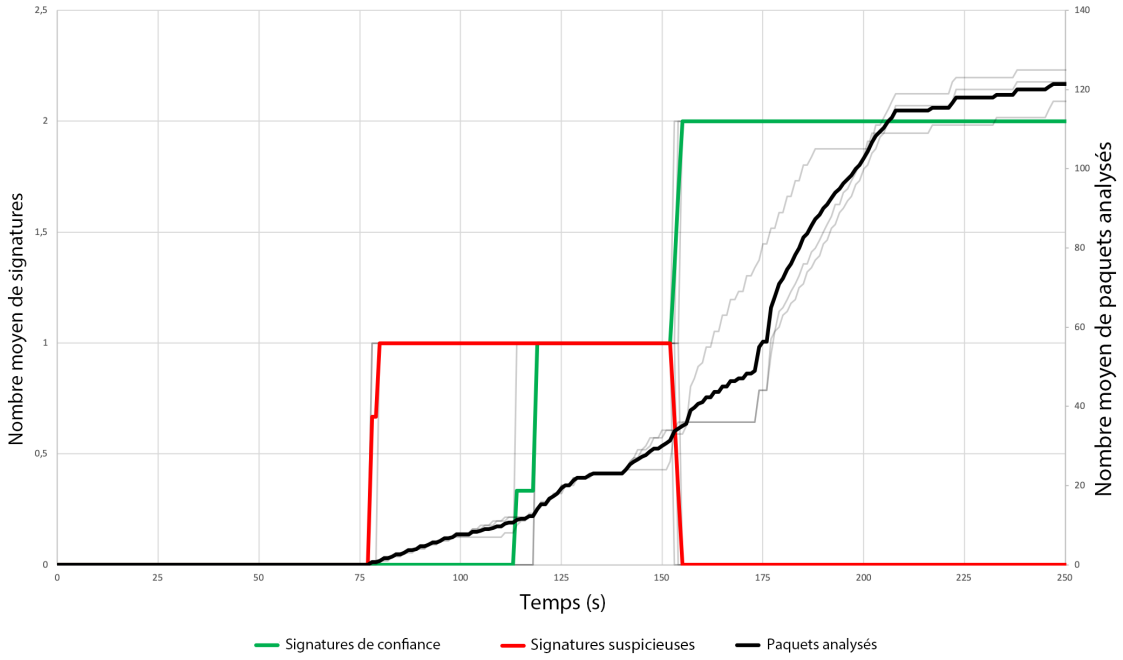


Figure 4.7 Diagramme de convergence

On constate sur la figure 4.7 que les trois ampoules présentaient des comportements globalement analogues, mais pas strictement identiques. On constate aussi que les sentinelles ont convergé sur une politique de sécurité en moins de deux minutes. Dès lors, tout le trafic autre que le trafic NTP vers les serveurs de Google et le trafic TCP vers l'API de LIFX était bloqué.

Conclusion

Comme indiqué dans le tableau 4.2, cette expérience valide par test sur un réseau IoT réel le fonctionnement du système selon les deux critères fixés. Il est important de noter que le concept est validé mais qu'il sera à consolider par un test réel avec une plus grande variété de dispositifs.

Tableau 4.2 Résultats de l'expérience sur des sentinelles et des dispositifs réels

Valider le fonctionnement du système par test sur un réseau IoT réel	
Critère	Résultat
Convergence des sentinelles pour construire et mettre en vigueur une politique de sécurité partagée pour les ampoules LIFX Smart Bulb mini	**Oui**
Lors d'interactions de l'utilisateur avec les dispositifs via l'application, seul le trafic correspondant aux dispositifs IoT est transmis.	**Oui**

4.2.2 Tests sur sentinelles et dispositifs simulés

Le banc de test avec des dispositifs simulés a permis d'évaluer le système à plus grande échelle, avec différents modèles de dispositifs et un plus grand nombre de sentinelles. Le banc de test simulé supportait ainsi jusqu'à 1000 sentinelles. Chaque sentinelle était exécutée dans un conteneur Docker séparé et nous utilisions Docker Swarm pour orchestrer le cluster et déployer les conteneurs des sentinelles. La simulation était hébergé sur Amaon AWS et nous utilisions 10 instances Amazon EC2 c5.2.xlarge, hébergeant chacune 100 sentinelles et connectées via un réseau virtuel Docker. La figure 4.8 représente l'architecture du banc de test simulé.

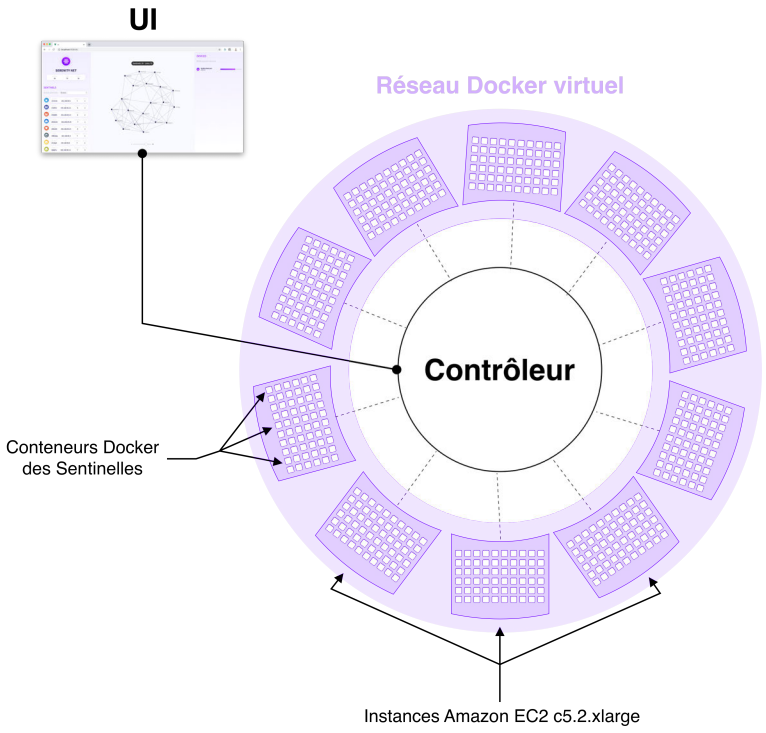


Figure 4.8 Architecture de la simulation sur Amazon AWS

Nous avons utilisé un contrôleur pour centraliser les flux de données remontant des sentinelles. Ces données étaient alors mises en forme et affichées dans l'interface web qui permettait de suivre l'état du système et le déroulement de la simulation en temps réel. L'interface web était aussi utilisée pour évaluer la réussite ou l'échec de l'expérience.

Méthodologie

L'expérience est répétée sur plusieurs périodes de temps (de plusieurs heures à plusieurs jours) et avec différents nombres de sentinelles (de 20 à 1000) afin de tester le système dans différentes configurations. Les sentinelles traitent à l'aide du module de simulation un ensemble de dispositifs issus de la base de données de Alrawi et al. [2]. Cette base de données fournit des captures réseau d'un réseau IoT domestique avec 53 dispositifs différents sur une période de 9 jours. Pour chaque itération :

- Initialisation des sentinelles sans aucune connaissance préalable des dispositifs et avec des politiques de sécurité vides.
- Test des politiques de sécurité en introduisant du trafic malveillant dans le comportement de certains dispositifs afin de vérifier que seul le trafic listé dans les politiques de sécurité est autorisé. Le trafic malveillant utilisé provient de données d'attaques issues de la base de données de Hyunjae et al. [49].

Objectifs de l'expérience

- Valider la compatibilité du système avec différents dispositifs IoT par simulation.
- Valider l'extensibilité du système par simulation.

Critères de succès

- Convergence des sentinelles pour construire et mettre en vigueur des politiques de sécurité partagées pour les dispositifs protégés.
- Le trafic malveillant injecté dans certains dispositifs est bloqué.

Analyse des données

Nous avons réalisé dans un premier temps une analyse des données comportementales des dispositifs simulés. La fonction de répartition (figure 4.9) représente l'évolution du nombre de signatures de paquets uniques enregistrées pour les dispositifs de la base de données de Alrawi et al. [2] sur une période de 9 jours. Le diagramme à bande (figure 4.10) représente

le nombre de signatures de paquets uniques enregistrées pour les dispositifs de la base de données de Alrawi et al. [2] sur une période de 9 jours.

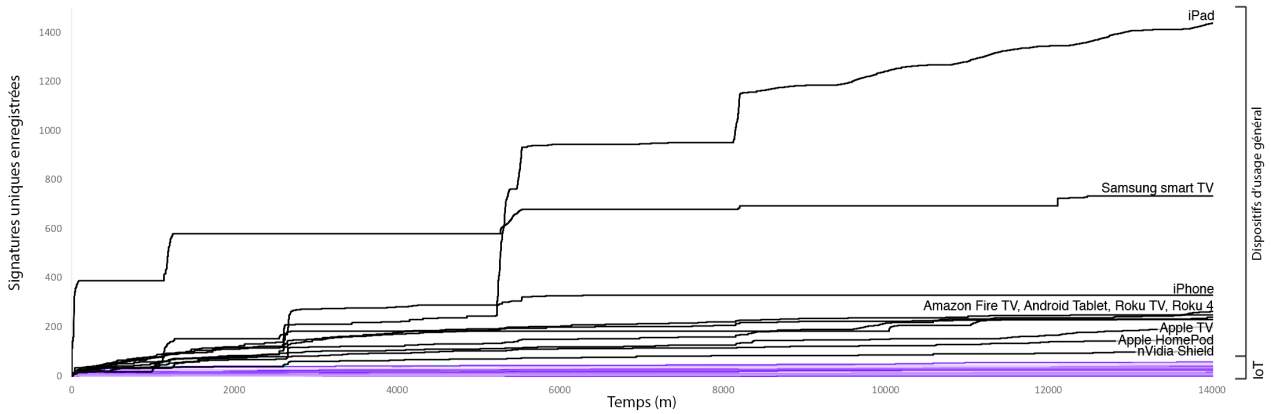


Figure 4.9 Fonction de répartition des signatures de paquets issues des dispositifs

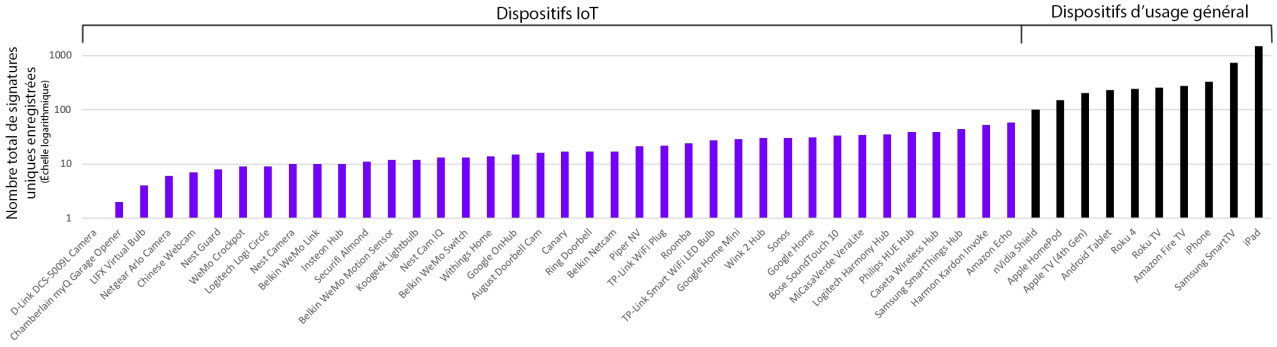


Figure 4.10 Diagramme à bande du nombre de signatures enregistrées par dispositif

Les figures 4.9 et 4.10 font apparaître clairement deux classes de dispositifs IoT : les dispositifs ayant un comportement simple caractérisé par un petit nombre de signatures de paquets réseau telles que la LIFX Smart Bulb, le TPLink WiFi Plug ou le Nest Guard et les dispositifs plus complexes tels que les iPads, les smart TVs, etc. Nous observons aussi sur la figure 4.9 que les dispositifs avec peu de fonctionnalités sont caractérisés par un comportement stable dans le temps, évoluant peu.

