Master 1 Informatique

PJI - Projet Individuel - Sujet no 104

Systèmes de détection d'intrusion pour l'Internet des Objets

Auteurs :
M. Théo Plockyn
M. Rémy Debue

Encadrant :
Pr. Gilles Grimaud

Version 0.5 du
17 mai 2016

Remerciements

Nous remercions tout d'abord l'équipe pédagogique, administrative et intervenants du Master 1 informatique de nous avoir encadré, aidé et assuré les enseignements dont nous avons disposé cette année.

Nous tenons aussi à remercier et à témoigner notre reconnaissance aux personnes suivantes :

Gilles Grimaud, notre encadrant, pour nous avoir proposé le sujet, nous avoir suivi et conseillé tout au long de ce projet.

Michaël Hauspie, pour ses consignes et sa participation dans les décisions du déroulement du projet.

Nadir Cherifi, pour son aide précieuse et ses connaissances des technologies utilisées qui nous ont débloqué à plusieurs reprises.

Samuel Hym, François Serman, Christophe Bacara, Quentin Bergougnoux, et toute l'équipe 2XS pour leur accueil sympathique et leur soutien tout au long de ce projet.

Table des matières

Table des figures

Liste des sigles et acronymes

6LoWPAN	*IPv**6** **Lo**w power **W**ireless **P**ersonal **A**rea **N**etworks*
LoWPAN	***Lo**w power **W**ireless **P**ersonal **A**rea **N**etworks*
IRCICA	**I**nstitut de **r**echerche sur les **c**omposants logiciels et matériels pour l'**i**nformation et la **c**ommunication **a**vancée de Lille
2XS	*e**X**tra **S**mall e**X**tra **S**afe* – L'équipe de recherche
CFS	***C**offee **F**ile **S**ystem* – Le système de fichier de Contiki
DSL	***D**omain **S**pecific **L**anguage* – Langage dédié
IDS	***Intrusion Detection System*** – Système de Detection d'Intrusions
PJI	**P**roj**j**et **i**ndividuel
RFC	***R**equest **F**or **C**omments* – Documents de spécifications
OS	***O**perating **S**ystem* – Système d'exploitation

Introduction

Dans le cadre de notre cursus en Master Informatique à Lille 1, nous avons eu lopportunité de réaliser un projet sur lensemble du semestre appelé PJI. Chaque étudiant ou binôme pouvait choisir un sujet sur lequel travailler parmi une liste mais également proposer le sien. Nous nous sommes intéressés à un sujet proche de l'informatique embarquée, plus particulièrement dans le domaine de l'Internet des Objets. Notre sujet se porte sur la détection d'attaques dans un réseau 6LoWPAN.

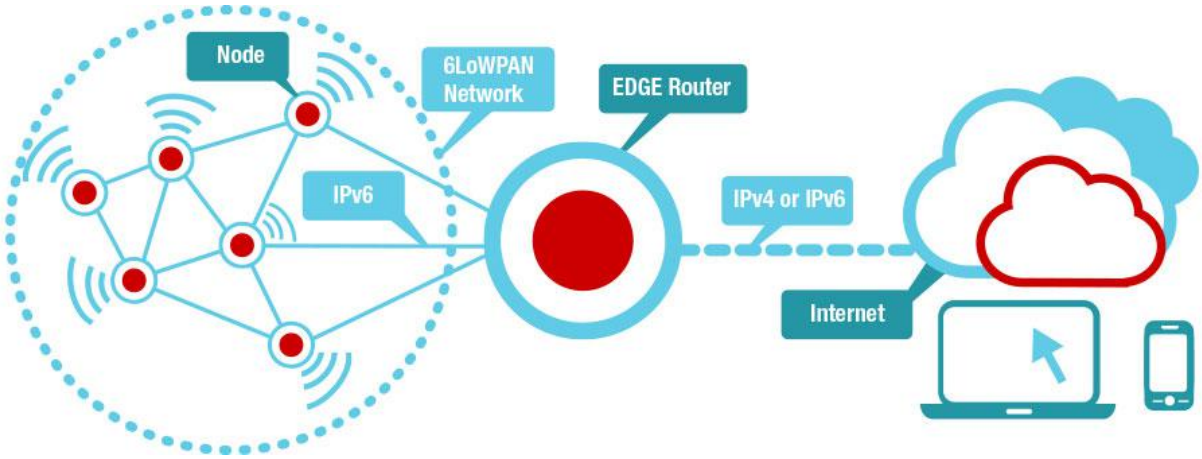

FIGURE 1 – Diagramme d'explication de 6LoWPAN.

L'équipe de recherche proposant ce sujet est le groupe 2XS **eXtra Small eXtra Safe** composée de notamment **Gilles GRIMAUD** notre encadrant, **Michael HAUSPIE** son collègue proche de ce sujet et bien sûr le reste de l'équipe. L'équipe se focalise sur les problématiques de sécurité dans les systèmes embarqués contraints, notamment fournir des solutions logicielles prouvées.

Chapitre 1

Contexte du sujet

Notre projet est de produire une sonde qui renifle (ou sniffe) le trafic réseau dans le contexte de l'Internet des Objets. Cette sonde est un noeud dans ce réseau, et a pour but de transmettre des informations utiles à sa sécurisation et dans une certaine mesure à l'analyse de ces informations.

1.1 Analyse de l'existant

1.1.1 Internet des objets

L'Internet des Objets, ou Internet of Things en anglais, correspond à l'extension d'internet aux éléments ou lieux du monde physique, là où l'internet habituel s'arrête au domaine du virtuel. Cette technologie est implantée dans notre société avec diverses applications comme la domotique, le médical, la gestion des déchets, mais pas limité à ceux là. Notre projet s'inscrit donc dans cet univers puisque les sondes surveillent le trafic de différents éléments d'un sous-réseau d'objets physiques, d'un bâtiment par exemple.
L'Internet des objets regroupe différents modes de communications entre les noeuds d'un réseau tels que le Wi-Fi, le courant porteur ou le bluetooth.

