

UNIVERSITÉ DE LAUSANNE

PROJET DE MASTER  
2017-2018

# Internet des Objets : Traces, Vulnérabilités et Défis Forensiques

---

*Internet of Things : Traces,  
Vulnerabilities and Forensic  
Challenges*

*Francesco Servida*

Professeur responsable  
Prof. Eoghan CASEY

En collaboration avec SecuLabs SA :  
Supervisé par Thibault SOUBIRAN

Expert  
Thomas SOUVIGNET

*Unil*  
UNIL | Université de Lausanne  
Faculté de droit,  
des sciences criminelles  
et d'administration publique

**SecuLabs**  
IT SECURITY SERVICES 





Don't Panic.

---



## Résumé

Dans les dernières années, les dispositifs IdO se sont répandus à un rythme croissant. Le niveau de sécurité généralement faible en fait une cible de choix pour les criminels, qui peuvent les exploiter pour commettre des crimes dans le cyberspace ou pour faciliter d'autres infractions. Ces dispositifs collectent aussi une quantité importante de données sur les utilisateurs et les activités dans l'environnement. Ces données pourraient être exploitées dans le cadre d'une investigation. Avec l'étude d'une sélection de dispositifs ce travail veut dévoiler quels types de traces sont générées par les dispositifs IdO où elles sont stockées et comment les obtenir. Dans ce cadre, une méthodologie pour l'étude des traces présentes sur de nouveaux dispositifs IdO est proposée. Finalement, un scénario pour le challenge forensique IdO du DFRWS est développé.

## Mots Clés

IdO, Vulnérabilités, Analyse Forensique, Botnets, Smartphones, Méthodologie d'analyse.

## Abstract

In recent years, IoT devices have spread at an increasing rate. The generally low level of security makes them a prime target for criminals, who can exploit them to commit crimes in cyberspace or to facilitate other crimes. These devices also collect a significant amount of data about users and activities in the environment. This data could be exploited as part of an investigation. With the study of a selection of devices this work wants to reveal what types of traces are generated by IoT devices, where they are stored and how to obtain them. In this context, a methodology for the study of the traces present on new IoT devices is proposed. Finally, a scenario for the DFRWS IoT forensic challenge is developed.

## Mots Clés

IoT, Vulnerabilities, Forensic Analysis, Botnets, Smartphones, Analysis Methodology.

---

## Sommario

Negli ultimi anni la popolarità dei dispositivi IoT è cresciuta rapidamente. Il livello di sicurezza normalmente basso di questi dispositivi li rende un obiettivo di prima scelta per i criminali, che possono quindi sfruttarli per commettere crimini nel cyberspazio o per facilitare altre infrazioni. Inoltre questi dispositivi raccolgono una quantità estremamente elevata di informazioni sugli utilizzatori e le attività nei loro dintorni. Questi dati potrebbero quindi essere utilizzati nel contesto di un'inchiesta. Tramite lo studio di una selezione di dispositivi questa ricerca vuole svelare i tipi di tracce generate da questi apparecchi, dove sono conservate e come estrarle. A tal fine il lavoro propone una metodologia per l'analisi delle tracce presenti sui nuovi dispositivi a cui un'investigazione potrebbe essere confrontata. In ultima istanza è stato sviluppato uno scenario per la competizione forense IoT del DFRWS.

## Parole Chiave

IoT, Vulnerabilità, Analisi Forense, Botnets, Smartphones, Metodologia d'analisi

# **Remerciements**

Je tiens à remercier le Prof. Eoghan Casey d'avoir accepté de superviser ce projet ainsi que le support, l'expertise et les retours fournis tout le long du travail.

Je remercie aussi vivement Seculabs SA et en particulier Thibault Soubiran pour le soutien apporté lors de l'analyse physique et des vulnérabilités des dispositifs.

Je suis également reconnaissant à l'Association des Diplômés en Sciences Criminelles pour le support important pour ma participation dans le cadre de ce travail au DFRWS EU 2018.

De même, je sais gré à Timothy Bollé, Xavier Burri et Adrien Vincart d'avoir pris part à la relecture de mon rapport.

Mes remerciements vont aussi à tous mes amis et à ma famille qui m'ont soutenu pendant ce travail ainsi que pendant toutes les années le précédent.



# Table des matières

<b>Résumé</b>	<b>I</b>
<b>Remerciements</b>	<b>III</b>
<b>Acronymes</b>	<b>XI</b>
<b>Résumé des Accomplissements</b>	<b>1</b>
<b>1 Introduction</b>	<b>3</b>
<b>2 Revue bibliographique</b>	<b>5</b>
<b>3 Méthodologie</b>	<b>9</b>
3.1 Analyse préliminaire . . . . .	10
3.2 Installation . . . . .	12
3.3 Analyse du réseau . . . . .	14
3.4 Analyse applications officielles . . . . .	14
3.5 Recherche de vulnérabilités . . . . .	15
3.6 Analyse Physique & Accès Root . . . . .	16
<b>4 Résultats</b>	<b>17</b>
4.1 Analyse préliminaire . . . . .	17
4.2 Analyse du Réseau . . . . .	18
4.3 Artefacts Applications pour Smartphone . . . . .	26
4.4 Analyse des Vulnérabilités . . . . .	34
4.5 Accès root et analyse physique . . . . .	37

<b>5 Discussion</b>	<b>45</b>
5.1 Limites & Développements Futurs . . . . .	50
<b>6 Conclusion</b>	<b>53</b>
<b>Bibliographie</b>	<b>55</b>
<b>A Annexes</b>	<b>61</b>
A.1 Analyse Préliminaire des Dispositifs . . . . .	61
A.2 Raspberry Pi . . . . .	61
A.3 QBee - Connexions Réseau . . . . .	62
A.4 Arlo - Connexion locale avec Smartphone . . . . .	64
A.5 QBee - Déchiffrement des Réglages . . . . .	65
A.6 iSmartAlarm - Parsing Base de Données . . . . .	67
A.7 iSmartAlarm - Obtention du Firmware . . . . .	69
A.8 iSmartAlarm - Logs de Diagnostique . . . . .	70
A.9 Arlo - Accès aux données sur le Cloud . . . . .	72
A.10 iOS - Traces sur iOS 11 . . . . .	73
A.11 Rapports de Vulnérabilités . . . . .	74
A.12 Nest - USB Analysis . . . . .	74
A.13 Arlo - Physical Analysis . . . . .	74
A.14 iSmartAlarm - Extraction Physique . . . . .	75
A.15 DFRWS . . . . .	75

# Table des figures

2.1	Modèle de Référence pour l’IdO . . . . .	5
3.1	Position des dispositifs dans le local . . . . .	11
3.2	Topologie logique du réseau . . . . .	12
4.1	Visualisation du trafic réseau sur Kibana . . . . .	20
4.2	Relations entre dispositifs . . . . .	21
4.3	Frame du vidéo extrait de frames_database . . . . .	31
4.4	JSON contenant les informations sur l’utilisateur pour le Arlo . . . . .	32
4.5	Wink - Détection de Fumée . . . . .	34
4.6	Intérieur de la station de base du Arlo et du CubeOne de iSmartAlarm . . . . .	37
4.7	Arlo - Collecte mémoire vers serveur TFTP . . . . .	39
4.8	CubeOne - Analyse avec binwalk de ismart_80.img . . . . .	41
5.1	Application de la méthodologie proposée dans l’investigation numérique des IdO	50
A.1	Raspberry Pi - Page d’administration (Pi-Pineapple) . . . . .	62
A.2	Topologie physique du réseau . . . . .	63
A.3	Extrait des connexions d’une session "Keep-Alive" de la caméra QBee . . . . .	64
A.4	QBee - Tentatives d’ouverture d’une porte dans le pare-feu (Logs de Linux IGD)	64
A.5	Arlo - Etablissement de la connexion en locale . . . . .	65
A.6	Comparaison des fonctions de déchiffrement . . . . .	66
A.7	Fonction de dérivation de la clé AES . . . . .	67
A.8	QBee - Artefacts personnalisés dans Autopsy . . . . .	67
A.9	iSmartAlarm - Comparaison horodatages en fonction du logiciel . . . . .	68
A.10	iSmartAlarm - IPUDairy . . . . .	69
A.11	iSmartAlarm - Récherche URL du Firmware . . . . .	70