Résultats

Nous avons observé lors de l'expérience le fonctionnement correct de la génération et la mise en vigueur des politiques de sécurité pour les dispositifs supportés par SERENIoT. De plus, lors de l'injection de trafic malveillant, les sentinelles ont identifié et rejeté les signatures anormales grâce au consensus fourni par la blockchain. Les paquets malveillants ont été bloqués sans impacter le trafic «légitime» des dispositifs. La figure 4.11 montre l'identification et la réjection d'un *fork* avec des signatures de paquet malveillantes. Nous avons aussi observé que le point de rupture intervenait généralement lorsqu'environ 50% des sentinelles observaient une même signature de paquet (le point de rupture correspond au pourcentage de sentinelles reportant une même Signature nécessaire pour que cette signature soit incorporée dans la politique de sécurité).

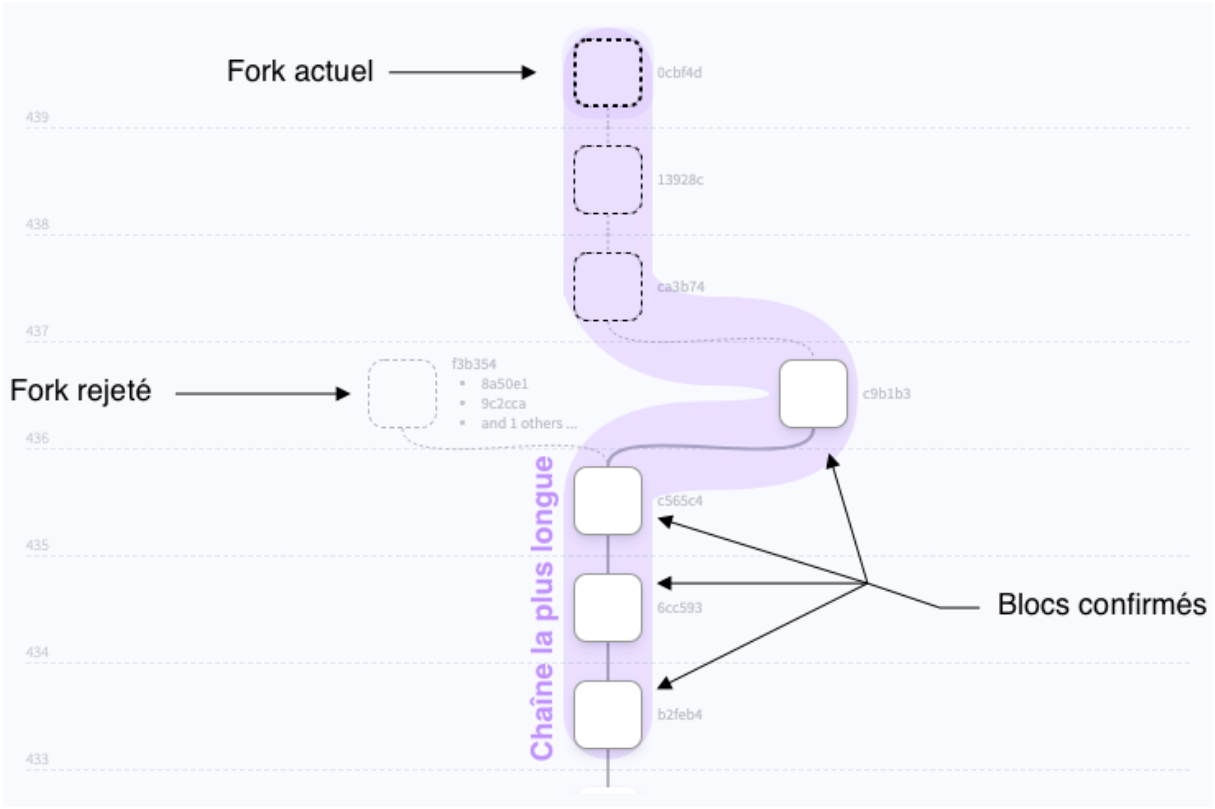


Figure 4.11 Réjection d'un *fork* contenant des signatures anormales

Conclusion

Comme indiqué dans les tableaux 4.3 et 4.4, les tests valident la compatibilité et l'extensibilité du système avec différents modèles de dispositifs IoT selon les critères fixés. Il est important

de noter que le concept est validé mais qu'il sera à consolider par un test réel avec une plus grande variété de dispositifs.

4.2.3 Évaluation des performances de la blockchain

Nous avons mesuré l'évolution de la taille des blockchains et relevé les *runtime metrics* des containers Docker exécutant les sentinelles afin d'évaluer la capacité de notre système à fonctionner sur le long terme.

Taille des blockchains

Une expérience de 24 heures nous a permis de mesurer la taille des blocs stockés par les sentinelles après 1 heure, 5 heures et 24 heures. Pour cela, nous avons utilisé 20 sentinelles simulées ayant chacune une caméra *Belkin Netcam* connectée. Le nombre de sentinelles dans le réseau n'influence pas la taille des blockchains, car la fréquence de production des blocs est fixée par l'algorithme de consensus de la chaîne de contrôle. Les sentinelles suppriment les blocs des *forks* rejetés dès qu'elles convergent sur la chaîne la plus longue.

Le tableau 4.5 montre que la taille moyenne des blocs de la chaîne de contrôle a tendance à rester constante au cours du temps. La taille des blocs de la chaîne de contrôle est en effet déterminée par le nombre de modèles de dispositifs différents protégés par les sentinelles. Chaque modèle de dispositif possède sa chaîne spécifique et chaque chaîne de dispositif est indexée dans la chaîne de contrôle. Les blocs de la chaîne de contrôle listent les empreintes numériques des derniers blocs produits pour les chaînes de dispositifs et contiennent ainsi au maximum «le nombre de chaînes de dispositifs» empreintes numériques de blocs. Les empreintes numériques de blocs sont des hachés SHA-256 et ont une taille fixe de 32 octets. Il est de ce fait facile d'estimer la taille de la chaîne de contrôle pour un nombre donné de dispositifs IoT. En se basant sur nos relevés, la chaîne de contrôle avec 1 dispositif IoT devrait

Tableau 4.3 Résultats de l'expérience sur des sentinelles et des dispositifs simulés selon le premier critère de réussite

Valider la compatibilité du système avec différents dispositifs IoT par simulation	
Critère	Résultat
Convergence des sentinelles pour construire et mettre en vigueur des politiques de sécurité partagées pour les dispositifs protégés	**Oui**
Le trafic malveillant injecté dans certains dispositifs est bloqué.	**Oui**

Tableau 4.4 Résultats de l'expérience sur des sentinelles et des dispositifs simulés selon le second critère de réussite

Valider l'extensibilité du système par simulation.	
Critère	Résultat
Convergence des sentinelles pour construire et mettre en vigueur des politiques de sécurité partagées pour les dispositifs protégés	**Oui**
Le trafic malveillant injecté dans certains dispositifs est bloqué.	**Oui**

Tableau 4.5 Taille de la chaîne de contrôle pour 1 dispositif IoT

Temps écoulé	**Nombre de blocs**	**Taille**	**Taille moyenne des blocs**
1 heure	205	64Ko	312 octets
5 heures	918	291Ko	316 octets
24 heures	4384	1.4Mo	316 octets

occuper 511Mo après 1 an d'exécution.

$$1.4Mo * 365jours = 511Mo/an$$

Si nous considérons 10 000 modèles de dispositifs IoT différents protégés par les sentinelles, la chaîne de contrôle devrait occuper 511Go après 1 an d'exécution.

$$4384blocs * 10000 * 32o + 1.4Mo = 1.4Go/jour$$

$$1.4Go * 365jours = 511Go/an$$

Tableau 4.6 Taille d'une chaîne de dispositif

Temps écoulé	**Nombre de blocs**	**Taille**	**Taille moyenne des blocs**
1 heure	193	59Ko	304 octets
5 heures	906	268Ko	296 octets
24 heures	4367	1.3Mo	295 octets

Le tableau 4.6 montre que la taille moyenne des blocs des chaînes des dispositifs tend aussi à être constante. En effet, la majorité des blocs dans les chaînes des dispositifs sont vides, car aucune nouvelle signature de paquet n'a été observée. L'enregistrement de nouvelles signatures est rare et la majorité des blocs seront donc vides sur le long terme.En se basant sur nos relevés, les chaînes de dispositifs devraient donc occuper 474Mo après une année

d'exécution.

$$1.3Mo * 365jours = 474Mo/an$$

Les sentinelles doivent conserver une copie de la chaîne de contrôle. Elles doivent cependant télécharger uniquement les chaînes correspondant aux dispositifs qu'elles protègent. Ainsi une sentinelle protégeant 10 dispositifs IoT devrait maintenir une copie de la chaîne de contrôle ainsi que 10 chaînes spécifiques aux dispositifs. Une version améliorée de SERENIoT pourrait avoir une fonctionnalité d'expiration des blocs afin de supprimer automatiquement les anciens blocs. Cette fonctionnalité empêcherait la blockchain de grandir indéfiniment tout en permettant aux anciennes politiques de sécurité d'être mises à jour en supprimant les signatures de paquets n'étant plus utilisées par la majorité des dispositifs.

Runtime metrics des sentinelles

Nous avons utilisé la commande *docker stats* afin d'enregistrer les *Runtime metrics* [50] des sentinelles. Nous avons enregistré les *Runtime metrics* de 20 sentinelles au cours d'une expérience de 24 heures après 1 heure, 5 heures et 24 heures. Les les *Runtime metrics* détaillent l'utilisation de processeur, de la mémoire et de la bande passante. Nous observons sur la figure 4.12 que les sentinelles utilisent environ 140 MiB [4] de RAM après 24h d'exécution et que l'usage de la bande passant est corrélé avec l'évolution de la taille des blockchains.Les sentinelles ne téléchargent les blocs que pour la chaîne de contrôle et pour les chaînes correspondant aux dispositifs qu'elles protègent. L'utilisation du réseau entre deux sentinelles est donc susceptible de varier en fonction du nombre de dispositifs différents qu'elles protègent. Enfin, les sentinelles ont simulé l'algorithme de consensus *proof of work* lors de l'expérience. Aussi, les mesures d'utilisation du processeur ne sont pas représentatives d'une utilisation réelle de la *proof of work*. Dans le cas d'une utilisation réelle de la *proof of work*, nous observerions une charge CPU maximisé à 100% pour toutes les sentinelles.