1.1.2 Technologies de communication

6LoWPAN est une spécification du principe des LoWPAN, c'est à dire un ensemble d'équipements aux ressources limitées, puissance, autonomie entre autres, reliés dans un réseau au débit limité. Typiquement, ces réseaux sont constitués d'un grand nombre d'éléments ou noeuds dans le réseau. Basé sur l'IPv6, quelques problèmes se posent avec la spécification standard de celui ci. Ce protocole de communication possède une taille d'entête importante, couplée aux contraintes de tailles de paquets imposées, cela pose des soucis

de fragmentation et de réassamblage excessif pour des contrôleurs aux capacités limitées. La spécification de 6LoWPAN et ses RFC (4919 et 4944) définissent donc des solutions à ces problèmes, et on peut aujourd'hui utiliser plusieurs implémentations de 6LoWPAN, tel que ZigBee. Linky, le nouveau compteur communicant d'ERDF utilise cette technologie. Bien sûr, on peut trouver pléthore de projets et d'objets de domotique se servant de la spécification et de ses implémentations pour communiquer.

1.1.3 Sécurité des communications

Les différentes RFC (Request For Comment) définissent un ensemble de consignes sur l'implémentation de la sécurité des réseaux 6LoWPAN notamment sur les différentes couches de la pile protocolaire.

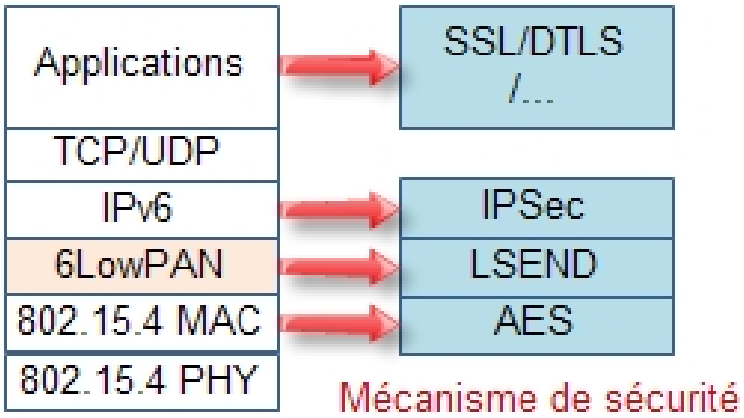

FIGURE 1.1 – Diagramme d'explication de la sécurité des couches de 6LoWPAN.

— Sur la couche MAC : l'algorithme AES (Advanced Encryption Standard) doit être utilisé pour sécuriser la couche liaison.
— Sur la couche réseau : l'utilisation d'IPsec (Internet Protocol Security) est possible, mais coûteuse et un échange de clés habituel pour le chiffrement n'est pas possible. Une extension du protocole SEND – ***SE**cure **N**eighbor **D**iscovery protocol* – (RFC 3971) permettant de sécuriser ce mécanisme a été mis en place pour les réseaux 6LoWPAN, appelé LSEND – ***L**ightweight **SE**cure **N**eighbor **D**iscovery protocol* –.
— Sur la couche application : une solution possible est de mettre en place la sécurisation via SSL.

Dans les faits, la sécurité étant difficile à mettre en place à cause des contraintes de l'embarqué (performance, stockage des données), elle est parfois insuffisante pour garantir un échange de données sécurisées.

1.2 Objectif du projet

1.2.1 Détail du projet

Nous avons expliqué l'intitulé du projet mais non pourquoi ces sondes pouvaient être utiles. Les sondes sont des noeuds, ou mote dans le vocabulaire de Contiki, qui vont sonder et analyser le trafic circulant. Les informations qu'elles récupèrent permettent la détection d'intrusions.
En effet, le trafic et les paquets sont soumis à des formats spécifiques, contenant des informations qui doivent s'y conformer. Dans le cas d'écarts par rapport au format réglementaire, on se retrouve face à une anomalie, et possiblement une attaque.
La difficulté de cette analyse réside autant dans les contraintes du matériel, qui est limité en puissance et en mémoire, que dans les solutions mises en places par le réseau pour pallier à ces contraintes, par exemple la compression d'en-tête.

1.2.2 Où s'inscrit le projet ?

Comme illustré précédemment, l'Internet des Objets trouve son utilité dans de nombreux domaines. Certaines applications, comme l'industrie ou le médical, font circuler des données sensibles sur le réseau, et des personnes mal intentionnées pourraient causer de graves problèmes sans être détectés si le réseau n'est pas protégé.
On peut retrouver d'autres exemples ayant moins de conséquences comme les hackers changeant le nombre de places des panneaux de parking par des injures. Dans cet exemple, la personne a simplement usurpé l'identité de la machine qui met à jour les places, et a envoyé des paquets falsifiés contenant ces injures grâce à un manque de sécurité sur le réseau.

1.3 Réponse à un besoin de l'équipe

1.3.1 Focalisation sur la sécurité par 2XS

L'équipe de recherche 2XS se focalise sur la création de systèmes sûrs, par le biais de différents mécanismes, dont les preuves formelles de programmes, le développement de librairies, et bien d'autres projets visant à sécuriser les systèmes.

La problématique de la sécurité ne s'arrête pas à la frontière du système en lui-même, il communique avec d'autres. C'est là que les failles et les fuites d'informations sont les plus nombreuses, même si l'intégrité du système n'est pas en cause.
L'un de leurs projets pour répondre à ce problème est Discus.