---

A.12 iSmartAlarm - Extrait connexion pour les logs de diagnostique . . . . .	71
A.13 iSmartAlarm - Fermeture de la porte dans les logs de diagnostic . . . . .	72
A.14 Arlo - Téléchargement Videos Dernière Semaine . . . . .	73
A.15 QBee Multi-Sensor Camera - Stockage du Mot de Passe . . . . .	83
A.16 QBee Multi-Sensor Camera - Trafic en clair . . . . .	84
A.17 iSmartAlarm - Diagnostique CubeOne . . . . .	85
A.18 iSmartAlarm - Stockage du Mot de Passe . . . . .	86
A.19 Nest Camera - Connexion USB . . . . .	87
A.20 Nest Protect - Connexion USB . . . . .	87
A.21 Arlo - Boot Log . . . . .	88
A.22 Arlo - Extrait Réglages NVRAM . . . . .	88
A.23 iSmartAlarm - Commande <i>md.b</i> . . . . .	88

# Liste des tableaux

3.1	Méthodologie de travail . . . . .	9
3.2	Dispositifs étudiés . . . . .	10
4.1	Résumé des résultats principaux . . . . .	17
4.2	Ports ouverts - Amazon Echo . . . . .	22
4.3	Ports ouverts - QBee Multi-Sensor Camera . . . . .	23
4.4	Ports ouverts - iSmart Alarm Cube One . . . . .	23
4.5	Ports ouverts - Nest Protect . . . . .	24
4.6	Ports ouverts - Wink Hub . . . . .	25
4.7	Samsung - Images Extraites . . . . .	26
4.8	Ports ouverts - Station de Base Arlo - Réseau Interne . . . . .	40
4.9	iSmartAlarm - Régions de Memoire (Taille 8Mio) . . . . .	41
A.1	Résumé des chips d'intérêt et portes d'accès physique disponibles identifiés suite à l'analyse préliminaire . . . . .	79
A.2	Endpoint API découverts via analyse réseau sur caméra QBee . . . . .	80
A.3	iSmartAlarm - IDs de la table TB_IPUDairy . . . . .	81
A.4	iSmartAlarm - IDs de la table TB_SensorDairy . . . . .	81
A.5	Fichiers d'intérêt sur iOS 11 . . . . .	82



# Acronymes

**DFRWS** Digital Forensic Research Workshop.

**DoS** Denial of Service.

**dTLS** Datagram Transport Layer Security.

**ESC** Ecole des Sciences Criminelles.

**FCC** Federal Communications Commission.

**FOSS** Free and Open Source Software.

**MITM** Man In The Middle.

**MQTT** Message Queue Telemetry Transport.

**OWASP** Open Web Application Security Project.

**SSDP** Simple Service Discovery Protocol.

**STUN** Session Traversal Utilities for Network Address Translators.

**UNIL** Université de Lausanne.

**UPnP** Universal Plug and Play.



# Résumé des Accomplissements

Ce travail, au travers de l’analyse d’une sélection de dispositifs IdO, veut étudier les types de traces d’intérêt forensique générées et les failles de sécurité de ces dispositifs. Les achèvements principaux sont exposés ci-après.

- Des plug-ins pour la plateforme d’analyse Autopsy ont été développés, permettant l’extraction automatisée de traces des applications associées aux dispositifs IdO (iSmartAlarm, QBee & Swisscom Home App et partiellement Arlo).
- Les identifiants des comptes liés aux dispositifs IdO de deux applications (QBee & Swisscom Home App) ont pu être déchiffrés à partir de leurs réglages.
- Un outil a été développé permettant l’extraction et le *parsing* sans authentification nécessaire des logs de diagnostic de la station de base de iSmartAlarm.
- Un outil a été développé afin d’intercepter les communications en clair entre la caméra QBee et l’application pour smartphone et réutiliser celles-ci pour désactiver la caméra.
- Le processus de démarrage de trois dispositifs, les stations de base de Arlo et iSmartAlarm ainsi que le WinkHub, a pu être interrompu, et ainsi extraire des images de mémoire.
- Une console administrateur a été obtenue dans le cas du Arlo et du WinkHub, permettant l’extraction de données au niveau du système de fichiers.
- Quatre vulnérabilités ont été découvertes dans les dispositifs de iSmartAlarm et QBee ainsi que les applications liées ; les respectifs producteurs (iSmartAlarm, QBee/Askey, Vestiacom et Swisscom) ont été contactés.
- Un scénario pour le challenge forensique IdO du Digital Forensic Research Workshop (DFRWS) a été développé et sera utilisé pour la compétition 2018-2019.



# 1 Introduction

Dans les dernières années, en particulier à partir du 2009, on a pu assister à une croissance élevée de l'intérêt envers le domaine de l'Internet des Objets (IdO)<sup>1</sup>. Ce terme, introduit en 1999 par Kevin Ashton (Ashton, 2009), représente un réseau interconnecté d'objets et de logiciels qui interagissent entre eux et participent activement aux activités commerciales et sociales (Cluster of European Research Project on the Internet of Things [CERP-IoT], 2010, p. 43).

Les dispositifs qui composent l'Internet des Objets font partie d'une variété de domaines. Il s'agit de dispositifs dédiés à la gestion industrielle automatisée, aux "smart grids" ou encore vers le marché privé, dans la domotique ou surveillance. Les dispositifs appartenant à cette dernière catégorie sont de plus en plus répandus et médiatisés ; l'intérêt toujours plus élevé a poussé un nombre toujours plus élevé de fabricants à proposer des dispositifs faisant partie de cette gamme. Cependant, ces dispositifs posent des questions de sécurité. La nécessité d'une mise en place facilitée et la puissance de calcul limitée ont comme conséquence une réduction du niveau de sécurité qu'ils offrent (Barnard-Wills, Marinos & Portesi, 2014 ; Kolias, Kambourakis, Stavrou & Voas, 2017). Le botnet Mirai, utilisé pour conduire de massives attaques DDoS en 2016 (Holub & Colford, 2017), n'est qu'un exemple du faible niveau de sécurité de ces dispositifs. Les failles de sécurité de ces dispositifs peuvent être exploitées pour les intégrer à des Botnets, pour les utiliser comme têtes de pont pour des réseaux plus sécurisés (Cisco, p. d.) ou encore pour désactiver ces dispositifs s'il s'agit par exemple de systèmes d'alarme.

Ces dispositifs domotiques collectent aussi une quantité élevée de données sur les activités des utilisateurs (Barnard-Wills et al., 2014). Ces données pourraient être utilisées à des fins forensiques lors de l'investigation d'un crime traditionnel.

Leur présence dans la vie des utilisateurs pose aussi des questions de protections de la vie privée et de protection de leur abus à des fins de harcèlements. Dans ces cas, qui sont récemment devenus problématiques (Loung, 2018), l'analyse des dispositifs IdO peut aussi se révéler utile.

---

1. Internet of Things (IoT)

**Objectifs et Questions de Recherche** C'est dans ce contexte que ce travail se situe, dans le but d'évaluer les traces collectées par ces dispositifs, leur extraction, la sécurité des dispositifs et les possibilités d'exploitations des failles de sécurité.

Pour cela, les questions de recherche suivantes ont été formulées :

- Quelles traces sont disponibles sur les dispositifs IdO ?
- Quelles traces sont générées par leurs applications mobiles ?
- Quelles informations ces traces peuvent fournir à un enquêteur lors de l'investigation d'un crime ?
- Quelles vulnérabilités sont présentes dans ces dispositifs ?
- Leur connaissance peut-elle être utile à des fins forensiques ?

## 2 Revue bibliographique

**Dispositifs IdO** L'Internet des Objets se réfère à un réseau interconnecté d'objets et applications interagissant entre eux et participant à des activités commerciales et sociales (CERP-IoT, 2010, p. 43). Selon l'Internet Business Solutions Group de Cisco, l'Internet des Objets "représente le moment temporel auquel le nombre de dispositifs connectés à Internet dépasse celui des personnes", et estiment que cela a été atteint entre 2008 et 2009 Evans (2011)