4. Mebi est le préfix binaire IEC pour 2^{20} (1 MiB = 1,048,576 bytes). Mega est le péfix décimal du système international pour 10^6 (1 MB = 1,000,000 bytes)

Après 1 heure

CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK I/O	PIDS
116ddc8315e3	sereniot_sentinel_17	8.26%	121.1MiB / 62.9GiB	0.19%	8.54MB / 8.3MB	4.32MB / 1.54MB	72
3ee2c7e0b4c0	sereniot_sentinel_11	6.03%	117.9MiB / 62.9GiB	0.18%	5.06MB / 4.93MB	4.62MB / 1.54MB	66
2dca5d88f443	sereniot_sentinel_9	5.43%	117.7MiB / 62.9GiB	0.18%	5.12MB / 4.95MB	4.68MB / 1.54MB	64
34b49ee42043	sereniot_sentinel_10	7.60%	119.4MiB / 62.9GiB	0.19%	7.68MB / 7.44MB	4.46MB / 1.57MB	72
d71f74aa8472	sereniot_sentinel_1	4.33%	117.2MiB / 62.9GiB	0.18%	3.39MB / 3.29MB	4.4MB / 1.54MB	60
80644372246c	sereniot_sentinel_3	5.03%	120.1MiB / 62.9GiB	0.19%	6.03MB / 5.86MB	4.1MB / 1.58MB	66
79d94ff6e382	sereniot_sentinel_8	4.54%	119.4MiB / 62.9GiB	0.19%	5.94MB / 5.75MB	4.27MB / 1.54MB	66
e1931d8e5dbe	sereniot_sentinel_20	5.92%	118.2MiB / 62.9GiB	0.18%	5.89MB / 5.69MB	4.33MB / 1.54MB	66
06c65698a8ff	sereniot_sentinel_6	8.23%	131.6MiB / 62.9GiB	0.20%	6.04MB / 5.86MB	5.69MB / 1.58MB	72
90871ef9a878	sereniot_sentinel_4	3.73%	119.3MiB / 62.9GiB	0.19%	5.17MB / 4.99MB	4.21MB / 1.57MB	64
80e66a223e37	sereniot_sentinel_14	7.54%	118.5MiB / 62.9GiB	0.18%	7.63MB / 7.4MB	4.39MB / 1.57MB	70
05802a55ef96	sereniot_sentinel_15	4.73%	118.7MiB / 62.9GiB	0.18%	5.89MB / 5.7MB	4.56MB / 1.54MB	66
39533976f69c	sereniot_sentinel_7	6.77%	118.9MiB / 62.9GiB	0.18%	5.05MB / 4.89MB	4.23MB / 1.54MB	66
5ac0f1396855	sereniot_sentinel_18	7.57%	119MiB / 62.9GiB	0.18%	7.72MB / 7.52MB	4.35MB / 1.58MB	70
14f1d55676ea	sereniot_sentinel_12	7.00%	120.4MiB / 62.9GiB	0.19%	6.76MB / 6.56MB	4.28MB / 1.54MB	68
5f8a06de25e8	sereniot_sentinel_5	6.68%	117.2MiB / 62.9GiB	0.18%	5.89MB / 5.7MB	4.69MB / 1.54MB	66
0da0631aca94	sereniot_auto_config_service_1	0.00%	74.62MiB / 62.9GiB	0.12%	18.3kB / 13.7kB	881kB / 98.3kB	42
45b573c7b844	sereniot_sentinel_13	8.05%	120.7MiB / 62.9GiB	0.19%	8.51MB / 8.25MB	4.52MB / 1.54MB	72
b04de7370511	sereniot_sentinel_19	5.41%	116.8MiB / 62.9GiB	0.18%	5.98MB / 5.78MB	4.35MB / 1.54MB	66
1341ba5687b7	sereniot_sentinel_2	6.18%	119.8MiB / 62.9GiB	0.19%	5.87MB / 5.68MB	4.04MB / 1.54MB	66
8e3d3ab4a327	sereniot_sentinel_16	5.02%	118.8MiB / 62.9GiB	0.18%	5.95MB / 5.76MB	4.09MB / 1.54MB	66

Après 5 heures

CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK I/O	PIDS
116ddc8315e3	sereniot_sentinel_17	10.81%	125.6MiB / 62.9GiB	0.19%	37.2MB / 36.1MB	4.32MB / 1.54MB	72
3ee2c7e0b4c0	sereniot_sentinel_11	7.10%	125.9MiB / 62.9GiB	0.20%	22.4MB / 21.8MB	4.62MB / 1.54MB	66
2dca5d88f443	sereniot_sentinel_9	6.40%	124.2MiB / 62.9GiB	0.19%	22.5MB / 21.8MB	4.68MB / 1.54MB	64
34b49ee42043	sereniot_sentinel_10	9.50%	127.7MiB / 62.9GiB	0.20%	33.5MB / 32.5MB	4.46MB / 1.57MB	72
d71f74aa8472	sereniot_sentinel_1	4.37%	118.1MiB / 62.9GiB	0.18%	15MB / 14.7MB	4.4MB / 1.54MB	60
80644372246c	sereniot_sentinel_3	7.65%	124.6MiB / 62.9GiB	0.19%	26.2MB / 25.5MB	4.1MB / 1.58MB	66
79d94ff6e382	sereniot_sentinel_8	7.68%	127.7MiB / 62.9GiB	0.20%	26.1MB / 25.3MB	4.27MB / 1.56MB	66
e1931d8e5dbe	sereniot_sentinel_20	7.40%	125MiB / 62.9GiB	0.19%	26.1MB / 25.3MB	4.33MB / 1.54MB	66
06c65698a8ff	sereniot_sentinel_6	8.15%	138.5MiB / 62.9GiB	0.22%	26.2MB / 25.5MB	5.69MB / 1.58MB	72
90871ef9a878	sereniot_sentinel_4	6.47%	124.9MiB / 62.9GiB	0.19%	22.5MB / 21.8MB	4.21MB / 1.57MB	64
80e66a223e37	sereniot_sentinel_14	9.87%	128MiB / 62.9GiB	0.20%	33.4MB / 32.5MB	4.39MB / 1.57MB	70
05802a55ef96	sereniot_sentinel_15	7.83%	123.9MiB / 62.9GiB	0.19%	26.1MB / 25.3MB	4.56MB / 1.54MB	66
39533976f69c	sereniot_sentinel_7	7.27%	126.1MiB / 62.9GiB	0.20%	22.4MB / 21.8MB	4.23MB / 1.54MB	66
5ac0f1396855	sereniot_sentinel_18	9.22%	125.1MiB / 62.9GiB	0.19%	33.5MB / 32.7MB	4.35MB / 1.58MB	70
14f1d55676ea	sereniot_sentinel_12	8.27%	129.4MiB / 62.9GiB	0.20%	29.7MB / 28.9MB	4.28MB / 1.54MB	68
5f8a06de25e8	sereniot_sentinel_5	7.27%	125.4MiB / 62.9GiB	0.19%	26.1MB / 25.3MB	4.69MB / 1.54MB	66
0da0631aca94	sereniot_auto_config_service_1	0.00%	74.62MiB / 62.9GiB	0.12%	18.8kB / 13.7kB	881kB / 98.3kB	42
45b573c7b844	sereniot_sentinel_13	10.70%	124.5MiB / 62.9GiB	0.19%	37.1MB / 36.1MB	4.52MB / 1.54MB	72
b04de7370511	sereniot_sentinel_19	7.79%	124.1MiB / 62.9GiB	0.19%	26.2MB / 25.4MB	4.35MB / 1.54MB	66
1341ba5687b7	sereniot_sentinel_2	7.28%	129.4MiB / 62.9GiB	0.20%	26MB / 25.3MB	4.04MB / 1.54MB	66
8e3d3ab4a327	sereniot_sentinel_16	7.71%	125.3MiB / 62.9GiB	0.19%	26.1MB / 25.3MB	4.09MB / 1.54MB	66

Après 24 heures

CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK I/O	PIDS
116ddc8315e3	sereniot_sentinel_17	10.97%	142MiB / 62.9GiB	0.22%	179MB / 174MB	4.49MB / 1.54MB	72
3ee2c7e0b4c0	sereniot_sentinel_11	7.23%	141.1MiB / 62.9GiB	0.22%	108MB / 106MB	4.88MB / 1.58MB	66
2dca5d88f443	sereniot_sentinel_9	6.64%	140.5MiB / 62.9GiB	0.22%	109MB / 105MB	4.69MB / 1.54MB	64
34b49ee42043	sereniot_sentinel_10	10.29%	143.6MiB / 62.9GiB	0.22%	161MB / 157MB	4.48MB / 1.59MB	72
d71f74aa8472	sereniot_sentinel_1	5.32%	133.4MiB / 62.9GiB	0.21%	72.8MB / 71.2MB	4.61MB / 1.6MB	60
80644372246c	sereniot_sentinel_3	7.65%	141.4MiB / 62.9GiB	0.22%	126MB / 123MB	4.2MB / 1.58MB	66
79d94ff6e382	sereniot_sentinel_8	7.26%	140.8MiB / 62.9GiB	0.22%	126MB / 122MB	4.37MB / 1.6MB	66
e1931d8e5dbe	sereniot_sentinel_20	7.23%	140.9MiB / 62.9GiB	0.22%	126MB / 122MB	4.57MB / 1.6MB	66
06c65698a8ff	sereniot_sentinel_6	20.96%	157.8MiB / 62.9GiB	0.24%	126MB / 123MB	6.25MB / 1.58MB	77
90871ef9a878	sereniot_sentinel_4	6.69%	140MiB / 62.9GiB	0.22%	109MB / 105MB	4.23MB / 1.57MB	64
80e66a223e37	sereniot_sentinel_14	9.24%	132.5MiB / 62.9GiB	0.21%	161MB / 157MB	4.73MB / 1.61MB	70
05802a55ef96	sereniot_sentinel_15	7.83%	133.1MiB / 62.9GiB	0.21%	126MB / 123MB	4.98MB / 1.6MB	66
39533976f69c	sereniot_sentinel_7	7.55%	138.9MiB / 62.9GiB	0.22%	108MB / 105MB	4.26MB / 1.6MB	66
5ac0f1396855	sereniot_sentinel_18	9.27%	139.6MiB / 62.9GiB	0.22%	161MB / 157MB	4.68MB / 1.59MB	70
14f1d55676ea	sereniot_sentinel_12	8.70%	142.2MiB / 62.9GiB	0.22%	144MB / 140MB	4.69MB / 1.6MB	68
5f8a06de25e8	sereniot_sentinel_5	7.59%	140.4MiB / 62.9GiB	0.22%	126MB / 123MB	4.84MB / 1.58MB	66
0da0631aca94	sereniot_auto_config_service_1	0.00%	74.62MiB / 62.9GiB	0.12%	21.5kB / 13.7kB	881kB / 98.3kB	42
45b573c7b844	sereniot_sentinel_13	9.92%	142MiB / 62.9GiB	0.22%	179MB / 174MB	4.6MB / 1.58MB	72
b04de7370511	sereniot_sentinel_19	7.61%	139.1MiB / 62.9GiB	0.22%	126MB / 123MB	4.67MB / 1.54MB	66
1341ba5687b7	sereniot_sentinel_2	7.98%	143.7MiB / 62.9GiB	0.22%	126MB / 122MB	4.11MB / 1.6MB	66
8e3d3ab4a327	sereniot_sentinel_16	7.72%	141.8MiB / 62.9GiB	0.22%	126MB / 122MB	4.11MB / 1.54MB	66

Figure 4.12 *Runtime metrics* des sentinelles

4.2.4 Évaluation de sécurité

Lorsqu'un dispositif IoT est connecté à une sentinelle, la sentinelle détermine la politique de sécurité à souscrire en se basant sur le comportement du dispositif. Ainsi les dispositifs ayant un comportement similaire seront groupés sur la même politique de sécurité et les dispositifs déjà compromis lors de leur connexion seront assignés à une politique de sécurité différente. Grâce à ce mécanisme, tous les dispositifs assignés à une même politique de sécurité se comportent initialement identiquement. Les dispositifs non compromis sont alors groupés ensemble et la politique de sécurité correspondante ne contiendra que les signatures de paquets légitimes reflétant le comportement de ces dispositifs. Dans cette section, nous évaluons les différentes attaques pouvant mener à l'incorporation de signatures de paquets malveillantes dans la politique de sécurité ou à l'exploitation des dispositifs afin de modifier leur comportement.