1.3.2 Discus

Discus est une architecture d'IDS – Système de détection d'intrusion – massivement distribuée, qui est configurée grâce à un DSL – Langage dédié – Discus-script. Le principe de Discus est d'abstraire la définition des contraintes de sécurité sur un réseau, qu'il soit ethernet, bluetooth ou Wi-Fi.
Notre projet est donc directement en rapport avec celui-ci, fournissant la couche matérielle nécessaire à Discus pour analyser le réseau afin d'y appliquer ces contraintes de sécurité.

1.4 Technologies et systèmes utilisés

Pour développer notre sonde renifleuse, nous avons utilisé plusieurs outils que nous allons présenter ici :

1.4.1 Contiki

Contiki est un système d'exploitation léger et flexible avec pour cible les capteurs miniatures en réseau. Ses atouts sont sa flexibilité, sa portabilité, sa faible consommation énergétique, et surtout dans notre cas, son support des protocoles IPv6 et 6LoWPAN. Il répond à une attention importante de la communauté scientifique portée aux réseaux de capteurs sans fil. Il a été créé par une équipe du centre suédois de recherche scientifique SICS.

1.4.2 Outils de simulations

Notre sonde a été créée dans un environnement de développement fourni par le site officiel de Contiki, la machine virtuelle InstantContiki3.0, en utilisant principalement pour les tests l'outil de simulation Cooja.
Cooja est un simulateur de matériel pour Contiki permettant de créer virtuellement un réseau de capteurs, de les positionner à notre envie et de charger les différents programmes pour les noeuds (ou motes dans le jargon Contiki) à la volée. Nous avons donc passé beaucoup de temps à l'utiliser pour tester notre programme dans des situations réalistes.

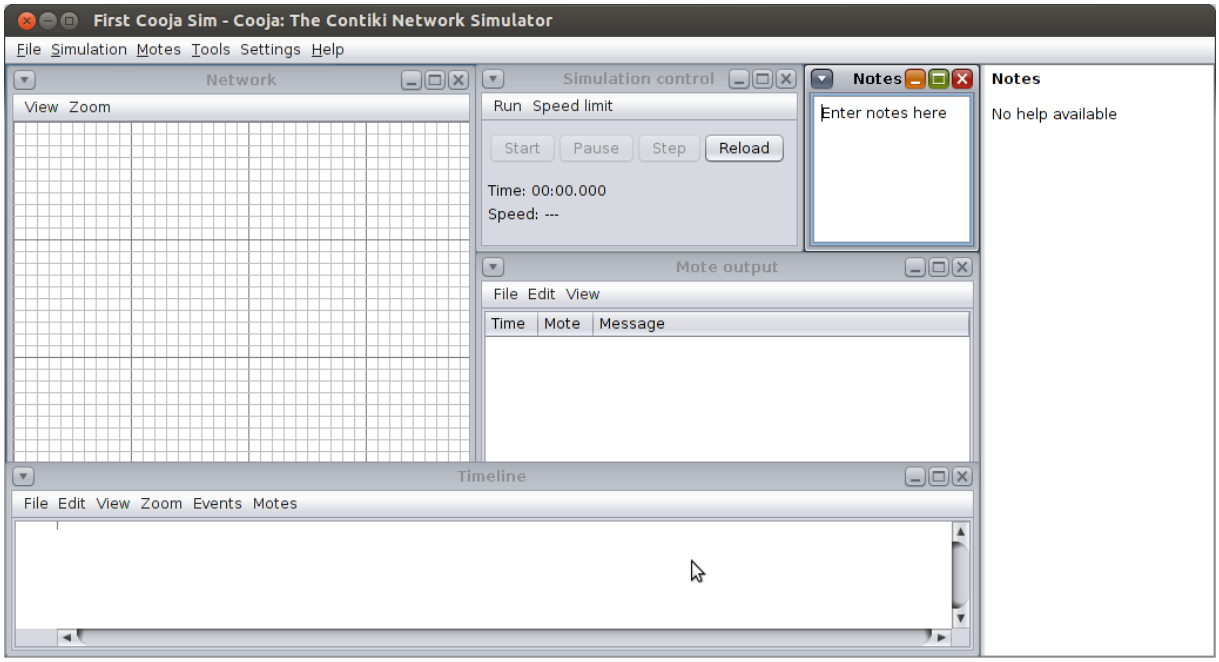

FIGURE 1.2 – Capture d'écran de Cooja.

1.4.3 Langage C embarqué et sa chaîne de compilation

La création de la sonde s'est fait sur Contiki et le programme a dû être adapté aux contraintes du matériel pour lequel il est créé. Pour cela, nous devions rendre le code le plus léger et proche du matériel, ceci s'illustre par l'absence des librairies standard du langage C, par exemple stdlib ou unistd.
La compilation des programmes se fait avec des versions de GCC spécifiques aux architectures matérielles que nous utilisons.

1.4.4 Git

Git est un gestionnaire de version de projet. Celui-ci permet de synchroniser le travail de notre binôme.
Nous avons choisi d'utiliser la plateforme GitHub pour accueillir notre dépôt Git, afin de faciliter l'accès à notre code.

Chapitre 2

Explications techniques

2.1 Contiki

Afin de créer notre sonde, nous avons utilisé Contiki, notamment la Pile réseau IPv6 et les buffers cycliques.

2.1.1 Pile réseau de Contiki

Contiki possède trois piles réseau :
— Rime
— IPv4
— IPv6
Bien évidemment, celle qui nous intéresse ici est la pile IPv6.