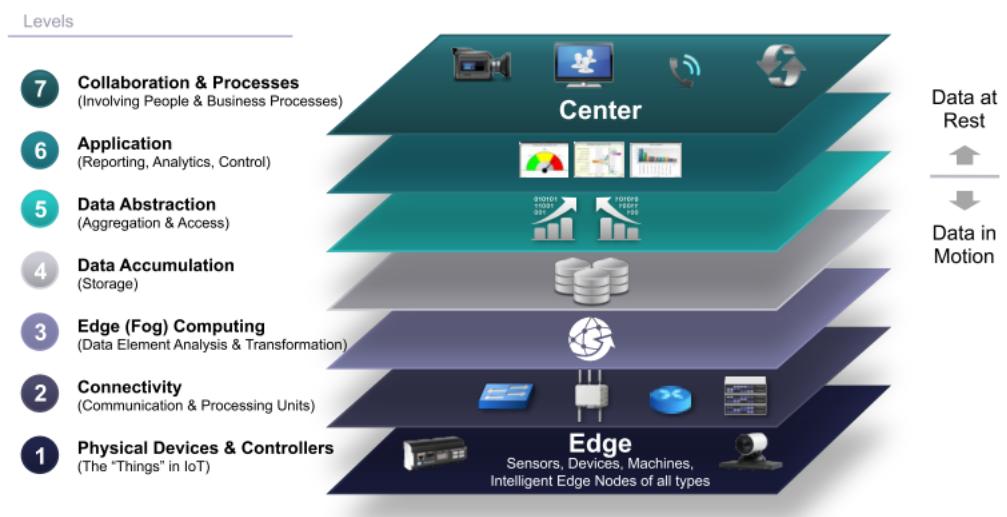


FIGURE 2.1 – Modèle de Référence pour l'IdO (Cisco, 2014b)

Cisco (2014b) propose un modèle pour l'Internet des Objets divisé en sept différents niveaux, des dispositifs physiques jusqu'au niveau du cloud et des applications qui y sont connectées. Une distinction notable se fait entre *data at motion* et *data at rest*. La différence se fait ainsi entre la partie inférieure du modèle qui comprend les dispositifs et les applications qui génèrent et/ou traitent les données, mais qui n'en stockent qu'une petite quantité, et la partie supérieure du modèle qui correspond aux applications qui stockent et consomment ces données. Dans le cadre des dispositifs IdO domestiques, les dispositifs appartiennent aux premiers niveaux du modèle, tandis que les services cloud et les applications pour smartphones se situent dans les quatre derniers niveaux.

**Analyse Forensique** L’analyse forensique de ces dispositifs pose des contraintes particulières par rapport à la forensique numérique traditionnelle ; la quantité de données générées, l’hétérogénéité des protocoles utilisés, le nombre de dispositifs et la nature distribuée des dispositifs et des traces en sont des exemples (Oriwoh, Jazani, Epiphaniou & Sant, 2013 ; Zawoad & Hasan, 2015).

Selon (Oriwoh et al., 2013 ; Zawoad & Hasan, 2015) le modèle traditionnel de l’analyse forensique numérique peut être représenté par quatre étapes : identification, collecte, organisation et présentation. Un modèle générique pour l’analyse forensique est aussi proposé par Pollitt, Casey, Jaquet-Chiffelle et Gladyshev (2018), qui divisent le processus forensique en *Survey, Preservation, Examination, Documentation, Analysis, Integration et Interpretation*.

Oriwoh et al. (2013) estiment l’étape d’identification comme étant la plus importante pour l’analyse forensique dans le contexte de l’IdO. Comme ces dispositifs génèrent une quantité de traces beaucoup plus élevée que les dispositifs numériques traditionnels, il est primordial de savoir quels types de traces sont présents et où les trouver, sous peine de perdre une quantité de temps non négligeable pour trier les données non pertinentes.

Un autre modèle est proposé par Oriwoh et al. (2013), qui divise le champ d’investigation en trois zones distinctes pour la recherche de traces : Réseau Interne, Intermédiaire et Réseau Externe. Un investigateur pourra ensuite se concentrer sur les dispositifs faisant partie de la zone d’intérêt.

Selon Zawoad et Hasan (2015), l’analyse forensique de l’IdO se divise en trois niveaux : *Device Level Forensics, Network Forensics et Cloud Forensics*. Cette division en trois couches est très similaire à la proposition des trois zones de Oriwoh et al..

Kebande et Ray (2016) proposent un cadre de travail pour l’analyse forensique des dispositifs IdO, divisant celle-ci en un processus proactif, une analyse forensique et une approche réactive.

Chung, Park et Lee (2017) proposent une approche automatisée pour l’analyse de l’écosystème de Amazon Alexa basé sur la collecte de données depuis le cloud et l’analyse des données de l’application pour smartphone.

Dans l’article de Akatyev et James (2017), on trouve enfin une analyse des crimes potentiellement assistés par les dispositifs IdO ainsi que les dispositifs sur lesquels un investigateur devrait se concentrer lors d’une investigation selon le type de celle-ci. D’autres types de menaces sont également décrites dans les articles de Barnard-Wills et al. (2014), Dorsemaine, Gaulier, Wary, Kheir et Urien (2016).

Publié en mars 2018 Amar et al. (2018) proposent une approche pour la collecte et l’analyse du trafic de plusieurs dispositifs IdO. Leur travail présenté se limite à l’analyse du trafic de base des dispositifs, sans interaction de l’utilisateur.

Une partie des articles traitant l’analyse forensique des IdO propose des méthodologies d’analyse avec des approches proactives, nécessitant la mise en place préalable d’une collecte de donnée, soit au niveau des dispositifs, soit au niveau du réseau (Kebande & Ray, 2016; Zawoad & Hasan, 2015). Cela n’est pas le cas pour l’analyse forensique de dispositifs IdO domotiques ; la faute à des contraintes de coûts, de performance et de vitesse de développement, ces dispositifs ne sont déjà pas conçus avec la sécurité en premier plan. L’intégration d’une composante forensique, telle qu’une collecte de logs vers un stockage centralisé (Zawoad & Hasan, 2015), augmenterait seulement la complexité et donc le coût de ces dispositifs, ce qui est contraire aux buts des fabricants.

Les méthodologies proposées par ces auteurs s’adaptent mieux à un environnement IdO industriel, où l’on a les connaissances, mais aussi où la nécessité de collecter préventivement logs et trafic réseau est présente afin de faciliter une éventuelle investigation lors d’un incident.

Mis à part l’approche proposée par Chung et al. pour l’analyse forensique de l’environnement de Alexa ou par Amar et al. pour l’analyse réseau, les approches présentées dans la littérature sont de haut niveau, des méthodologies génériques. Ils ne traitent pas comment l’on pourrait concrètement procéder aux analyses dans les différentes étapes des méthodes proposées.

**Analyse de Sécurité** Un modèle d’analyse du niveau de sécurité des dispositifs IdO est proposé par le projet IoT de OWASP<sup>1</sup>, un premier résultat du projet fut en 2014 la liste des Top Ten IoT Vulnerabilities ; Cela a désormais été supplanté par une nouvelle version orientée aux surfaces d’attaque et non aux vulnérabilités, et qui idéalement devrait pouvoir être appliquée de manière très générique aux dispositifs IdO (Open Web Application Security Project [OWASP], 2016, 2018).

Sachidananda et al. (2017) propose une approche automatisée pour l’analyse des vulnérabilités d’un dispositif IdO. Cependant une telle approche se limite principalement à l’analyse de vulnérabilités connues.

---

1. Open Web Application Security Project (OWASP)



## 3 Méthodologie

Afin de répondre aux questions de recherche de ce travail une méthodologie est exposée, qui peut s'adapter à l'analyse forensique d'un dispositif IdO ainsi que de ses vulnérabilités. On envisage que cette méthodologie, présentée dans le tableau 3.1, puisse être généralisée à l'étude de nouveaux dispositifs IdO auxquels une investigation policière pourrait être confrontée.

Dans le cadre de ce travail un scénario pour le *IoT Forensic Challenge* du DFRWS 2018-2019 est développé et présenté à l'annexe A.15.

TABLE 3.1 – Méthodologie de travail

1	Analyse préliminaire
2	Installation
3	Analyse du réseau
4	Analyse des applications officielles
5	Recherche de vulnérabilités et des points d'entrée
6	Analyse physique

Dans cette recherche plusieurs dispositifs connectés ont été utilisés, une partie d'entre eux était déjà à disposition, tandis que d'autres ont dû être achetés. Le choix de ceux à acheter a été basé sur plusieurs facteurs, notamment leur popularité au niveau du marché suisse ainsi que le niveau d'étude déjà existant ; leur popularité a été évaluée sur la base des plus vendus sur les deux principaux marchés en ligne Digitec<sup>1</sup> et Microspot<sup>2</sup>, le niveau d'étude déjà existant a été utilisé afin de cibler des dispositifs potentiellement intéressants, par exemple sur lesquels certaines vulnérabilités étaient déjà connues, mais, en même temps, éviter ceux déjà étudiés de manière étendue dans la littérature, tel que le Amazon Echo (Chung et al., 2017; Hyde & Moran, 2017; Rajewski, 2016, 2017) . Un appareil Amazon Echo a tout de même été intégré

---

1. [www.digitec.ch](http://www.digitec.ch)

2. [www.microspot.ch](http://www.microspot.ch)

dans l'environnement d'étude afin d'étudier les interactions avec les autres dispositifs.