Attaque sur les dispositifs durant la phase de profilage

Durant la phase de profilage, les sentinelles autorisent l'intégralité du trafic et ne mettent en vigueur aucun filtrage des paquets pour les dispositifs récemment connectés. Bien que cette phase ne dure généralement que quelques minutes, un attaquant pourrait en profiter pour attaquer le dispositif. Dans ce cas, l'attaque modifierait l'empreinte réseau du dispositif qui serait donc assigné à une chaîne différente des autres dispositifs «sains» du même modèle. Cette chaîne regrouperait alors tous les dispositifs ayant généré la même empreinte réseau durant la phase de profilage (c.-à-d. tous les dispositifs du même modèle ayant été la cible de la même attaque durant la phase de profilage) et les sentinelles ne filtreraient pas les paquets anormaux de ces dispositifs.

Les dispositifs pourraient aussi avoir été infectés avant leur connexion aux sentinelles. Dans ce cas, si le comportement des dispositifs infectés était similaire à celui des dispositifs «sains» du même modèle, tous les dispositifs seraient assignés à la même chaîne et les actions malveillantes seraient filtrées. Cependant, si le comportement des dispositifs infecté était différent, ils seraient assignés à une chaîne différente comme dans le cas des attaques durant la phase de profilage.

Attaque sur les dispositifs

Deux types d'attaques contre les dispositifs peuvent être identifiées lorsque les sentinelles sont en phase de mise en vigueur de la politique de sécurité (voir section 3.7) : Les attaques provenant d'internet et celles provenant du réseau local.

— Les attaques provenant du réseau local ont de fortes chances d'aboutir. Les sentinelles n'effectuent en effet aucun filtrage au niveau du réseau local. Cependant les sentinelles bloqueront tout comportement vers internet déviant des spécifications. Cela protège contre des dispositifs infectés sur le réseau local essayant d'attaquer des cibles sur internet en les restreignant au strict comportement listé dans les spécifications. De plus, grâce à la nature ouverte et décentralisée de sa blockchain, SERENIoT met à disposition des experts de nouvelles métriques permettant de monitorer en temps réel l'évolution des comportements d'un grand nombre de dispositifs IoT. Grâce à cette fonctionnalité, SERENIoT aide aussi les experts à prendre des actions pour bloquer les menaces à mesure qu'elles grandissent.

— Les attaques provenant d'un hôte distant et ciblant des dispositifs particuliers sont bloquées par les sentinelles tant qu'une minorité des sentinelles observent le même modèle d'attaque. Ainsi ces attaques deviennent plus difficiles à réaliser à mesure que le nombre de sentinelles protégeant un même modèle de dispositif IoT augmente. La réussite d'une telle attaque est conditionnée à ce qu'une majorité de sentinelles observent une même empreinte réseau (c.-à-d. que les sentinelles enregistre des signatures de paquets avec les mêmes adresses IP). Pour réaliser une telle attaque doit alors cibler 51% des dispositifs du modèle ciblé en moins d'un intervalle de bloc (le délai entre deux blocs). Pour les modèles de dispositifs populaires, il est peu probable qu'un attaquant dispose des ressources nécessaires pour cibler simultanément un grand nombre de dispositifs depuis un même hôte. Les acteurs malveillants utilisent généralement des botnets pour mener à bien des attaques de grande envergure contre les dispositifs IoT. Les botnets IoT tels que Mirai [6], Brickerbot [51] et Hajime [52] partagent des empreintes réseau similaires durant la phase d'infection : Ils effectuent des balayages de port pour trouver des ports telnet, ssh ou http ouvert puis essayent de bruteforcer les identifiants. Cependant la majorité de ces botnets utilisent des hôtes différents pour lancer des attaques sur de nouveaux dispositifs. Ils utilisent en effet les dispositifs infectés comme relais pour en infecter de nouveaux. Les signatures des paquets résultant de ces attaques de botnet varieront donc d'une cible à l'autre, car l'adresse IP et le port utilisé par l'hôte attaquant ne sont pas consistants pour toutes les cibles. SERENIoT est efficace par design contre ce type d'attaque du fait que ces signatures varient d'une sentinelle à l'autre et ont donc peu de chances d'être ajoutées dans les listes blanches des dispositifs ciblés.

Attaque sur les sentinelles

Les sentinelles jouent les rôles de noeuds blockchain et de points de filtrage du trafic réseau. Ainsi, si une sentinelle est compromise, un attaquant peut être en mesure d'insérer ou de supprimer des règles des listes blanches locales [5]. Cependant, l'attaque d'un noeud spécifique de la blockchain ne permet pas l'inclusion de signatures de paquet malveillantes dans la blockchain, les signatures étant validées et confirmées par la majorité des sentinelles avant d'être ajoutées dans une chaîne. Les chaînes de SERENIoT sont vulnérables à deux attaques :

— Attaques sur la majorité de la chaîne de contrôle. Ces attaques peuvent aboutir si un attaquant possède une puissance de calcul supérieure à celle de plus de la moitié des sentinelles. Un attaquant est alors en mesure de résoudre les problèmes de *proof of work* plus rapidement que les sentinelles et est alors en mesure d'ajouter des signatures dans les chaînes/politiques de sécurité. Ce type d'attaque est dévastateur pour le réseau, car les listes blanches sont partagées entre toutes les sentinelles. Nous discutons un protocole consensus alternatif permettant de diminuer les chances de réussite d'une telle attaque dans la section 5.

— Attaques sur la majorité des chaînes spécifiques aux dispositifs. Ces attaques peuvent aboutir si un attaquant contrôle plus de la moitié des sentinelles inscrites à une chaîne. Un attaquant est alors en mesure d'ajouter des blocs à ces chaînes plus souvent que le reste des sentinelles ce qui peut conduire à l'ajout de blocs malveillants dans la plus longue chaîne. Cette attaque devient plus difficile à réaliser lorsque le nombre de sentinelles protégeant un même modèle de dispositif augmente. Pour les dispositifs IoT les plus populaires avec un grand nombre de sentinelles contribuant à leurs chaînes, la difficulté de cette attaque est similaire à une attaque sur la majorité de la chaîne de contrôle.

4.3 Conclusion

Afin de pouvoir apprécier et mesurer les performances et l'efficience de SERENIoT, nous avons procédé à quatre types d'évaluations :

— **Fonctionnelle** à partir de dispositifs réels et simulé afin de valider la méthode de génération de politiques de sécurité proposée par SERENIoT. Nous avons alors observé que les sentinelles ont été en mesure de générer des politiques de sécurité et de les mettre en vigueur pour bloquer les paquets anormaux.

— **De compatibilité** avec différents modèles de dispositifs IoT simulés. SERENIoT

5. Ce principe s'applique aussi aux pare-feu et routeurs qui sont généralement mieux protégés que les autres hôtes internes au réseau.

semble ainsi être adapté aux dispositifs IoT présents au sein des maisons connectées.

— **De performance** des sentinelles afin d'observer l'espace de stockage nécessaire pour faire fonctionner SERENIoT sur le long terme.

— **De sécurité** en regard au *threat model* exposé dans la section 3.2.

Ces simulations, expériences et mises en condition ont permis d'obtenir des résultats probants qui valident globalement le concept et les choix fonctionnels et structurels de SERENIoT mais qui mettent aussi en évidence des limites et améliorations à prévoir. L'analyse et l'interprétation des résultats sont discutées dans le chapitre suivant.

CHAPITRE 5 DISCUSSION

Au cours de cette recherche, nous avons étudié la possibilité de déployer une solution autonome permettant la génération et la mise en vigueur de politiques de sécurités spécifiques pour réseaux IoT domestiques. Nous avons exploré dans le chapitre 3 la piste d'un système de détection collaboratif basé sur la blockchain exploitant l'hypothèse que la majorité des dispositifs IoT présents au sein des maisons connectées ont une empreinte réseau facilement caractérisable et évoluant peu au cours du temps. Aussi, la solution que nous avons conçue génère des politiques de sécurités en se basant sur des observations du trafic réseau des dispositifs au sein de différentes installations domestiques. Nous considérons que les comportements réseau les plus observés correspondent aux comportements «normaux» des dispositifs et sont donc autorisés. La blockchain est utilisée pour agréger toutes les observations en politiques de sécurité et pour faciliter la synchronisation et la distribution des politiques de sécurité. Nous avons évalué cette méthode dans le chapitre 4 à l'aide du prototype que nous avons développé au cours de deux expériences réelles et simulées visant à valider nos hypothèses et le fonctionnement du système. Ce chapitre présente les observations et conclusions tirées lors de la conception et de l'évaluation de SERENIoT.

5.1 Implications des résultats expérimentaux

5.1.1 Analyse du comportement des dispositifs IoT

Le bon fonctionnement de la stratégie de génération de politique de sécurité de SERENIoT repose sur l'hypothèse que les dispositifs IoT grand publique ont une empreinte réseau :

- **Simple**, composée d'une dizaine de connexions réseau. Une empreinte réseau simple facilite la génération d'une politique de sécurité en observant le trafic, car le dispositif ne communique qu'avec un nombre réduit d'hôtes sur internet.
- **Majoritairement constante**, évoluant peu au cours du temps. Une empreinte réseau majoritairement constante facilite la mise en vigueur d'une politique de sécurité qui nécessite moins régulièrement d'être mise à jour.

Afin de vérifier cette hypothèse, nous avons analysé dans la section 4.2.2 le comportement de 53 dispositifs IoT issus de la base de données de Alrawi et al. [2].

Empreinte réseau simple

Le nombre de signatures uniques par dispositif enregistré lors de l'analyse de la base de données est représenté sur la figure 4.10. On remarque que deux classes de dispositifs se dénotent clairement :

- Les dispositifs ayant un comportement simple caractérisé par une dizaine de signatures de paquets réseau telles que la LIFX Smart Bulb, le TPLink WiFi Plug ou le Nest Guard. Les empreintes réseau de ces dispositifs sont caractérisées par leurs fonctionnalités. Ces fonctionnalités sont identiques pour tous les dispositifs d'un même modèle.
- Les dispositifs plus complexes tels que les iPads, les smart TVs, etc. Contrairement aux dispositifs mentionnés précédemment, l'empreinte réseau de ces dispositifs dépend majoritairement de l'utilisateur et des apps qu'il installe. Par exemple l'empreinte réseau d'un iPad dépend des applications installées mais aussi des sites web que l'utilisateur consulte. Tous les iPads peuvent donc potentiellement avoir des empreintes réseaux différentes.

Ainsi, les résultats de l'analyse indiquent que la majorité des dispositifs IoT présents dans la base de données ont une empreinte réseau relativement simple constituée d'une dizaine de signatures. On peut cependant s'interroger sur la complétude de la base de données. Les captures ne couvrent en effet qu'une période de neuf jours, relativement courte à l'échelle de la vie des dispositifs IoT. Il se pourrait ainsi que ces dispositifs n'aient au cours de ces neuf jours pas exposé l'intégralité de leur comportement ou que leur comportement vienne à changer ultérieurement. Bien que cela soit vrai pour les dispositifs plus complexes tels que les ordinateurs personnels dont l'empreinte réseau varie grandement en fonction de leur utilisation (et peut donc difficilement être caractérisée sur une période si courte), les dispositifs IoT en question ne disposent que d'un nombre très réduit de fonctionnalités. Leur empreinte réseau étant la manifestation réseau de ces fonctionnalités, elle est donc peu susceptible d'évoluer ultérieurement sans mise à jour (contrairement à un ordinateur personnel où l'utilisation de différents logiciels et donc de différentes fonctionnalités change son empreinte réseau). Ce résultat valide donc que la majorité des dispositifs IoT semblent avoir empreinte réseau simple.

Empreinte réseau majoritairement constante

La fonction répartition de la figure 4.9 représente l'évolution du nombre de signatures uniques par dispositifs sur la période de neuf jours couverte par la base de données. Comme précédemment, nous constatons que les dispositifs peuvent être répartis en deux catégories :

— Les dispositifs dont le comportement reste relativement constant.
— Les dispositifs plus complexes tels que les iPads, les smart TVs, etc. dont le nombre de signatures uniques enregistrées ne cesse de croître.