Organisation de la pile IPv6

L'implémentation de la pile réseau nous permet d'utiliser les communications avec aisance. C'est grâce à elle que nous pouvons envoyer et recevoir des messages.

La pile réseau se découpe en quatre couches :
— Couche réseau
— Couche MAC – Medium Access Control
— Couche RDC – Radio Duty Cycling
— Couche radio

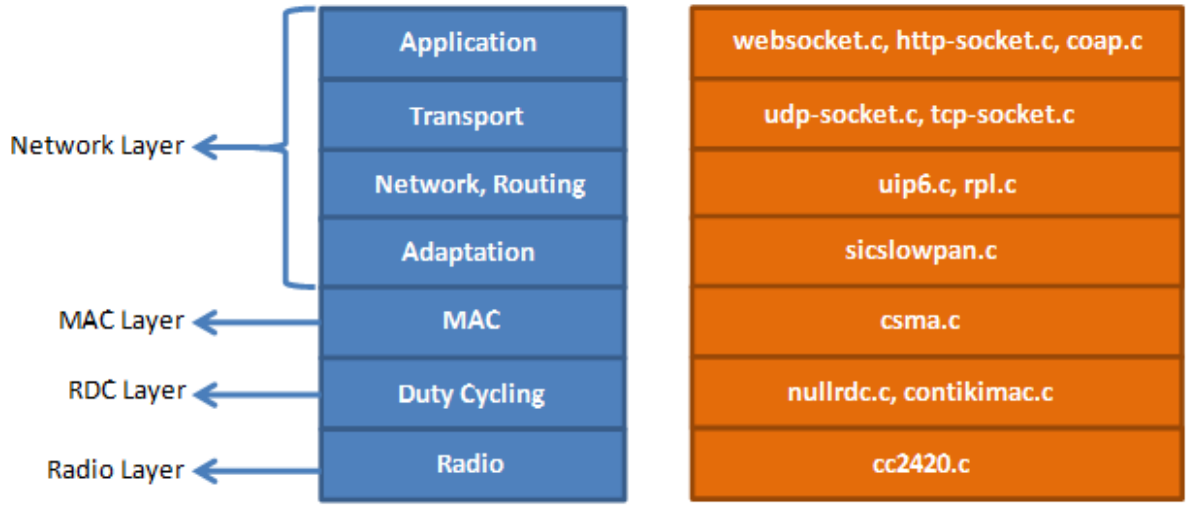

FIGURE 2.1 – Organisation de la pile réseau de Contiki.

Packetbuf

Pour envoyer et recevoir des paquets, Contiki se base sur un buffer unique de paquets – paquetbuf, qu'ils soient entrants ou sortants. Le buffer est découpé en deux parties, la première pour l'en-tête, et la deuxième pour les données.

FIGURE 2.2 – Découpage du buffer de paquets de Contiki.

Une différence est faite entre les paquets sortants et les paquets entrants.

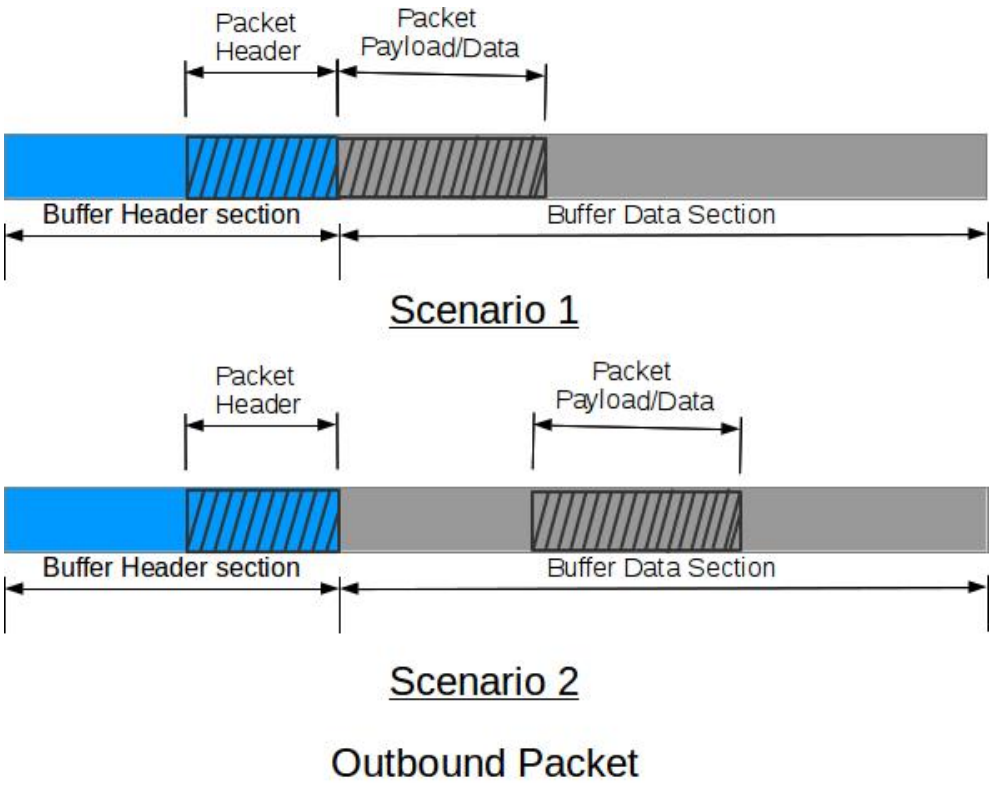

FIGURE 2.3 – Répartition des données pour un paquet sortant.