Ce processus de sélection a permis de choisir les dispositifs exposés au tableau 3.2 :

TABLE 3.2 – Dispositifs étudiés

Producteur	Dispositif	Raison	Fonction
Amazon	Echo	A disposition	Assistant personnel
Askey	Qbee Multi-Sensor Camera <sup>3</sup>	Populaire en Suisse	Camera de surveillance multifonction
iSmartAlarm	Cube One & accessories <sup>4</sup>	Populaire en Suisse	Système d'alarme avec station de base, senseurs de contact et de mouvement
Nest	Camera	A disposition	Camera de surveillance
Nest	Protect	A disposition	Détecteur de fumée
Netgear	Arlo Pro	Populaire en Suisse	Camera de surveillance

Ce travail ne voulait pas être une simple analyse en laboratoire de ces dispositifs, pris séparément, mais voulait également prendre en compte les interactions entre les différents dispositifs dans une situation plus proche de la réalité. Pour cela un faux laboratoire de production de stupéfiants a été mis à disposition par l'Ecole des Sciences Criminelles (ESC) ; dans ce laboratoire les dispositifs ont ensuite été installés et configurés. Un croquis représentant l'installation physique des dispositifs est présenté à la figure 3.1.

## 3.1 Analyse préliminaire

Dans cette étape, pour chaque dispositif, on procède à l'établissement d'une collection de documentation existante sur le dispositif. Il s'agit notamment de rechercher toute publication décrivant quels protocoles sont utilisés par le dispositif, quels points d'entrée existent (API, interfaces d'administration ou de diagnostique), ainsi que les vulnérabilités connues du dispositif. Ces informations pourront ensuite être utilisées pour cibler des sources potentielles de données forensiques ou aider dans l'investigation d'un incident.

Des sources possibles pour cette recherche sont notamment des publications forensiques,

3. Aussi commercialisé en Suisse comme "Swisscom QBee Multi-Sensor Camera"

4. iSmartAlarm Home Security System Starter Pack

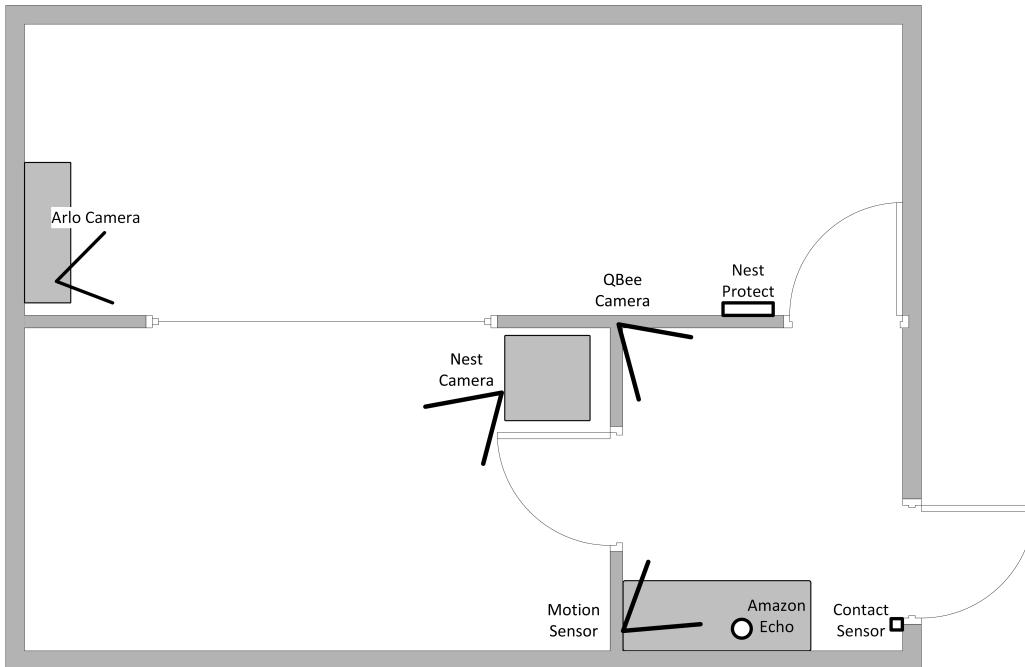


FIGURE 3.1 – Position des dispositifs dans le local

la documentation du fabricant, les documents de certification pour le Federal Communications Commission (FCC) ainsi que les bases de données de vulnérabilités<sup>5</sup>.

**Documentation FCC** La documentation de référence soumise au FCC est de particulière importance en ce qui concerne l’analyse physique des dispositifs. Les photos de l’intérieur des dispositifs permettent de déterminer à l’avance quels chips sont présents et quels ports de diagnostic sont à disposition. La connaissance des différents chips aidera à évaluer quelles informations pourraient être obtenues, mais aussi comment elles pourraient l’être. Par exemple, un dispositif de surveillance vidéo avec une mémoire flash de seulement quelques dizaines de mégaoctets ne contiendra vraisemblablement pas un cache de la vidéo acquise, tandis que si la mémoire est de plusieurs centaines de mégaoctets il pourrait être envisagé de trouver des séquences vidéo en cache. Si l’on remarque que le chip de stockage est du eMMC, on pourrait envisager une acquisition des données par chip-off de manière simplifiée tandis que, si le chip est de type NAND, une acquisition par chip-off pourra se relever plus compliquée.

La connaissance des ports de diagnostic (JTAG, Série (UART) ou USB) donnera aussi une indication sur les possibilités d’accès physique au dispositif. A noter que la présence de ports de diagnostics dans les dispositifs utilisés pour la documentation du FCC n’implique pas forcément que les ports soient facilement accessibles dans les dispositifs analysés. Cela est par

5. <https://cve.mitre.org>, <https://nvd.nist.gov> ou encore <https://www.cvedetails.com>

exemple le cas pour les ports série, qui sont souvent présents, mais sans broches soudées.

Cette partie préliminaire de récolte de données permet ensuite de prioriser les analyses à conduire sur les dispositifs.