L'analyse de la figure 4.9 indique donc que la majorité des dispositifs de la base de données ont une empreinte réseau relativement constante. On constate cependant que les courbes de certains dispositifs IoT augmentent aux alentours de 8500 minutes (6 jours). Ces augmentations sont potentiellement dues à des actions de l'utilisateur ou à l'établissement de connexions vers des serveurs dont les adresses IP n'ont pas pu être associées à des noms de domaine par *reverse DNS* lors de l'analyse des captures réseaux de la base de données. Ces évolutions restent tout de même relativement rares comparé à celles des courbes des dispositifs plus complexes tels que l'iPad. Cette observation valide donc notre hypothèse de départ et suggère que l'empreinte réseau initiale des dispositifs IoT évolue peu.

5.1.2 Génération des politiques de sécurité

Les politiques de sécurité de SERENIoT sont générées à partir des signatures observées par les sentinelles. Les signatures reportées par la majorité des sentinelles sont ainsi considérées «légitimes» et ajoutées aux politiques de sécurité. Le bon fonctionnement de ce procédé est conditionné par la capacité de la majorité sentinelles à converger sur une liste de signatures. Aussi, nous avons cherché à vérifier que les sentinelles étaient en mesure de converger sur une liste de signatures pour les dispositifs IoT présents au sein de notre banc de test. Pour ce faire, nous avons réalisé dans la section 4.2 deux expériences avec des dispositifs réels et simulés. Nous avons alors observé que :

— Les sentinelles ont convergé sur une liste de signatures pour les dispositifs ayant un comportement simple et majoritairement constant.
— Les politiques de sécurité résultantes se sont montrées efficaces pour bloquer le trafic parasite injecté.

Ces résultats valident qu'il est possible de générer des politiques de sécurité à partir des signatures reportées par la majorité des sentinelles. Cependant, cette méthode fonctionne uniquement pour les dispositifs ayant une empreinte réseau simple et relativement constante. Les dispositifs avec une empreinte réseau complexe et fortement variable ont en effet moins de chance d'exposer des comportements concordants et les sentinelles risquent ainsi de diverger d'avantage sur les signatures reportées.

5.2 Mise en contexte et considérations

SERENIoT a été conçu pour répondre à la problématique de sécurisation des maisons connectées. Comme exposé dans le chapitre 2, la majorité des solutions existantes ne sont en effet pas adaptées à cet environnement particulier où les utilisateurs ont souvent peu de connaissances en sécurité et ne sont pas en mesure de maintenir d'infrastructure de sécurité complexe. Nous avons ainsi conçu le système pour qu'il soit totalement autonome et nécessite un minimum d'actions de la part de l'utilisateur pour fonctionner. Tout dispositif IoT connecté à une sentinelle est en effet automatiquement protégé et aucune action de l'utilisateur n'est requise pour télécharger la politique de sécurité correspondante. L'ajout d'un dispositif IoT se résume alors à le connecter au réseau WiFi émis par la sentinelle, similairement à la connexion d'un dispositif IoT au réseau local. De même, la mise à jour des politiques de sécurité ne requiert aucune action de l'utilisateur et est effectuée de manière transparente pour l'utilisateur. Cette facilité d'utilisation nécessite cependant certains compromis et peut par exemple poser des problèmes aux utilisateurs plus avancés désireux d'avoir plus de contrôle sur les politiques de sécurités mises en vigueur pour leurs dispositifs.

5.2.1 Interruptions de service lors de la génération initiale des politiques de sécurité

Lors de la génération initiale des politiques de sécurité, les sentinelles bloquent les comportements réseau inconnus des dispositifs tant que les signatures de paquets correspondantes ne sont pas listées dans les politiques de sécurité. Cela permet de bloquer les comportements anormaux reportés seulement par une minorité de sentinelles. Il est cependant possible que certains comportements légitimes exposés tardivement soient aussi bloqués par les sentinelles avant d'être autorisés par les politiques de sécurité. Certains dispositifs peuvent alors ne pas être totalement fonctionnels tant que les politiques de sécurité correspondantes sont incomplètes. Bien que ces «interruptions de service» soient problématique en termes d'utilisabilité, elles n'impactent que les premiers utilisateurs qui connectent un dispositif inconnu à SERENIoT. La majorité des utilisateurs devraient ainsi directement bénéficier de politiques de sécurité complètes lors de la connexion de dispositifs grand public répandus et ne devraient pas observer d'«interruptions de service». Il est à noter que lors de nos expérimentations en laboratoire, ces «interruptions de service» se sont avérées relativement courtes (de l'ordre de quelques minutes). Une étude plus complète sur un large panel de dispositifs réels devrait cependant être menée pour évaluer l'impact de ces «interruptions de services» en vue d'un déploiement de SERENIoT.

5.2.2 Délai de mise à jour des politiques de sécurité

Au cours de nos tests nous avons observé que le point de rupture pour la mise à jour d'une politique de sécurité intervient lorsqu'environ 50% des sentinelles observent une même signature de paquet. Cela signifie que la moitié des sentinelles du système doivent observer une même signature de paquet pour avoir une influence sur la plus longue chaîne et inclure cette signature dans la politique de sécurité. Ainsi les dispositifs populaires ont moins de risques d'être compromis, car d'avantage de sentinelles doivent être «infectées» afin d'incorporer une signature malveillante dans la politique de sécurité. Cependant, ces dispositifs bénéficient d'une fréquence de mises à jour plus faible, car il faut attendre plus de retours de signatures de paquets avant que la mise à jour soit considérée comme légitime. Cela peut s'avérer problématique dans le cas d'une mise à jour changeant complètement l'empreinte réseau d'un dispositif. L'adoption d'une mise à jour par la majorité des utilisateurs peut en effet prendre des mois, rendant le dispositif inutilisable tant que la majorité des utilisateurs n'ont pas installé la mise à jour. Toutefois, les mises à jour logicielles pour un même produit sont majoritairement incrémentales et il est rare d'avoir une mise à jour provoquant un changement radical de comportement. Similairement, SERENIoT empêche aussi les *early adopters* de bénéficier des nouvelles fonctionnalités dès leur sortie. Pour palier à ce problème, il serait possible d'ajouter une option dans l'interface de SERENIoT afin de permettre aux experts de réinitialiser localement le comportement appris pour un dispositif afin de le traiter comme un nouveau dispositif. Cependant, l'utilisation de cette option se ferait aux dépens de la sécurité du dispositif concerné. Cela aurait sûrement pour effet d'assigner le dispositif mis à jour à une chaîne différente, moins populaire et donc plus susceptible d'être la cible d'acteurs malveillants.

5.2.3 Confirmation des blocs

Le nombre de confirmations d'un bloc désigne le nombre de blocs suivant dans la blockchain. Aussi, chaque nouveau bloc ajouté à la chaîne confirme les blocs précédents. Un bloc est considéré «confirmé» lorsque le seuil de confirmations requises est atteint. Tous les noeuds considèrent alors ce bloc et son contenu comme fiable. Le seuil de confirmations requises est de ce fait un paramètre important car il définit la vitesse de convergence du consensus et de résolution des forks.

Dans le cas de SERENIoT, les sentinelles considèrent uniquement les signatures des blocs confirmés. Les signatures listées dans les blocs confirmés sont ainsi ajoutées aux politiques de sécurité et les paquets réseau correspondants sont autorisés par les sentinelles. Le seuil de confirmations a donc un impact direct sur le contenu des politiques de sécurité et sur leur

délai de mise à jour :

— Un seuil de confirmations élevé augmente le temps nécessaire à la résolution des forks. Il augmente la résistance du système contre l'incorporation de signatures malveillantes, mais augmente aussi le risque que certains forks ne soient jamais résolus. De plus, un seuil de confirmation élevé entraîne une mise à jour des politiques de sécurité plus lentes.

— Un seuil de confirmations bas permet aux forks de se résoudre rapidement et aux sentinelles de converger rapidement. Cependant un seuil de confirmation bas augmente aussi les chances de confirmer un bloc contenant des signatures malveillantes.

Il faut donc trouver un compromis entre sécurité et réactivité du système en ajustant le seuil de confirmations. Lors de nos expérimentations, le seuil de confirmations était défini à 0 pour la chaîne de contrôle et à 3 pour les chaînes spécifiques aux dispositifs afin de privilégier la réactivité du système. Pour un déploiement hors laboratoire ces seuils devront être définis de lors de tests de mise en exploitation afin de trouver le meilleur compromis entre la sécurité du système et les délais de mise à jour des politiques de sécurité.

5.2.4 Confidentialité et vie privée

L'utilisation d'une blockchain publique peut soulever des problèmes de confidentialité et de vie privée. Le registre est en effet répliqué chez tous les participants et ces derniers ont accès aux données qui y sont stockées. En utilisant SERENIoT, le comportement des dispositifs IoT est ainsi publié dans une structure de données publique, libre d'accès. Un inconvénient de cette transparence réside dans le fait qu'il est possible de trouver la liste des chaînes auxquelles chaque sentinelle contribue en observant les blocs qu'elles publient. Cela peut permettre d'identifier les dispositifs IoT protégés par les différentes sentinelles et donc potentiellement la constitution de certains réseaux domestiques. Une solution à ce problème serait d'anonymiser totalement les blocs publiés par les sentinelles afin qu'il ne soit pas possible de relier un bloc à une sentinelle. Un autre inconvénient de l'architecture ouverte distribuée est que les sentinelles peuvent s'interroger entre elles pour savoir la disponibilité d'un bloc d'une chaîne particulière. Cela peut permettre à un attaquant connaissant le lien entre une blockchain et un modèle de dispositif de demander à une sentinelle si elle possède ce dispositif. Cette attaque peut être contrée en limitant le nombre de requêtes autorisées par adresse IP source et en ajoutant un délai d'attente exponentiel pour les requêtes provenant d'une même source.

L'utilisation d'une structure de données publique a aussi des avantages tels que la mise à disposition de la communauté d'une base de données complète et mise à jour en temps réel. En analysant le contenu des chaînes, SERENIoT pourrait par exemple fournir des statistiques

utiles à la communauté scientifique et aux experts en sécurité telles que la vitesse de déploiement, la fréquence et le taux d'adoption des mises à jour, la répartition des modèles de dispositifs, etc. La blockchain de SERENIoT pourrait aussi permettre aux auditeurs et régulateurs d'analyser les comportements des dispositifs pour mettre en place de nouvelles régulations sur les futurs dispositifs.

Architectures alternatives

La méthode de génération de politiques de sécurité de SERENIoT se base sur la blockchain. Cette dernière fournit le registre décentralisé permettant de stocker les politiques de sécurité et fait en sorte que les signatures inscrites dans le registre reflètent les observations de la majorité des sentinelles. Différentes architectures alternatives peuvent cependant être envisagées. Une alternative pourrait être d'avoir une entité centrale (par exemple un éditeur d'antivirus, une organisation à but non lucratif, etc.) gérant la génération et le stockage des politiques de sécurité. Avec cette approche, l'entité centrale maintient une base de données contenant les politiques de sécurité et met à disposition des sentinelles une API permettant de télécharger les mises à jour des politiques de sécurité et de reporter les nouvelles signatures de paquets. Cette approche possède certains avantages par rapport à la blockchain de SERENIoT :

— Elle est beaucoup moins complexe. La génération des politiques de sécurité est centralisée et gérée par l'entité centrale. Les sentinelles sont uniquement chargées de reporter les signatures qu'elles observent. Le registre contenant les politiques de sécurité n'est ainsi pas répliqué sur tous les participants et les sentinelles ne communiquent pas entre elles ce qui simplifie la topologie réseau du système.

— La technologie est plus mature. Cette architecture se retrouve en effet sur la majorité des services web et la mise en place d'une API REST et d'une base de données est un standard. Pour comparaison, la blockchain est une technologie plus récente qui est sûrement moins bien maîtrisée.