Le buffer pour les paquets sortants permet de construire ce qu'on veut envoyer de façon structurée, avec une séparation de l'en-tête et des données qui permet de modifier l'un sans influer sur l'autre.
Ce découpage donne accès aux attributs des paquets, ce qui comprend 4 niveaux d'utilisations :

— Local
— Entre deux voisins
— Entre les noeuds en bout de la communication
— L'expéditeur et le receveur locaux, et l'expéditeur et le receveur finaux

Ces attributs sont toutes les informations utiles à la diffusion et au routage du paquet, par exemple le canal, le numéro de séquence, le nombre de sauts, et bien d'autres, 26 au total.

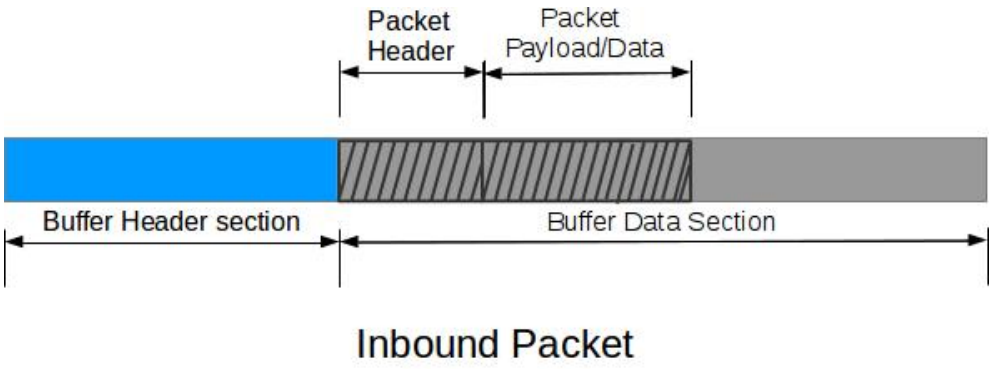

FIGURE 2.4 – Répartition des données pour un paquet entrant.

Le buffer pour les paquets entrant ne fait pas la différence entre en-tête et données, et met tout dans la partie données. Cela permet d'économiser de la puissance de calcul en évitant d'interpréter l'en-tête.
Les fonctions à disposition permettent uniquement de récupérer le paquet brut. Pour effectuer nos vérifications, ils nous faut donc étudier la construction des paquets.

2.1.2 Systèmes de stockage Contiki

Pour nos vérifications superficielles, il nous a été demandé de stocker temporairement quelques informations à propos des paquets du trafic. Nous avons considéré plusieurs options pour stocker ces informations.

Contiki Coffee file system

Contiki possède plusieurs systèmes de fichiers qui implémentent l'interface de Contiki File System, dont Coffee. Coffee est utilisé sur les appareils équipés avec de la mémoire flash ou de l'EEPROM. Contiki s'occupe de l'implémentation matérielle, et Coffee fournit une API qui est similaire aux opérations sur les fichiers du langage C standard.

L'intérêt d'un système de fichier est d'envoyer d'un seul coup plusieurs données, afin de limiter le nombre de transmissions, et donc de consommer moins d'énergie. Aussi, garder les données en mémoire de façon locale permet, en cas de transmission échouée, de ne pas les perdre définitivement.

Nous avons préféré nous tourner vers la mémoire volatile pour enregistrer les informations, pour avoir une solution moins lourde.

Volatile

La mémoire volatile de Contiki peut s'utiliser de plusieurs façons :

Listes

La librairie de Contiki pour les listes procure une liste chaînée dans laquelle les informations souhaitées sont stockées puis récupérées par la suite. La librairie est utilisée partout dans Contiki pour stocker les listes des processus, les files de paquets, les listes des voisins, et d'autres tables.
Les objets de la liste sont des structures qui sont définies par le modules qui utilise la liste.

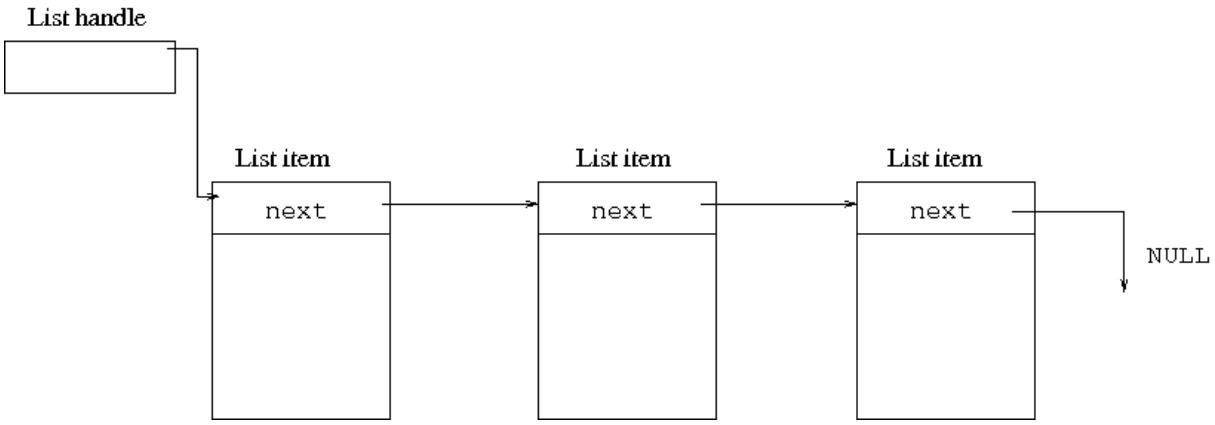

FIGURE 2.5 – Architecture d'une liste chaînée sur Contiki.