## 3.2 Installation

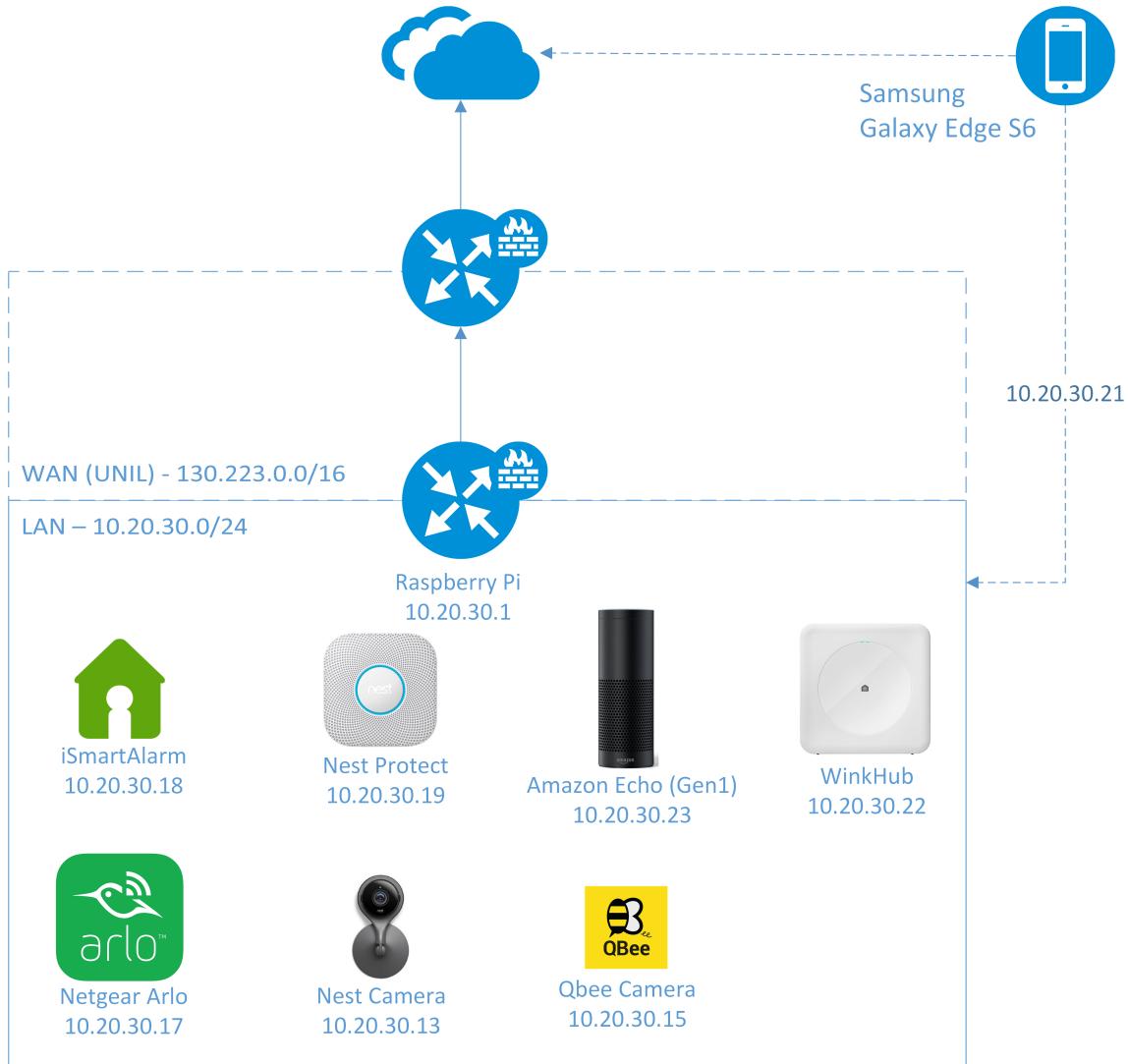


FIGURE 3.2 – Topologie logique du réseau

Afin de pouvoir procéder à la génération, collecte et analyse des données, les dispositifs ont été installés dans un faux laboratoire de stupéfiants mis à disposition par l'ESC.

L'installation du réseau repose sur un Raspberry Pi 3<sup>6</sup> fournissant les services de routage, pare-feu avec nat, point d'accès WiFi et serveur DHCP, ainsi que sur un commutateur à 5 portes.

6. <https://www.raspberrypi.org>

Les schémas 3.2 et A.2 résument la topologie du réseau installé.

Les dispositifs à disposition ont des moyens différents de se connecter au réseau :

- **WiFi** - Amazon Echo, Nest Camera & Nest Protect, WinkHub
- **WiFi (WPS)** - QBee Multi-Sensor Camera
- **Ethernet** - Netgear Arlo, iSmartAlarm CubeOne

Le Raspberry Pi ne présente qu'un seul port Ethernet. Afin de pouvoir le connecter à Internet via le réseau de l'Université de Lausanne (UNIL) il a donc été configuré comme un "one-armed router" (Cisco, 2014a). Une sous interface (eth0.10) de l'interface Ethernet principale (eth0) a donc été créée en tant que VLAN avec ID 10. Le seul port Ethernet du Raspberry Pi a ensuite été lié au commutateur avec une connexion en modalité trunk (VLAN Native & VLAN 10).

**WAN** Le trafic vers Internet est transmis sur le VLAN natif. Celui-ci est connecté au réseau de l'UNIL via un port standard du commutateur.

**LAN** Le trafic local est de deux types, via Ethernet ainsi que via WiFi. Le trafic Ethernet est isolé sur le VLAN 10 et les dispositifs sont connectés à des ports du commutateur configurés en modalité d'accès sur le VLAN 10. Le réseau WiFi (wlan0) a été lié en mode passerelle (bridge) avec le réseau câblé (eth0.10) afin d'avoir un seul réseau avec topologie plate (br0). Structuré ainsi ce réseau représente la situation typique des routeurs domestiques, avec les dispositifs qui peuvent communiquer librement entre eux dans le LAN au niveau 2. De plus, cela permet d'avoir un seul point de passage pour tous les paquets, qu'ils soient transmis via WiFi ou Ethernet. Ainsi il est aisément possible de capturer tout le trafic généré par les dispositifs, avec l'exception de celui entre le Arlo et le iSmartAlarm, si existant, à cause du commutateur présent entre ceux-ci et le Raspberry Pi.

Le trafic sur l'interface *br0* à une destination externe au sous-réseau 10.20.30.0/24 est masqué par NAT avec l'adresse de la Raspberry Pi et acheminé sur *eth0* vers le réseau de l'UNIL.

**Logiciels** Les logiciels suivants ont été utilisés afin de fournir les services réseau sur le Raspberry Pi ; la configuration a été préalablement testée sur une machine virtuelle Debian afin de s'assurer de son bon fonctionnement (cf. Annexe A.2).

- **NAT** - iptables
- **WiFi & WPS** - hostapd
- **DHCP** - isc-dhcp-server

#### — UPnP - Linux IGD

**Liens entre dispositifs** Dans le but d'étudier les interactions dans un environnement hautement connecté, les comptes de Arlo et de Nest ont été liés au compte Wink. De plus la *Skill* de iSmartAlarm a été installée sur l'Echo, afin de pouvoir contrôler le système d'alarme à travers de Alexa.

## 3.3 Analyse du réseau

Les dispositifs IdO sont par nature des dispositifs connectés, la majeure partie des données ne réside pas sur les dispositifs, qui sont principalement des senseurs, mais sont directement envoyées à des services cloud centralisés pour être traitées et mises à disposition de l'utilisateur.

L'analyse du trafic généré par ces dispositifs est donc d'une importance primordiale. D'une part, cela permet de comprendre quelles données sont envoyées, en quelles quantités et à qui. D'autre part, cela peut donner un excellent aperçu de quels dispositifs utilisent des protocoles peu voire non sécurisés et pourraient donc être vulnérables à des attaques sur le réseau local ou via Internet.

Dans cette situation, la distinction est faite entre les dispositifs qui envoient les données directement au cloud, et ceux qui exploitent une passerelle pour l'envoyer aux services cloud. Cette deuxième catégorie est composée principalement par les dispositifs qui utilisent un protocole différent du WiFi ou de l'Ethernet (eg. 802.15.4/Zigbee, Zwave, Bluetooth LE ou d'autres protocoles propriétaires) ; ils nécessitent un intermédiaire qui puisse retransmettre les données vers Internet et des dispositifs utilisant d'autres protocoles ; IKEA's Trådfri smart lights, Philips Hue ou encore le système de iSmartAlarm ne sont que des exemples de ce type de dispositifs. Il existe par contre aussi des exemples de dispositifs utilisant des protocoles communs tels que le WiFi, mais qui néanmoins utilisent une passerelle afin de créer un propre réseau isolé, comme c'est le cas du Netgear Arlo, qui utilise la station de base pour créer son propre réseau WiFi.

## 3.4 Analyse applications officielles

Les dispositifs IdO font partie des niveaux les plus bas du modèle de Cisco, il ne s'agit en principe que de senseurs, les données générées sont envoyées aux services cloud (ou à des passerelles locales), traitées, et ensuite mises à disposition de l'utilisateur pour la consommation

via des applications en ligne ou des applications pour smartphone. Dans cette étape, on procède à l'analyse des applications sur le smartphone dans le but de déterminer quelles informations sont disponibles à travers l'interface officielle, mais aussi quelles traces et quels éléments potentiellement utiles sont sauvegardés et cachés sur le smartphone.

Afin de pouvoir différentier les traces stockées en arrière-plan et celles disponibles suite à un téléchargement de données de la part de l'application, un protocole d'extraction du téléphone analysé est proposé ci-dessous :

1. Extraction du téléphone, physique si possible
2. Analyse des données disponibles dans l'interface des applications (si possible, dans un premier temps sans connexion réseau et après avec les applications connectées à Internet)
3. Deuxième extraction du téléphone, physique si possible

Cela permettra de savoir à quelles traces on pourra s'attendre lors d'une investigation dans laquelle le smartphone n'a pas été utilisé après la génération d'événements par les dispositifs IdO.

A ce stade l'analyse de données comprendra une étape de décodage des traces et une corrélation avec les activités génératrices qui seront connues. Sur cette base, il sera possible de développer des scripts qui pourront aider à extraire les informations pertinentes, et les intégrer en tant que plug-ins pour des plateformes tels que Autopsy<sup>7</sup> ou Plaso<sup>8</sup>.

Dans le cadre de ce travail, un Samsung Galaxy Edge S6 ainsi qu'un iPhone 6 étaient à disposition. L'analyse s'est concentrée sur les données disponibles sur le système Android pour lequel il était possible effectuer une extraction physique. Une extraction de l'iPhone a été effectuée seulement après la mise en scène du challenge forensique.

## 3.5 Recherche de vulnérabilités et points d'entrée

Alors que l'étape précédente se concentrait sur la recherche et analyse de traces d'intérêt forensique, le but principal est ici de trouver des vulnérabilités dans les dispositifs.