— Elle est moins énergivore qu'une blockchain exécutant l'algorithme de *proof of work* pour un déploiement à grande échelle.

Cependant cette solution souffre aussi de plusieurs inconvénients :

— Elle a un coût important pour l'entité centrale. Le coût pour maintenir l'accès et garantir la disponibilité d'un tel système augmente en effet avec le nombre de sentinelles.

— Elle est contrôlée par l'entité centrale qui peut décider des dispositifs supportés, des conditions pour qu'une sentinelle puisse bénéficier des politiques de sécurité, etc.

— L'entité centrale doit gérer l'identité des sentinelles pour se prémunir contre les attaques Sybil [41] et s'assurer qu'un attaquant ne puisse pas submerger le système avec de fausses sentinelles reportant de mauvaises signatures de paquets. Ce problème

non trivial complique la gestion du système et la mise en place des sentinelles. Pour comparaison, l'algorithme de consensus de la blockchain permet de se prémunir des attaques Sybil.

Aussi, bien que d'autres architectures soient envisageables, la blockchain semble être plus adaptée pour un système où la collecte des signatures de paquets des dispositifs IoT est effectuée par les sentinelles. La blockchain permet en effet de :

— Déployer à moindre coût un système hautement disponible fournissant des mises à jour de politiques de sécurité en temps réel.
— Rendre le système accessible sans restriction d'utilisation à certaines marques de dispositifs.
— Déployer un système résistant aux attaques Sybil grâce à l'utilisation d'un algorithme de consensus.

La blockchain permet aussi de s'affranchir tout système tiers en répartissant l'hébergement et la génération des politiques de sécurité sur les sentinelles. Cependant, il serait possible de mettre en place une blockchain «autorisée», sous contrôle d'une entité centralisée pour une utilisation commerciale de SERENIoT. Un tel système bénéficierait des avantages de la blockchain tout en donnant le contrôle sur les sentinelles autorisées à participer à un tiers.

5.2.5 Utilisation de SERENIoT pour la génération de MUD files

Ultimement, SERENIoT pourrait aussi servir à automatiser la génération de fichiers MUD [8]. Les fichiers MUD définissent les spécifications des dispositifs IoT selon un standard établi par l'IETF. Ces fichiers décrivent le trafic réseau des dispositifs et peuvent être utilisés en tant que politique de sécurité pour limiter l'accès réseau des dispositifs aux seules connexions définies par les spécifications. Ces fichiers sont normalement fournis par les fabricants. Cependant ce standard n'est pas encore correctement appliqué et de nombreux dispositifs n'ont pas de fichier MUD. SERENIoT pourrait aider à la génération de fichiers manquants en fournissant les spécifications pour les communications réseaux à partir des politiques de sécurité générées par les sentinelles.

La figure 5.1 montre un exemple de fichier MUD, tiré de mudmaker.org [53], et légèrement modifié pour l'exemple d'un dispositif IoT établissant des connexions IP vers l'API de son fabricant. La liste de contrôle d'accès surligné en bleu peut ainsi être fournie par SERENIoT. Bien qu'il soit possible de créer manuellement les fichiers MUD pour des dispositifs IoT en observant leur trafic réseau, l'utilisation de SERENIoT permet de bénéficier de listes de contrôle d'accès établies à partir de l'observation du comportement de nombreux dispositifs. Ces listes de contrôle d'accès sont plus fiables, car moins sensibles aux communications

Figure 5.1 Exemple de fichier MUD, tiré de mudmaker.org [53]

anormales observées localement sur un dispositif.

5.3 Travaux futurs et pistes d'améliorations

SERENIoT est un système expérimental conçu afin de valider qu'il est possible de déployer une solution autonome permettant la génération et la mise en vigueur de politiques de sécurités spécifiques pour réseaux IoT domestiques. Nous avons ainsi adopté l'approche de conception "Produit minimum viable" afin de valider les concepts sous-jacents préalablement au développement d'un produit fini. Cette approche nous a permis de nous concentrer sur les aspects les plus importants du système tout en prenant compte des limitations et des contraintes de ressources et de temps à notre disposition. Ainsi certaines décisions telles que l'utilisation de l'algorithme de *proof of work* ou l'utilisation d'un algorithme de génération de signatures «simple» ont été motivées par leur facilité d'implémentation lors de la réalisation du prototype. La version actuelle du système présente donc un certain nombre d'aspects qui peuvent être approfondis afin d'améliorer la stabilité en vue d'un déploiement à grande échelle. Cette section présente différentes pistes d'améliorations du système.

5.3.1 Algorithme de calcul des signatures de paquets

L'algorithme de calcul de signatures de paquets actuellement utilisé par SERENIoT permet de différencier les paquets des connexions des dispositifs IoT en fonction de l'hôte et du service contacté. Cette approche est relativement simple, mais permet de détecter les paquets anormaux issus de connexion vers des hôtes non répertoriés. Elle est suffisante pour valider le concept de génération de politiques de sécurité basée sur la blockchain, mais pourrait être améliorée dans une version future de SERENIoT. Une version améliorée de l'algorithme de signature pourrait :

— Inclure une analyse plus poussée de la payload IP et des données des services applicatifs. Il serait ainsi possible de prendre en compte la taille de la payload IP et d'analyser les champs non chiffrés des protocoles des couches applicatives. Le protocole TLS dispose par exemple d'une extension *Server Name Indication* qui permet au client de spécifier le nom de domaine avec lequel il souhaite établir la connexion dans le cas des hébergements mutualisés. Ce champ pourrait être utilisé afin d'identifier de manière plus précise le service contacté par le dispositif IoT dans le cas de connexions TLS chiffrées.

— Inclure des données comportementales comme par exemple la fréquence des échanges avec les différents hôtes, la quantité de données échangées. Cela permettrait de renforcer la détection de dispositifs IoT compromis dont le trafic réseau augmente fortement et diminuerait la capacité d'un attaquant à détourner un dispositif IoT en limitant la quantité de données possible de lui envoyer. Une analyse comportementale pourrait

aussi permettre de détecter les paquets anormaux issus d'hôtes «de confiance» infectés dans le cas par exemple de l'exploitation d'une vulnérabilité dans l'API du fabricant.
— Supporter les communications sur le réseau local afin d'empêcher un dispositif infecté d'en infecter d'autres sur le réseau local. Cela nécessiterait une adaptation de l'algorithme de calcul des signatures, car les dispositifs ne peuvent pas être identifiés par un nom de domaine ni par leur adresse IP locale (le plan d'adressage IP des réseaux locaux n'est en effet pas consistant pour toutes les installations). Il serait par exemple possible pour les communications locales de s'intéresser uniquement au contenu des paquets (qui sont ne sont généralement pas chiffrés), sans prendre en compte l'identité des dispositifs destinataires.

Ces améliorations nécessitent d'être étudié en détail pour assurer la consistance et l'unicité des signatures générées (comme spécifié dans la section 3.4). L'algorithme de génération de signatures devrait aussi être testé sur un ensemble complet de dispositifs IoT réels avec plusieurs dispositifs de chaque modèle afin de s'assurer de sa compatibilité et de son bon fonctionnement avec une large gamme de dispositifs.

5.3.2 Blockchain

La blockchain de SERENIoT est le composant clef permettant la génération de politiques de sécurité collaborative. Elle constitue le registre décentralisé permettant l'agrégation des signatures et la synchronisation des politiques de sécurité entre les sentinelles. L'ajout de données dans le registre est ainsi réglementé par l'algorithme de consensus qui fait en sorte que seules les signatures reportées par la majorité des sentinelles sont inscrites dans les politiques de sécurités. La blockchain de SERENIoT a été conçue avec pour objectif de faciliter l'implémentation d'un prototype et se base ainsi sur le concept initial de blockchain introduit par Satoshi Nakamoto en 2008 [42]. Certains aspects seront donc à améliorer dans les versions futures de SERENIoT afin d'améliorer les performances du système.

Algorithme de consensus

SERENIoT utilise actuellement l'algorithme de *proof of work*, tel que décrit dans la section 3.5. Cet algorithme est adapté pour l'implémentation d'un prototype du fait de sa simplicité, mais son utilisation n'est pas réaliste pour un cas d'utilisation réel. La majorité des sentinelles seraient en effet déployées sur des dispositifs avec peu de puissance de calcul tels que des routeurs. L'algorithme de *proof of work* repose sur des calculs intensifs pour sécuriser la blockchain et un attaquant disposant d'une grande puissance de calcul pourrait alors facilement dominer même un grand nombre de routeurs.

Pour pallier à ce problème, un algorithme de consensus alternatif peut être utilisé. En complément des algorithmes cités dans la section 2.3, de nouvelles alternatives prometteuses reposent sur le *Safe Guard Extension*(SGX) [1] [54, 55] des processeurs Intel [56] :

- L'algorithme de *Proof of Elapsed Time (PoET)* [57] repose sur le même principe que la PoW. Il introduit aléatoirement un délai lors la production des blocs pour qu'elle soit répartie uniformément entre les noeuds du réseau dans le temps. L'exécution du code introduisant le délai est certifiée par le SGX ce qui assure que tous les noeuds observent un temps d'attente aléatoire à chaque production de bloc.
- L'algorithme de *Robust Round Robin (RRR)* [58] sélectionne les noeuds tour à tour pour produire les blocs et répartit ainsi la production de blocs sur l'ensemble des participants du réseau. La sélection des noeuds se base sur le temps écoulé depuis le dernier bloc qu'ils ont miné. Ainsi le noeud n'ayant pas miné de bloc depuis le plus longtemps est sélectionné. Pour rejoindre le réseau, les noeuds nécessitent une identité unique fournie par SGX. Cela garantit que chaque participant est unique, car il n'est pas possible de générer plusieurs identités avec un même SGX.

Contrairement à la PoW, PoET et RRR ne nécessitent pas d'effort de calcul et sont de ce fait moins énergivore. Ils sont donc plus adaptés pour un déploiement sur des routeurs domestiques. Cependant ces méthodes fonctionnent actuellement uniquement sur les processeurs Intel et sont de ce fait exposé aux vulnérabilités de cette plateforme [59–61]. L'utilisation de PoET et RRR représente ainsi un compromis en termes de décentralisation. Avec ces algorithmes, la sécurité de la blockchain repose en effet intégralement sur l'utilisation d'un service fourni par un tiers (dans ce cas Intel) ce qui peut s'avérer problématique si ce tiers s'avère compromis ou malveillant.

La mise à jour de SERENIoT pour utiliser PoET ou RRR nécessiterait de les réimplémenter pour fonctionner dans la blockchain SERENIoT. PoET est en effet actuellement disponible uniquement sur le framework Hyperledger et aucun code n'est disponible pour RRR. Il faudrait aussi remplacer les Raspberry Pi de notre banc de test par des machines avec des processeurs Intel ayant SGX.

Expiration des blocs

La taille de la blockchain de SERENIoT sur le long terme peut s'avérer problématique en vue d'un déploiement sur des dispositifs tels que des routeurs domestiques ne disposant que d'un espace de stockage limité. Nous avons en effet calculé dans la section 4.2.3 que la blockchain

1. SGX est une plateforme mettant à disposition des développeurs un environnement vérifiable capable d'attester l'exécution de certaines instructions.

de SERENIoT atteindrait 511Go après une année de fonctionnement pour 10 000 modèles de dispositifs protégés. Une piste de solution serait de supprimer les blocs expirés après une certaine période de temps. Cela permettrait de :

— Limiter la taille de la blockchain en ne conservant qu'un nombre limité de blocs.
— Forcer les sentinelles à mettre à jour les politiques de sécurité afin de supprimer les signatures de paquets correspondant à des comportements obsolètes.