La seule condition est que le premier objet de la liste doit être un pointeur, qui est utilisé par la librairie pour lier les objets ensemble dans la liste.
Une liste de Contiki consiste d'un pointeur et de zéro ou plus objets dans la liste, comme montré dans l'illustration ci-dessus. Le pointeur pointe vers le premier objet de la liste.

La liste était l'une des solutions potentielles à notre besoin de stockage, mais il a été décidé de faire appel à un moyen encore plus léger : le buffer cyclique.

Buffers cycliques

Il y a plusieurs endroits dans Contiki où un gestionnaire d'interruption doit garder des données qui peuvent être indépendamment être lues par d'autres parties de Contiki. Le gestionnaire et les fonctions de lectures doivent être synchronisées pour éviter les problèmes de concurrence, parce le gestionnaire peut préempter les parties qui lisent les données à n'importe quel moment. Les mécanismes de lecture et d'écriture sont donc pensées pour être indépendantes de chacune.

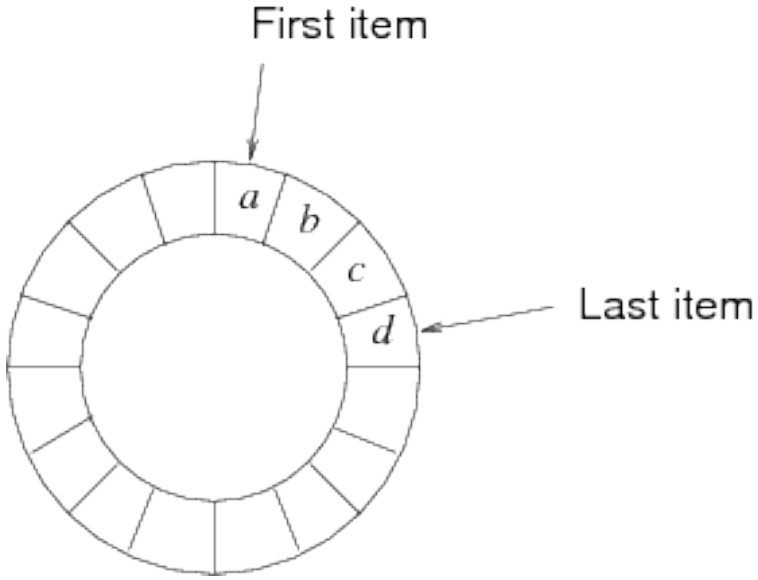

FIGURE 2.6 – Architecture d'un buffer cyclique sur Contiki.

Ci-dessus : Un buffer cyclique avec quatre objets, a, b, c et d. L'objet a est le prochain a sortir du buffer. Le d est l'objet le plus récemment inséré. Le premier et dernier pointeur permettent d'y accéder respectivement.

Un buffer cyclique (parfois appelé buffer en anneau) est une structure de données qui garde les données en mémoire sous la forme d'un anneau comme montré ce-dessus. Les données sont stockées dans le buffer, et sont lues dans le même ordre que celui dans lequel elles ont été insérées. La structure est donc proche d'une FIFO (First In First Out) ou file en français, mais avec la particularité d'être cyclique.
Contiki propose une librairie qui implémente les buffers cycliques, qui garde en mémoire des octets dans un tableau dont la taille doit être une puissance de deux, inférieure ou égale à 256. La raison pour la contrainte sur les puissances de deux est que la structure implémente les fonctions sur les pointeurs de début et de fin avec des opérateurs booléens. Aussi, la taille maximale de 256 octets est là pour permettre à ces pointeurs de début et de fin de tenir dans une structure de 8 bits. L'octet habituel est la seule structure de

donnée qui peut être mise à jour de façon atomique, ce qui est important pour assurer la synchronisation entre les lectures et écritures sur le buffer cyclique, même si l'un préempte l'autre.

Malgré les contraintes imposées par cette structure, sa légèreté et ses propriétés de synchronisation sont des gros avantages sur les autres structures : Les paquets n'attendent pas que Contiki ait fini d'écrire dans sa mémoire.

2.2 6LoWPAN

2.2.1 Compression des headers

Comme évoqué dans l'analyse de l'existant, les communications via IPv6 se révèlent peu pratiques pour des systèmes contraints, surtout à cause des headers (ou entêtes) trop volumineux qui réduisent la charge utile de données contenues par un paquet, multipliant le nombre de paquets.
La RFC 4944 définit LOWPAN_HC1, le mécanisme de compression des en-têtes IPv6 pour les LowPAN. Elle intègre aussi la compression de l'en-tête UDP sur 4 octets, mais n'autorise pas la compression du Checksum. De plus, elle restreint la plage des ports UDP de 61616 à 61631 afin de compresser à 4 bits cette valeur.

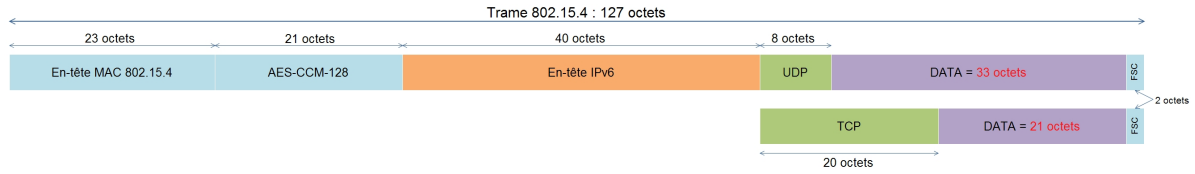

FIGURE 2.7 – Trame IPv6 non compressée.