A ce stade on veut déterminer quels points d'entrée existent sur un dispositif ainsi que ses vulnérabilités. La recherche des points d'entrée repose sur l'analyse du trafic faite préalablement, ainsi que sur les informations qu'on peut avoir obtenu par l'analyse et la décompilation des

---

7. <http://www.sleuthkit.org/autopsy/>

8. <https://github.com/log2timeline/plaso>

applications. Les vulnérabilités sont cherchées dans plusieurs surfaces d’attaque décrites par le "Surface Attack Project" de OWASP, notamment *Device Network Services*, *Network Traffic* et *Authentication/Authorization*. Pour les dispositifs dont on arrive à obtenir le firmware, la surface *Device Firmware* est aussi d’intérêt.

Les vulnérabilités trouvées seront évaluées selon l’échelle CVSS<sup>9</sup> et reportés aux producteurs selon un processus de divulgation responsable.

## 3.6 Analyse Physique & Accès Root

Dans cette étape on cherche à exploiter les connexions présentes sur les dispositifs, qu’elles soient USB ou UART, afin d’obtenir un accès permettant d’effectuer une extraction des données sur le dispositif, que cela soit avec une image du système de fichiers depuis une console du système, ou avec une image de la mémoire depuis le bootloader. L’accès aux ports JTAG ne sera pas testé faute du manque d’expertise et de matériel d’analyse. Pour l’accès aux ports série via UART, des dispositifs tels qu’un BusPirate ou des adaptateurs FTDI USB->Série peuvent être utilisés.

---

9. <https://nvd.nist.gov/vuln-metrics/cvss>

# 4 Résultats

TABLE 4.1 – Résumé des résultats principaux

Dispositif	Réseau	Applications mobiles	Physique	Accès par le Réseau	Vulnérabilités
ARLO	-	Identifiants Cloud (token) Dispositifs Liés Vignettes en cache	Mdp WiFi (Image Mémoire) Réglages et Logs (Accès Root)	Console Telnet (Depuis réseau interne)	-
NEST	-	Informations sur l'utilisateur Dispositifs Liés Événements Extraits Vidéos	Logs via USB (Nest Protect)	-	-
QBee	Trafic en clair Ouverture Porte par UPnP	Identifiants cloud	-	-	Trafic en clair Mot de passe chiffré sur Android
iSmartalarm	Logs de diagnostic	Identifiants Cloud Evènements enregistrés Infos sur Topics MQTT Dispositifs découverts par UPnP	Images mémoire	Accès aux logs de diagnostic	Accès aux logs non authentifié Mot de passe en clair sur Android
Wink	Announces SSDP	Informations sur l'utilisateur Dispositifs Liés Evénements (Stockage à long terme)	Image Système de Fichiers	Console SSH	-
Echo	Discovery SSDP	-	-	-	-

## 4.1 Analyse préliminaire

Cette section présente la littérature existante sur les dispositifs étudiés ainsi que les informations sur les vulnérabilités déjà connues. Un tableau résumant les informations concernant les chips d'intérêt des différents dispositifs ainsi que les méthodes de connexion physique potentielles, identifiées à l'aide de la documentation du FCC et d'autres sources ouvertes est présenté en annexe (tableau A.1).

**Nest Camera, Nest Protect, WinkHub & Amazon Echo** Rajewski (2016, 2017) présente les artefacts laissés sur un smartphone Android par les applications de Nest et de Wink. Les artefacts concernant l'application de Amazon Alexa pour Android sont traités en détail dans les travaux

de Chung et al. (2017), Hyde et Moran (2017), Rajewski (2016, 2017). Les API non officielles de Alexa sont décrites par Chung et al. (2017). Les dispositifs *Arlo*, *iSmartAlarm* et *QBee* et les artefacts de leurs applications n'ont pas été étudiés dans la littérature.

**Netgear Arlo Pro** NewSky Security (2016b) avaient étudié la première version de la station de base de Arlo. Dans leur recherche ils avaient obtenu un accès à la console administrative du système au travers une connexion série. Le réseau WiFi de la première version du Arlo était aussi vulnérable à une attaque par force brute (NewSky Security, 2016a) ; toutefois il est précisé que cette vulnérabilité a été corrigée dans les versions suivantes.

**iSmartAlarm** Ching (2017), Pidhirney (2017), Shnaidman (2017) ont étudié le système de iSmartAlarm. Il a été trouvé que le système était vulnérable à des attaques de type Man In The Middle (MITM) et Denial of Service (DoS) ; de plus deux vulnérabilités de type *Missing Access Control* et *Authentication Bypass* étaient présentes (Shnaidman, 2017). Selon les observations de Ching (2017) sur les firmwares 2.2.4.7 et 2.2.4.8 le fichier de log, sauvegardé sur une clé USB lors de sa connexion, est chiffré avec une clé symétrique ; la clé de chiffrement était disponible dans le firmware. Aucune mention n'était disponible pour savoir si cela était toujours le cas avec les versions plus récentes du firmware. Pidhirney (2017) avait étudié le dispositif plus pour ce qui concerne les vulnérabilités du protocole RF utilisé par le CubeOne pour communiquer avec les senseurs et la télécommande. Cependant il relève que la vulnérabilité de udhcpc de busybox (*CVE-2011-2716*) s'appliquait au CubeOne, busybox étant une version obsolète et il arrive correctement à l'exploiter.

## 4.2 Analyse du Réseau

L'analyse du réseau a été principalement conduite sur les données récoltées au cours d'une semaine d'utilisation des dispositifs et des applications reliés.

La capture principale a été effectuée avec *tcpdump* (The Tcpdump Group, 1999–) sur tout le trafic de l'interface *br0* entre le 19.03.2018 et le 28.03.2018. La capture a été interrompue brièvement le 27.03.2018 à 16 h 24 pour obtenir une première copie à analyser, l'enregistrement a été poursuivi ensuite pour encore une journée avant d'être arrêté.

Au total environ 27 Go de données de trafic ont été capturées, partagées sur 15 fichiers pcap de taille inférieure à 2Go.

Les données ont été analysées à l'aide de plusieurs logiciels. Au vu de la taille des données, il n'était pas possible de les analyser directement à l'aide du logiciel Wireshark (Combs & Contributors, 2006–); les données ont donc été indexées et chargées dans Elasticsearch (Elastic, 2010–) avec Moloch (AOL, 2012–), afin de pouvoir effectuer des visualisations et des analyses avec les outils intégrés de Moloch et avec Kibana (Elastic, 2014–).

Lors de cette première analyse, il a été constaté que la plupart des données étaient composées du trafic vidéo transmis en continu par la caméra de Nest.

En utilisant CapLoader ainsi que des outils Free and Open Source Software (FOSS), une version réduite de la capture, ne comprenant pas le trafic vidéo de la caméra de Nest, a été produite (filtered.pcap - 1.5 Go). A partir de ce fichier, le trafic lié à chaque dispositif a été filtré avec tcpdump, produisant 7 fichiers pcap. Le fichier de chaque dispositif a été ensuite analysé à l'aide de NetworkMiner (NETRESEC, 2007–) ainsi que Wireshark quand une analyse plus approfondie était nécessaire. Une recherche des ports ouverts pour chaque dispositif a aussi été faite avec *nmap* (Lyon & Contributors, 1997–).

Suite à des informations supplémentaires découvertes pendant le travail, plusieurs plus petites captures ont été effectuées. Ces captures ciblaient des dispositifs spécifiques, afin de mieux comprendre certaines interactions. Deux d'entre eux, arlo\_samsung\_1004.pcap, ismart\_diagnostics.pcap et qbee\_1605.pcap ont été ensuite ajoutés au dataset de Moloch.

Pour l'analyse du réseau, on s'intéresse uniquement au trafic généré par les dispositifs IdO, le smartphone Samsung ainsi que la passerelle (Raspberry Pi). D'autres dispositifs tels que l'ordinateur ou le portable personnel étaient aussi connectés au réseau afin de faciliter le travail, mais le trafic généré n'est pas pertinent dans le cadre de l'analyse, et a donc été filtré. Sur Moloch les protocoles de service, tels que DNS, NTP et DHCP ont été filtrés pour éviter de surcharger la visualisation relationnelle avec des données non pertinentes.

### 4.2.1 Composition générale du trafic

Comme mentionné précédemment, l'analyse du trafic en entier a été effectuée avec une combinaison de Moloch et Kibana avec Elasticsearch et de CapLoader.