Pour ce faire, il serait par exemple possible de conserver uniquement les 1000 derniers blocs. Cependant, une telle modification bouleverse complètement le principe d'immutabilité de la blockchain selon lequel l'intégralité de la chaîne de blocs doit être conservée. Les répercussions d'une telle modification en termes de sécurité et de fiabilité de la blockchain sont à éclaircir et une étude approfondie sur le sujet est donc nécessaire. De plus, la mise à jour des politiques de sécurité pour supprimer les comportements obsolètes pourrait avoir des répercussions néfastes sur la minorité de dispositifs n'ayant pas été mis à jour. La suppression des anciennes signatures de paquets pourrait ainsi les rendre inutilisables.

5.3.3 Evaluation

Evaluation fonctionnelle sur un banc de dispositifs réels

L'évaluation fonctionnelle sur dispositifs réels réalisée dans cette étude valide le concept de SERENIoT. Il serait cependant intéressant d'évaluer le système lors de différents tests sur un ensemble plus large de dispositifs. Un test grandeur nature pourrait ainsi être réalisé avec une centaine de maisons connectées afin de valider le système en conditions réelles. Cela permettrait d'exposer SERENIoT à de réelles menaces et d'observer l'efficacité des sentinelles à générer des politiques de sécurité hors laboratoire. De plus un test de compatibilité avec un grand nombre de dispositifs IoT pourrait aussi être réalisé afin de définir de manière plus précise le spectre de dispositifs compatibles avec le système. Ces expériences nécessiteraient différents modèles de dispositifs IoT grand public en plusieurs exemplaires.

Evaluation d'utilisabilité

Une évaluation d'utilisabilité pourrait aussi être menée sur un groupe d'utilisateurs afin de vérifier que les sentinelles s'intègrent correctement dans l'environnement de la maison connectée et correspondent au public visé. Cette evaluation permettrai aussi de valider que SERENIoT n'interfère pas avec les habitudes des utilisateurs et que les dispositifs qu'ils utilisent au quotidien ne sont pas perturbés par la mise en vigueur des politiques de sécurité. Pour mener à bien cette expérience, une collaboration avec des experts en interface homme-machine et

en utilisabilité pourrait être établie afin de définir les critères du test et de regrouper un échantillon d'utilisateurs représentatif du public ciblé par SERENIoT.

5.4 Conclusion

SERENIoT capitalise sur la simplicité et la constance de l'empreinte réseau des dispositifs IoT pour générer des politiques de sécurité et sécuriser les réseaux IoT domestiques. Ces politiques de sécurité viennent en effet compléter les systèmes de sécurité classiques tels que les pare-feu domestiques souvent trop laxistes à l'égard des dispositifs IoT.

Lors des expériences que nous avons mené, SERENIoT a été en mesure de générer correctement des politiques de sécurité et de les mettre en vigueur pour bloquer le trafic parasite injecté. Le test du système avec des dispositifs réels démontre ainsi que les sentinelles fonctionnent correctement avec des dispositifs IoT réels d'un même modèle. La politique de sécurité générée lors de cette expérience (tableau 4.1) reflète la simplicité du comportement des dispositifs protégés et conforte nos observations sur la simplicité et la constance des empreintes réseau des dispositifs IoT réalisées lors de l'analyse des données de la base de données de Alrawi et al. [2]. De plus, le test du système avec des dispositifs simulés montre que SERENIoT fonctionne correctement avec la majorité dispositifs IoT grand publique à notre disposition. Les politiques de sécurité à appliquer pour les sécuriser peuvent donc être apprises en observant leurs comportements sur les différents sites où ils sont installés. Les comportements majoritaires sont alors considérés «normaux» et les comportements observés chez une minorité sont bloqués. Ainsi, la configuration des politiques de sécurité et la construction des spécifications fonctionnelles des dispositifs ne reposent plus sur l'utilisateur final ni sur les fabricants et SERENIoT apprend dynamiquement les politiques de sécurité à appliquer.

L'utilisation d'une blockchain permet à SERENIoT de s'appuyer sur un consensus décentralisé pour agréger les signatures de paquets observées par les sentinelles en une politique de sécurité. La blockchain publique facilite aussi l'accessibilité du système en permettant aux utilisateurs de rejoindre le réseau SERENIoT sans avoir à payer d'abonnement ou à acheter une marque de dispositifs particulière. SERENIoT est ainsi compatible avec une large gamme de dispositifs sans prérequis de marque. L'architecture décentralisée de la blockchain permet aussi à SERENIoT d'être hautement redondant et disponible à moindre coût et de fonctionner sur le long terme, tant qu'il y a des sentinelles. Ainsi, contrairement à un système centralisé maintenu par un tiers, SERENIoT n'a pas de seuil de rentabilité et n'a pour seul objectif que de générer des politiques de sécurité.

SERENIoT est conçu pour fonctionner sans aucune interaction avec l'utilisateur. Nous es-

pérons que cette facilité d'utilisation encouragera l'adoption de tels systèmes par le grand public. Cette facilité d'utilisation nécessite cependant certains compromis. Par exemple, dans le cas d'une mise à jour logicielle qui modifierait le comportement d'un dispositif IoT, le système empêche l'utilisation de nouvelles fonctionnalités tant que la majorité des sentinelles n'observent pas aussi ce changement de comportement. Un faible taux d'installation d'une mise à jour empêcherait alors les *early adopters* d'utiliser les nouvelles fonctionnalités tant que la majorité des utilisateurs n'ont pas installé la mise à jour.

Enfin, nous avons évoqué différentes pistes pour améliorer la prochaine version de SERENIoT. L'algorithme de calcul des signatures de paquets pourrait ainsi analyser plus en profondeur la payload des paquets IP. La blockchain peut aussi être améliorée en remplaçant l'algorithme de consensus par un algorithme moins énergivore et plus adapté à être exécuté sur des routeurs domestiques. Une fonctionnalité d'expiration des blocs pourrait aussi être mise en place afin de limiter la taille de la blockchain sur le long terme, mais cette fonctionnalité nécessite cependant une étude approfondie afin de mesurer les répercussions sur la fiabilité et la sécurité de la blockchain.

CHAPITRE 6 CONCLUSION

Ce chapitre clôt le mémoire et présente les travaux réalisés au cours de cette recherche ainsi que notre contribution. Il revient ensuite sur les limitations de notre solution et sur les pistes d'amélioration future de SERENIoT.

6.1 Synthèse des travaux

6.1.1 Retour sur les objectifs de recherche

Cette recherche nous a permis de répondre à la question suivante :

Est-il possible de déployer une solution autonome permettant la génération et la mise en vigueur de politiques de sécurités spécifiques pour réseaux IoT domestiques ?

Pour cela, nous avons rempli les objectifs suivants :

1. Identifier les enjeux de la sécurisation des réseaux IoT domestiques. Lors de la revue de littérature et de l'étude des solutions existantes présentée dans le chapitre 2, nous avons identifié que les maisons connectées présentent de nombreuses vulnérabilités et sont régulièrement la cible d'attaques. Nous avons aussi observé que la majorité des solutions actuelles ne conviennent pas aux environnements domestiques car elles sont trop complexes à utiliser ou à déployer.
2. Valider qu'une majorité de dispositifs IoT domestiques ont une empreinte réseau simple et constante. Nous avons vérifié durant nos expérimentations 4.2.2 que les dispositifs IoT à notre disposition avaient une empreinte simple, composée d'une dizaine de signatures, et majoritairement constante durant la période d'observation.
3. Concevoir un système permettant de générer et de mettre en vigueur des politiques de sécurité réseau pour dispositifs IoT domestiques sans intervention humaine. Nous avons conçu SERENIoT, un système novateur de génération et de mise en vigueur de politiques de sécurité pour les réseaux IoT domestiques. Nous avons ainsi créé un mécanisme basé sur une blockchain permettant d'agréger des signatures de paquets pour générer des politiques de sécurité. Nous avons aussi développé une méthode de calcule de signatures de paquets assez générales pour générer les mêmes signatures pour tous les dispositifs d'un même modèle, mais assez précises pour différencier les communications vers des hôtes malveillants. La méthode de génération de politiques de sécurité de SERENIoT est détaillée dans le chapitre 3.

4. Valider le fonctionnement du système par test sur un réseau IoT réel. Nous avons réalisé une expérience sur un banc de test réel dans la section 4.2.1. Nous avons observé que le prototype de SERENIoT a correctement généré et mis en vigueur les politiques de sécurité pour les dispositifs utilisés lors de l'expérience.
5. Valider la compatibilité du système avec différents dispositifs IoT par simulation. Nous avons réalisé une expérience sur un banc de test simulé dans la section 4.2.2. Nous avons observé que le prototype de SERENIoT a correctement généré et mis en vigueur les politiques de sécurité lors de l'expérience ce qui valide sa compatibilité avec ces dispositifs.
6. Valider l'extensibilité du système par simulation. Lors des simulations, nous avons aussi observé que SERENIoT supportait correctement l'augmentation du nombre de dispositifs et de sentinelles.
7. Améliorer le système à l'aide des résultats expérimentaux. Nous avons amélioré le système et corrigé les bugs observés au cours des nombreux tests et simulations réalisées. Nous présentons d'autres pistes d'améliorations dans le chapitre 5.

6.1.2 Contribution

Notre principale contribution est une nouvelle approche permettant de générer de manière autonome des politiques de sécurité pour des systèmes dont le comportement est déterminé par un nombre limité d'actions. Cette contribution représente aussi une application concrète de la technologie blockchain hors du domaine classique de la fintech.

6.2 Limitations de la solution proposée

La taille de la blockchain sur le long terme et la puissance de calcul nécessaire pour exécuter l'algorithme de consensus sont peu adaptés pour un déploiement de SERENIoT sur des routeurs domestiques. Cependant, ces problèmes peuvent être adressés et nous discutons des possibles pistes d'amélioration dans le chapitre 5. L'accès public aux données de la blockchain peut aussi poser des problèmes de confidentialités en permettant à un attaquant de découvrir les dispositifs IoT présents au sein d'un ménage. Ceci peut être corrigé en anonymisant les blocs de la blockchain et en prenant des mesures contre les sentinelles effectuant trop de demandes de blocs.

Le délai lors de la génération et des mises à jour des politiques de sécurité peut être problématique pour les *early adopters* qui ne peuvent pas profiter pleinement des fonctionnalités de leurs dispositifs tant que les politiques de sécurité finalisées. Différentes solutions peuvent

être envisagées comme l'ajout d'une option permettant de désactiver temporairement la mise en vigueur des politiques de sécurité pour les utilisateurs expérimentés.

Notre système ne monitore que les connexions *LAN-to-WAN* et ne protège donc pas les dispositifs IoT des autres dispositifs infectés sur le réseau local. Bien que cette limitation rende possibles les infections paires à pair sur le réseau local, les dispositifs infectés localement ne seront pas en mesure d'attaquer des hôtes distants. En effet, les sentinelles bloquent tout trafic sortant ne correspondant pas à la liste blanche.

Notre système a été conçu pour fonctionner avec des dispositifs ayant une empreinte réseau constituée d'une dizaine de signatures. L'efficacité de notre système pour les systèmes plus complexes n'est pas garantie et une recherche plus approfondie doit être menée pour déterminer si un système de détection d'intrusion collaboratif peut converger sur une liste d'actions autorisées pour des dispositifs plus complexes. Une stratégie pour les dispositifs complexes pourrait être d'adopter un algorithme de signature de paquets moins spécifique, mais cela pourrait avoir comme inconvénients de laisser passer certaines attaques.

La version actuelle de SERENIoT bloque les connexions spécifiques aux utilisateurs (par exemple un FTP dans le cloud pour sauvegarder un flux vidéo). Les connexions vers des hôtes définis par les utilisateurs seront bloquées, car le trafic vers des hôtes arbitraires n'est pas observé par une majorité de sentinelles. Une option pour autoriser les connexions spécifiques aux utilisateurs serait d'ajouter une option dans l'interface graphique permettant aux utilisateurs avancés d'autoriser certaines connexions spécifiques sans impacter la liste blanche collective.