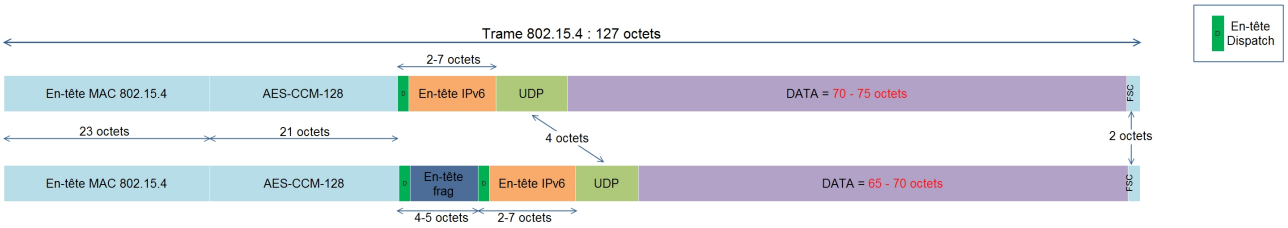

FIGURE 2.8 – Trame IPv6 compressée.

On remarque que la charge utile est doublée grâce à la compression d'en-tête.

2.2.2 Attaques possibles

Notre but étant de travailler vers la sécurisation d'un réseau 6LoWPAN, nous avons étudié différentes attaques possibles sur ce type de réseau. Elles se classent en deux catégories : Celles qui sont pertinentes pour notre sujet, et celles qui ne le sont pas.

Non-pertinentes

Les attaques passives telles que les écoutes, les attaques de brouillage des ondes, ou d'inondation de paquets sont difficilement repérables par un noeud du réseau. Ces attaques ne sont donc pas pertinentes dans le cadre de notre sujet, car il faudrait fait usage de matériel différent de nos capteurs.

Pertinentes

D'autres attaques, comme la duplication ou falsification de paquets, le spoofing et l'attaque Sybil sont des exemples que nous avons étudié pour préparer des vérifications sur notre capteur.

La **falsification** de paquets peut entrainer une nombre de situations d'attaque, tout en étant difficile à détecter. Un paquet peut être falsifié sur plusieurs champs ou un seul, pour avoir différents effets.
Le **spoofing** est la situation où un attaquant envoie un paquet en falsifiant l'adresse source pour se faire passer pour quelqu'un d'autre, généralement une adresse de confiance.
La **duplication** de paquets est une autre manière de mener une attaque contre un réseau, par exemple, en renvoyant un message d'authentification antérieur.
L'**attaque Sybil** est particulière, il s'agit d'un seul acteur possédant plusieurs identités, par exemple un seul noeud qui envoie des messages avec deux ou plus adresses sources qui n'étaient pas utilisées auparavant. C'est ce qui la différencie d'un simple spoof.

Chapitre 3

Déroulement du projet

3.1 Prise en main du sujet et des technologies

Après avoir développé la partie sur les explications techniques, nous allons détailler le déroulement du projet pour notre binôme. Pour commencer, nous devions prendre en main les différents outils nécessaires au projet.

3.1.1 Contiki

Comme nous avons déjà vu, Contiki est un système d'exploitation embarqué. Notre expérience avec le domaine de l'embarqué était limitée malgré notre curiosité, bien que Théo ait fait un stage en rapport pendant sa licence. Nous avons étudié les systèmes d'exploitation lors de cette année de master, mais pas les systèmes embarqués en particulier. L'interaction avec Contiki s'est fait assez rapidement, mais sa compréhension réelle s'est avérée cependant difficile.
Certains concepts, comme les proto-threads (parallélisation très légère avec une taille mémoire limitée) utilisés par Contiki, sont assez proches d'autres concepts présents dans les systèmes d'exploitation habituels, néanmoins la découverte des autres fonctionnalités de Contiki s'est fait au fur et à mesure. En effet Contiki possède de nombreuses caractéristiques en plus de celles qu'offrent les systèmes embarqués habituels.
Ces notions, en plus de celles déjà évoquées sur IPv6 et 6LoWPAN, nous confortent dans le choix de Contiki plutôt que FreeRTOS ou TinyOS, qui regroupe l'ensemble des caractéristiques des autres OS.

3.1.2 Chaîne de compilation et pilotes

L'utilisation des chaines de compilations s'est faite rapidement puisque la machine virtuelle InstantContiki3.0 en possède déjà une pour les projets existants et leur différente architecture supportée par Contiki. Les différents tutoriels présents sur le web ont facilités l'installation manuelle de ces chaînes de compilation.
En revanche, les pilotes des différents contrôleurs radios nous ont semblé difficiles à prendre en main. Nous avons également choisi de nous concentrer sur les contrôleurs cc2420 (utilisés pour les communications sans-fils) de Texas Instrument puisqu'ils sont déjà présents dans les autres projets.

3.2 Programme développé

Le programme développé a été une mise en pratique de nos recherches, mais comme évoqué avant, la prise en main des technologies utilisées a été plutôt longue.

3.2.1 Fonctionnement du projet

La sonde renifleuse capture les paquets dans le trafic du réseau, en les récupérant à la couche radio de la pile réseau, donc récupère tous les paquets, y compris ceux qui ne sont pas conformés à 6LoWPAN.
Les paquets sont triés et stockés dans les buffers cycliques pour permettre des vérifications plus en profondeur.
Le triage se fait en interprétant l'en-tête bit par bit, en se basant sur les explications des RFC pour déterminer les différentes caractéristiques du paquet.

Bit Pattern	Short Code	Description
00 xxxxxx	NALP	Not A LoWPAN Packet
01 000001	IPv6	uncompressed IPv6 addresses
01 000010	LOWPAN_HC1	HC1 Compressed IPv6 header
01 010000	LOWPAN_BC0	BC0 Broadcast header
01 111111	ESC	Additional Dispatch octet follows
10 xxxxxx	MESH	Mesh routing header
11 000xxx	FRAG1	Fragmentation header (first)
11 100xxx	FRAGN	Fragmentation header (subsequent)

FIGURE 3.1 – Exemple de tableau aidant à l'interprétation des paquets.