Une première analyse montrait un nombre élevé de connexions ayant comme adresse d'origine une adresse externe (dont par exemple environ 10 Go de données échangées entre 52.16.154.142 et la caméra Nest). Ceci était dû au fait que la connexion était déjà active quand la capture a débuté. La connexion est donc classée comme entrante, même si en réalité elle avait

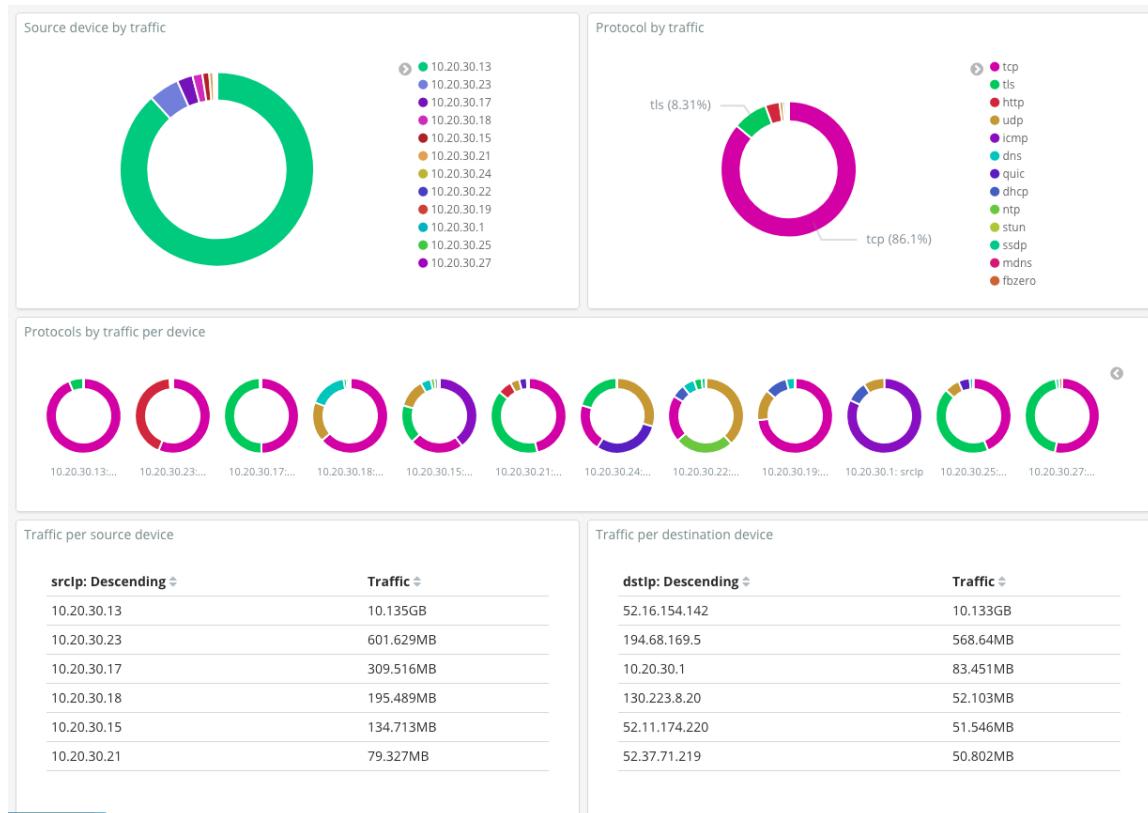


FIGURE 4.1 – Visualisation du trafic réseau sur Kibana

été établie en premier lieu depuis l'intérieur. Au vu du fait que des connexions entrantes, avec la configuration réseau implémentée, n'étaient pas possibles, et afin de faciliter l'analyse, seules les connexions avec origine depuis le réseau local ont été considérées (10.20.30.0/24).

On peut aisément voir dans la visualisation sur Kibana (figure 4.1 page 20) que la majeure partie du trafic (88%) est généré par la caméra de Nest.

En utilisant la visualisation relationnelle de Moloch, il a été possible d'avoir une vue d'ensemble des destinations des connexions de tous les dispositifs. Une telle visualisation est présentée à la figure 4.2 à la page 21.

Plusieurs observations peuvent être faites sur la base du schéma relationnel de Moloch :

- Le Arlo, et les dispositifs Nest ne sont pas reliés au Samsung (au centre)
- Le CubeOne<sup>1</sup> est lié indirectement au Samsung
- Le Qbee est lié directement et indirectement au Samsung
- Le Wink Hub est lié directement et indirectement au Samsung
- Le Wink Hub est lié directement et indirectement à la caméra QBee
- Les liens indirects avec le WinkHub sont à travers de l'adresse multicast 239.255.255.250

1. iSmartAlarm sur le schéma

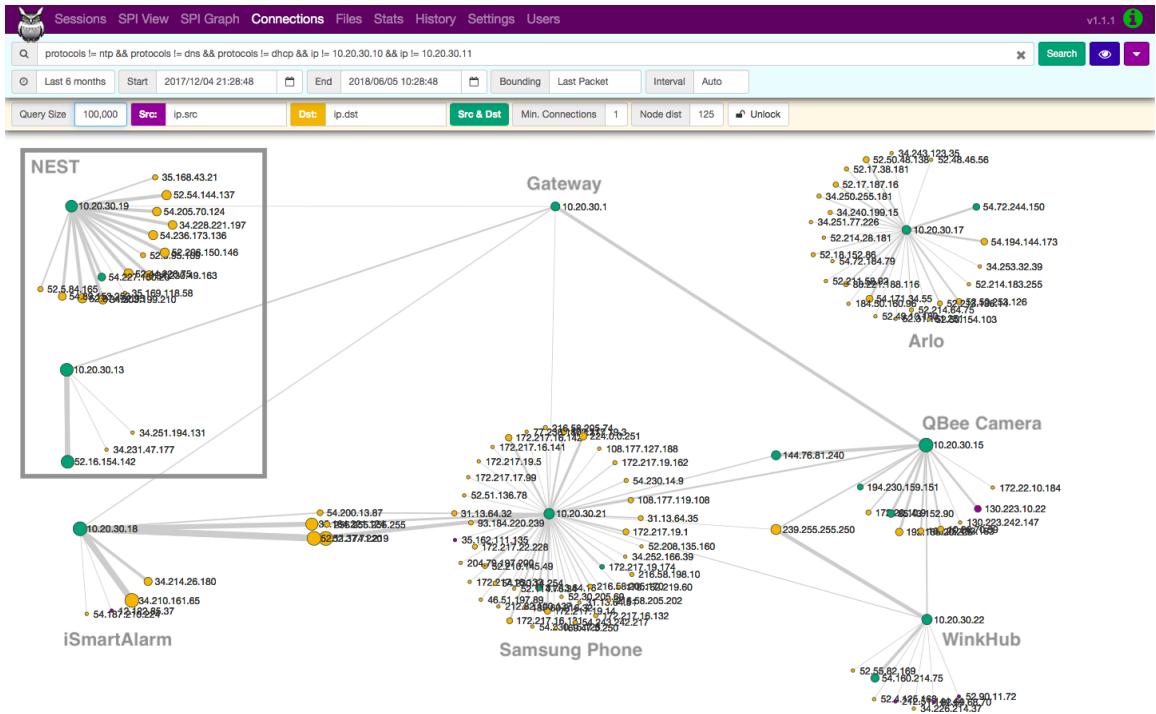


FIGURE 4.2 – Relations entre dispositifs

utilisée par le protocole Simple Service Discovery Protocol (SSDP)

- Le Amazon Echo n'est pas présent

### 4.2.2 Amazon Echo

L'analyse avec NetworkMiner ne révèle pas de connexions en entrée ; toutes les connexions sont établies par le dispositif. La plus grande partie du trafic du Amazon Echo est composé par du trafic HTTP avec les serveurs de Spotify (96%). On retrouve ensuite du trafic TLS vers des serveurs appartenant à Amazon, ainsi que des requêtes HTTP à spectrum.s3.amazonaws.com/kindle-wifi/wifistub-echo.html ; cela concorde avec les découvertes de Micaksica (2017). Une partie des requêtes HTTPS est faite à guipitan.amazon.com qui pointe vers la même page que le plus connu pitangui.amazon.com (Chung et al., 2017).

Ni le trafic HTTP avec Spotify ni celui avec spectrum.s3.amazonaws.com ne contiennent des informations sensibles ni utiles.

Du trafic UDP, dont une partie composée par des requêtes SSDP, est aussi présent.

Les ports ouverts trouvés sont présentés au tableau 4.2. Dans le cas de l'Amazon Echo, l'identification manuelle des services n'a pas été entreprise.

TABLE 4.2 – Ports ouverts - Amazon Echo

Port	Service selon nmap	Service identifié
4070/tcp	tribe	-
4071/tcp	aibkup	-
55442/tcp	nagios-nsca	-
55443/tcp	inconnu	-

### 4.2.3 QBee Multi-Sensor Camera

Le trafic généré par la caméra QBee est composé principalement par du trafic ICMP (40%) et TCP/TLS(39%), suivi du trafic UDP (12%) ainsi qu'en plus petite partie par le trafic de service (DNS/DHCP) et du trafic SSDP (UPnP).