Enfin, les performances de détection et de résistance aux attaques de SERENIoT s'améliore à chaque déploiement de sentinelles supplémentaire. En déployant plus de sentinelles, l'effort requis pour un attaquant afin de manipuler la liste blanche (c.-à-d. la blockchain) augmente. À l'inverse, dans le cas d'un petit nombre de sentinelles, la possibilité d'une attaque influençant les signatures de paquets qui sont incluses dans la liste blanche est plus grande.

6.3 Améliorations futures

Différentes pistes d'amélioration de SERENIoT ont été énoncées dans le chapitre 5. Certains aspects de la blockchain peuvent ainsi être retravaillés. L'algorithme de consensus pourrait être remplacé par un algorithme moins énergivore que la *proof of work* et plus adapté à être déployé sur des dispositifs réseau domestique tels que des routeurs. Une solution telle que l'expiration des blocs de la blockchain pourrait aussi être envisagée afin de réduire la taille de la blockchain sur le long terme et de forcer les sentinelles à «mettre à jour» les politiques

en supprimant les signatures de paquets correspondant à des comportements obsolètes. L'algorithme de calcul de signatures de paquets pourrait être amélioré en prenant en effectuant une analyse plus poussée des *payloads* des paquets IP.

RÉFÉRENCES

[1] D. Kumar *et al.*, “All things considered : An analysis of IoT devices on home networks,” dans *28th USENIX Security*, 2019.

[2] O. Alrawi *et al.*, “SoK : Security evaluation of home-based IoT deployments,” dans *IEEE 40th Symposium on Security and Privacy (S&P)*, 2019.

[3] E. Fernandes, J. Jung et A. Prakash, “Security analysis of emerging smart home applications,” dans *IEEE 37th Symposium on Security and Privacy (S&P)*, 2016.

[4] Ray et M. Huebler, “Moving from hacking IoT gadgets to breaking into one of europe’s highest hotel suites,” Black Hat USA, 2019.

[5] L. H. Newman, “An elaborate hack shows how much damage IoT bugs can do,” 2018. [En ligne]. Disponible : https://www.wired.com/story/elaborate-hack-shows-damage-iot-bugs-can-do/

[6] M. Antonakakis *et al.*, “Understanding the Mirai botnet,” dans *26th USENIX Security*, 2017.

[7] D. Barrera, I. Molloy et H. Huang, “IDIoT : Securing the internet of things like it’s 1994,” *arXiv preprint arXiv :1712.03623*, 2017.

[8] E. Lear, D. Romascanu et R. Droms, “Manufacturer Usage Description Specification,” Internet Engineering Task Force, Rapport technique 8520, 2019. [En ligne]. Disponible : https://tools.ietf.org/html/rfc8520

[9] “Gartner says 5.8 billion enterprise and automotive IoT endpoints will be in use in 2020,” Gartner, août 2019. [En ligne]. Disponible : https://www.gartner.com/en/newsroom/press-releases/2019-08-29-gartner-says-5-8-billion-enterprise-and-automotive-io

[10] V. Négrier et B. Régent, “Observatoire de la vie connectée,” 2017. [En ligne]. Disponible : https://www.emota.eu/media/1248/observatoire-de-la-vie-connect%C3%A9e.pdf

[11] Statista. Smart home market. [En ligne]. Disponible : https://www.statista.com/outlook/279/108/smart-home/canada?currency=cad

[12] L. Knud Lasse, “State of the IoT 2018 : Number of IoT devices now at 7b – market accelerating,” 2018. [En ligne]. Disponible : https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/

[13] S. Soltan, P. Mittal et H. V. Poor, “BlackIoT : IoT botnet of high wattage devices can disrupt the power grid,” dans *27th USENIX Security*, 2018.

[14] Underwriters Laboratories, “Cybersecurity considerations for connected smart home systems and devices,” 2018. [En ligne]. Disponible : https://industrie-4-0.ul.com/wp-content/uploads/2018/02/UL_Cybersecurity_SmartHome_White_Paper_en.pdf

[15] Apple. (2020) Use fall detection with Apple watch. [En ligne]. Disponible : https://support.apple.com/en-us/HT208944

[16] Vera. Vera smart controllers. [En ligne]. Disponible : https://getvera.com/

[17] OpenWrt. OpenWrt project. [En ligne]. Disponible : https://openwrt.org/

[18] Avast. Avast smart home security report. [En ligne]. Disponible : https://press.avast.com/hubfs/media-materials/kits/smart-home-report-2019/Report/Avast%20Smart%20Home%20Report_EN.pdf?hsLang=en

[19] A. Blaich et A. Hay, “Hello barbie initial security analysis,” 2016. [En ligne]. Disponible : https://static1.squarespace.com/static/543effd8e4b095fba39dfe59/t/56a66d424bf1187ad34383b2/1453747529070/HelloBarbieSecurityAnalysis.pdf

[20] J. Max, “Backdooring the frontdoor,” 2016. [En ligne]. Disponible : https://media.defcon.org/DEF%20CON%2024/DEF%20CON%2024%20presentations/DEF%20CON%2024%20-%20Jmaxxz-Backdooring-the-Frontdoor.pdf

[21] D. Meyer, “How criminals can mine cryptocurrency with your poorly-secured smart devices,” 2018. [En ligne]. Disponible : https://fortune.com/2018/03/01/mine-cryptocurrency-hack-iot-smart-devices-avast/

[22] A. Kumar et T. J. Lim, “Early detection of Mirai-like IoT bots in large-scale networks through sub-sampled packet traffic analysis,” dans *Future of Information and Communication Conference*, 2019.

[23] J. Habibi *et al.*, “Heimdall : Mitigating the internet of insecure things,” *IEEE Internet of Things Journal*, vol. 4, n°. 4, p. 968–978, 2017.

[24] Y. Meidan *et al.*, “N-BaIoT—Network-based detection of IoT botnet attacks using deep autoencoders,” *IEEE Pervasive Computing*, vol. 17, n°. 3, p. 12–22, 2018.

[25] R. Sommer et V. Paxson, “Outside the closed world : On using machine learning for network intrusion detection,” dans *IEEE 31st Symposium on Security and Privacy (S&P)*, 2010.

[26] M. Miettinen *et al.*, “IoT sentinel : Automated device-type identification for security enforcement IoT,” dans *IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, 2017.

[27] S. Goutam, W. Enck et B. Reaves, “Hestia : Simple least privilege network policies for smart homes,” dans *12th Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*, 2019.

[28] T. Golomb, Y. Mirsky et Y. Elovici, “CIoTA : Collaborative IoT anomaly detection via blockchain,” *arXiv preprint arXiv :1803.03807*, 2018.

[29] Bitdefender. Bitdefender BOX - home network security for all connected. [En ligne]. Disponible : https://www.bitdefender.com/box/

[30] Norton. Norton core router - secure WiFi router. [En ligne]. Disponible : https://us.norton.com/core

[31] ——. How do i purchase norton core ? [En ligne]. Disponible : https://support.norton.com/sp/en/ca/norton-core-security/current/solutions/v131932667

[32] P. C. Van Oorschot, *Computer Security and the Internet : tools and jewels.* Springer, 2020, p. 314–314.

[33] S. Ruoti *et al.*, “SoK : Blockchain technology and its potential use cases,” *arXiv preprint arXiv :1909.12454*, 2019.

[34] C. Cachin et M. Vukolić, “Blockchain consensus protocols in the wild,” dans *31st International Symposium on Distributed Computing*, 2017.

[35] A. Back, “A partial hash collision based postage scheme,” 1997. [En ligne]. Disponible : http://www.hashcash.org/papers/announce.txt

[36] A. Juels et J. Brainard, “Client puzzles : A cryptographic defense against connection depletion attacks,” dans *NDSS*, 1999.

[37] A. Kiayias *et al.*, “Ouroboros : A provably secure proof-of-stake blockchain protocol,” dans *Springer Advances in Cryptology (CRYPTO)*, 2017.

[38] D. M. Mendez Mena et B. Yang, “Blockchain-based whitelisting for consumer IoT devices and home networks,” dans *19th Annual SIG Conference on Information Technology Education*, 2018.

[39] “Your smart fridge could be mining Bitcoins for criminals,” CBC News, juin 2018. [En ligne]. Disponible : https://www.cbc.ca/news/technology/bitcoin-hacking-smart-devices-1.4728222

[40] Cisco. Introduction to Cisco IOS NetFlow - a technical overview. [En ligne]. Disponible : https://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/ios-netflow/prod_white_paper0900aecd80406232.html

[41] J. J. Douceur, “The sybil attack,” dans *1st International Workshop on Peer-to-Peer Systems*, January 2002.

[42] S. Nakamoto *et al.*, “Bitcoin : A peer-to-peer electronic cash system,” 2008.

[43] “Google denies Xiaomi access over security bug,” BBC, janv. 2020. [En ligne]. Disponible : https://www.bbc.com/news/technology-50981993

[44] Raspap. Simple AP setup & WiFi management for debian-based devices. [En ligne]. Disponible : https://raspap.com/

[45] Netfilter. The netfilter.org "iptables" project. [En ligne]. Disponible : https://www.netfilter.org/projects/iptables/index.html

[46] WebRTC. Real-time communication for the web. [En ligne]. Disponible : https://webrtc.org/

[47] I. Fette et A. Melnikov, "The websocket protocol," Internet Engineering Task Force, Rapport technique 6455, 2011. [En ligne]. Disponible : https://tools.ietf.org/html/rfc6455

[48] VueJS. The progressive javascript framework. [En ligne]. Disponible : https://vuejs.org/

[49] H. Kang *et al.*, "IoT network intrusion dataset," 2019. [En ligne]. Disponible : http://dx.doi.org/10.21227/q70p-q449

[50] Docker. Runtime metrics. [En ligne]. Disponible : https://docs.docker.com/config/containers/runmetrics/

[51] S. Kenin, "BrickerBot mod_plaintext Analysis," 2017. [En ligne]. Disponible : https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/brickerbot-mod_plaintext-analysis/

[52] S. Herwig *et al.*, "Measurement and analysis of Hajime, a peer-to-peer IoT botnet." dans *NDSS*, 2019.

[53] Mudmaker. MUD file example. [En ligne]. Disponible : https://www.mudmaker.org/cloud-service.json

[54] V. Costan et S. Devadas, "Intel SGX explained," Cryptology ePrint Archive, Report 2016/086, 2016. [En ligne]. Disponible : https://eprint.iacr.org/2016/086

[55] M. Hoekstra *et al.*, "Using innovative instructions to create trustworthy software solutions," dans *2nd International Workshop on Hardware and Architectural Support for Security and Privacy*, 2013.

[56] Intel. Software Guard Extensions. [En ligne]. Disponible : https://software.intel.com/en-us/sgx

[57] Hyperledger. PoET 1.0 specification. [En ligne]. Disponible : https://sawtooth.hyperledger.org/docs/core/releases/latest/architecture/poet.html

[58] M. Ahmed-Rengers et K. Kostiainen, "Don't mine, wait in line : Fair and efficient blockchain consensus with robust round robin," *arXiv preprint arXiv :1804.07391v3*, 2020.

[59] J. Van Bulck *et al.*, "Foreshadow : Extracting the keys to the Intel SGX kingdom with transient out-of-order execution," dans *27th USENIX Security*, 2018.

[60] J. Götzfried *et al.*, “Cache attacks on Intel SGX,” dans *10th European Workshop on Systems Security*, 2017.

[61] N. Weichbrodt *et al.*, “Asyncshock : Exploiting synchronisation bugs in Intel SGX enclaves,” dans *21st European Symposium on Research in Computer Security*, 2016.