3.2.2 État du projet

Ce projet a évolué au fur et à mesure de nos avancées, et bien que nous ayons défini des vérifications, nous n'avons pas pu implémenter tout ce que nous voulions.
Le triage des paquets est encore à gros grains, et l'utilisation des buffers cycliques ne marche pas complètement. Certains bugs à la compilation nous empêche de les utiliser comme nous le voulons.
Nous avions prévu d'implémenter des vérifications sur les tailles des paquets, la taille annoncée par rapport à la taille réelle, mais l'interprétation des en-têtes a été prioritaire car c'est elle qui nous permet d'accéder au champ en question.

3.3 Retours d'expérience

Durant notre projet, nous avons eu l'opportunité d'améliorer nos compétences dans les différents domaines du web des objets. Ce travail pourra bénéficier d'évolutions sur le court comme sur le long terme.

3.3.1 Évolutions à court terme

Sur le court terme, il serait possible d'ajouter d'autres types de détections d'attaques en fonction des besoins, mais ces ajouts ne sont pas pertinent dans le cadre de notre projet. En effet l'équipe de recherche 2XS souhaite intégrer la sonde dans Discus qui est déjà un système de détection d'intrusion, ce qui rendrait ces modifications redondantes.
La sonde pourrait donc fournir les informations dont Discus a besoin pour faire respecter les contraintes énoncée dans le script adéquat.
De ce fait, les vérifications d'attaques seraient donc effectuées par le système qui reçoit les informations, et non plus par les sondes directement. Ceci permet d'alléger la charge de travail sur les capteurs aux capacités restreintes. Aussi, les nouvelles vérifications d'intrusion pourront êtres implémentées sans reprogrammer les sondes.

3.3.2 Évolutions à long terme

Sur un plus long terme, il a été pensé d'éventuellement faire communiquer les sondes entre elles afin de créer une grille de capteurs. Cela permettrait, grâce aux différents RSSI associés à un seul paquet capté, de localiser les différents acteurs du réseau, et donc de localiser l'attaquant lors d'une anomalie.

3.3.3 Challenges

Ce projet a été enrichissant et concernait les domaines qui nous intéressent (comme l'embarqué et/ou la sécurité), mais nous avons dû faire face à plusieurs difficultés et challenges lors de son déroulement. Bien que ces contraintes ralentissaient l'avancement du projet, elles furent instructives à plusieurs niveaux.
Tout d'abord, le sujet se concentre sur des domaines et technologies dont nous n'étions pas très familiers. Le monde de l'informatique embarquée est fait de contraintes auxquelles il faut s'adapter pour être productif, les retours beaucoup moins verbeux lors d'erreurs, les limites de mémoire, de puissance, et parfois l'absence de librairies pour rendre le code assez léger pour la plateforme sont quelques exemples de difficultés lorsque l'on découvre l'embarqué. Pouvoir s'adapter peut paraître laborieux mais la solution est de bien organiser le contexte et prendre le temps nécessaire pour assimiler les bases.
Les découvertes étaient nettement plus nombreuses dans les technologies employées, notamment au niveau des systèmes d'exploitation embarqués, comme Contiki, et leurs technologies de communication. Nous avons lu beaucoup de spécifications, de RFC et de documentation et en rétrospective, nous aurions eu plus de facilité à établir un plan d'approche, organiser nos découvertes pour éviter la confusion. Par exemple, prendre du temps pour bien se renseigner sur Contiki, puis lorsque l'outil est maîtrisé, se renseigner sur 6LoWPAN, et continuer à procéder par étapes durant l'ensemble du pji.
Malgré nos recherches en profondeur, nous sommes parfois tombés sur des incohérences dans la documentation, ou des explications confuses, notamment sur le buffer de paquets qui n'a pas la même structure si les paquets sont entrants ou sortants. Pour pallier à ce souci, nous nous sommes documentés sur des sites et des encyclopédies en ligne (wikis) universitaires traitant de Contiki avec des tutoriels adaptés à notre besoin.
Une autre difficulté durant le projet venait de Contiki lui même. En effet, certains exemples de code déjà présents ne sont pas assez commentés, ceci peut être problématique pour assimiler le fonctionnement d'un programme. C'est pourquoi nous nous sommes rendus sur des forums traitant de ces sujets et avons sollicité des membres de l'équipe 2XS.

Conclusion et perspectives

Notre sujet de projet était de contribuer à un système de détection d'intrusion pour l'Internet des Objets. La principale difficulté de ce sujet était les nombreuses attaques possibles sur le réseau, notamment le détournement de routage pour l'écoute, le vol d'identité et la falsification de paquets. En partant de cette problématique, nous avons produit une sonde, un noeud sur le réseau qui écoute le trafic radio, afin de construire par dessus plusieurs couches de détections.

De ce fait, nous procurons à l'équipe de recherche un outil de base pour développer la sécurité réseau dans l'IDS de l'équipe – Discus, en rajoutant la plateforme radio aux autres déjà existantes.

En travaillant sur ce projet, nous avons pu découvrir des technologies que nous n'avons pas l'habitude de voir lors de nos cours. Cela nous a permis d'élargir nos horizons vis à vis des domaines de l'informatique, de tester de nouveaux paradigmes.
Nous avons pu nous améliorer sur notre travail en autonomie, nous entraîner à la recherche d'information, de documentation. Cet entraînement est aussi un regard vers ce qu'est le monde de la recherche, qui a permis de le découvrir ou le redécouvrir.