La caméra se connecte aux serveurs à distance "hma.vestiacom.com", avec TLS sur le port 443 et avec TCP en clair sur le port 7402. Les sessions sur le port 443 sont sporadiques, une partie d'entre elles (celles entre 7 h 33 et 11 h 4 du 28.03.2018) ont pu être corrélées avec des détections de mouvement de la part de la caméra. Les sessions sur le port 7402 par contre sont gardées actives en continu, il s'agit de la connexion "keep alive" permettant d'envoyer des commandes depuis le serveurs à la caméra. L'annexe A.3 présente plus en détail cette connexion. Les connexions sur le port 7402 n'utilisent pas TLS, mais les données sont encodées ou chiffrées de manière inconnue, et aucune information sensible n'a pu être retrouvé.

Concernant les connexions en entrée, la caméra reçoit des connexions de type HTTP sur le port 15700. Il s'agit de connexions faites par le smartphone Samsung à l'interface API de la caméra. L'analyse avec Moloch et Wireshark permet de relever les points d'entrée présentés dans l'annexe A.3 (Table A.2). Pour chacun, les messages échangés, ainsi que les cookies associés, ont pu être observés. Il a ainsi été possible d'établir la fonction de chaque point d'entrée.

La connexion à un de ces points d'entrée, "/verify", est faite de manière périodique par l'application, les autres points d'entrée sont contactés seulement suite à une interaction de l'utilisateur avec l'application.

Les ports ouverts trouvés sont présentés au tableau 4.3.

TABLE 4.3 – Ports ouverts - QBee Multi-Sensor Camera

Port	Service selon nmap	Service identifié
8889/tcp	HTTP	Inconnu
15700/tcp	-	API - HTTP
15701/tcp	Inconnu	API - HTTPS

Concernant l’accessibilité depuis l’extérieur, la caméra QBee est le seul dispositif à tenter d’ouvrir des portes dans le pare-feu par UPnP, comme décrit plus en détail à l’annexe A.3.

#### 4.2.4 iSmartAlarm CubeOne

Le CubeOne du système de iSmartAlarm génère du trafic HTTP vers api.ismartalarm.com sur le port 8443. Cela représente ~78% du trafic sortant. Le reste du trafic en sortie est de type UDP, utilisé pour les services DNS, DHCP et NTP.

En entrée, on retrouve du trafic TCP sur le port 12345, qui provient du téléphone Samsung. Le trafic sur ce dernier port a été présenté en détail par Shnaidman (2017) et correspond à celui observé.

TABLE 4.4 – Ports ouverts - iSmart Alarm Cube One

Port	Service selon nmap	Service identifié
12345/tcp	netbus	Contrôle
22306/tcp	-	Diagnostique <sup>2</sup>
54321/tcp	Inconnu	Inconnu

Du trafic additionnel sur le port 22306 est observé lors de la collecte des logs de diagnostic, et est présenté à l’annexe A.8.

#### 4.2.5 Nest Camera

Le trafic généré par la Nest Camera est composé exclusivement de trafic TCP (~10 Go), avec l’exception du trafic UDP de service tel que DNS, NTP et DHCP. Ce trafic TCP est adressé à nexus.dropcam.com ou nexus-ire.dropcam.com sur le port 443. De ce trafic, une partie est

2. Seul ceci a été observé, l’on ne peut pas exclure d’autres utilisations

indexée par moloch comme trafic TCP/TLS (~700 Mo), tandis qu'une partie est seulement marqué comme TCP (~9 Go). Cette deuxième partie est composée par des flux TCP considérés par Moloch comme incomplets, il s'agit de flux qui ont été capturés au milieu de la conversation. Dans les deux cas les connexions sont faites sur le port 443, il s'agit vraisemblablement dans sa totalité de trafic TLS, même si l'indexation n'est pas correcte.

Le dispositif ne présente pas de ports ouverts.

#### 4.2.6 Nest Protect

Le trafic généré par le Nest Protect est composé en majeure partie par du trafic TCP vers le port 11095 des serveurs de Nest (frontdoor.nest.com, czfe\*.front01.iad01.production.nest.com ou log-rts08-iad01.devices.nest.com). Le trafic restant est composé par le trafic UDP de service (DNS, DHCP et NTP).

Le port utilisé suggère qu'il s'agisse de trafic avec le protocole Weave de Nest<sup>3</sup>.

TABLE 4.5 – Ports ouverts - Nest Protect

Port	Service selon nmap	Service identifié
11095/tcp	weave	-

#### 4.2.7 Netgear Arlo

La station de base du Arlo Pro génère presque exclusivement (99.18%) du trafic TCP/TLS vers le port 443 de multiples serveurs de Netgear (www.downloads.netgear.com, mcs.netgear.com, arlo-device.messaging.netgear.com et vzwow\*\*-z1-prod.vz.netgear.com).

Une capture additionnelle (arlo\_samsung.pcap) a été effectuée pour mieux étudier l'interaction du Arlo avec l'application sur le réseau local. Dans cette capture il a été observé que la station de base utilise aussi le protocole Session Traversal Utilities for Network Address Translators (STUN) pour se connecter sur le port 19302 de relay02-z1-prod.vz.netgear.com ; deux ports source sont observés, 58811 et 45544. Entre le téléphone Samsung et la station de base du trafic de type STUN et Datagram Transport Layer Security (dTLS) entre le port 58811 (Arlo) et le port 48418 est échangé. Le trafic avec le téléphone Samsung est décrit plus en détail dans l'annexe A.4.

3. <https://openweave.io/>

Le dispositif ne présente pas de ports ouverts.

**Réseau Interne** Le trafic entre la caméra et la station de base se fait par WiFi sur un réseau isolé créé par la station de base. Ce réseau<sup>4</sup> est protégé par WPA2 et les camera se connectent via WPS. Le trafic ainsi que la procédure d'accès au réseau sont décrits dans la section 4.5.3.

#### 4.2.8 Wink Hub

Le trafic du Wink Hub est composé principalement par du trafic UDP de service. En effet le trafic combiné NTP, DNS et DHCP compte pour ~55% du trafic généré par le dispositif. Le dispositif se connecte aussi via HTTPS (~34%) à hub-api.winkapp.com sur le port 8080 ainsi qu'à cinq autres serveurs sur le port 443.

Du trafic SSDP est observé entre le dispositif, l'adresse multicast, la caméra QBee et le téléphone Samsung.

Du trafic TLS en entrée sur le port 8888, généré par le téléphone Samsung, est aussi présent.

Le dispositif signale https://10.20.30.22:8888 dans le champ "LOCATION" de ses informations sur Universal Plug and Play (UPnP).

TABLE 4.6 – Ports ouverts - Wink Hub

Port	Service selon nmap	Service identifié
80/tcp	HTTP - WinkHub 2 API HTTP	-
8886/tcp	-	-
8888/tcp	HTTP - WinkHub 2 API HTTP	-

#### 4.2.9 Samsung

Le Samsung contacte tous les dispositifs sur le port 137 avec le protocole Netbios. Les dispositifs Nest, Arlo, iSmartAlarm et le WinkHub répondent seulement avec un message ICMP "Port Unreachable", le Amazon Echo ne répond pas du tout au trafic. Le trafic n'a pas été jugé pertinent dans le cadre de ce travail, il s'agit probablement uniquement du système de découverte de dispositifs sur le réseau local de l'application "Peel Remote" du Samsung.

4. ESSID : NETGEAR76 avant la réinitialisation de la base et NTGR\_VMB\_1526289002 après

## 4.3 Artefacts Applications pour Smartphone

UFED 4PC de Cellebrite a été utilisé pour effectuer les extractions physiques du téléphone Samsung analysé. Pour cela l'extraction physique par bootloader a été choisie afin d'obtenir des images disque complètes du téléphone (cf. table 4.7).

TABLE 4.7 – Samsung - Images Extraites

Filename	Taille
blk0_sda.bin	32Go
blk16_sdb.bin	4Mo
blk32_sdc.bin	4Mo
procdata.zip	1Mo

L'analyse du contenu du téléphone a été faite avec le rapport UFED Reader (Exporté depuis l'extraction physique via UFED Physical Analyser) ainsi qu'avec Autopsy suite à l'importation de **blk0\_sda.bin**.

Dans cette section, le terme "données de l'application" se réfère aux données disponibles dans le dossier de l'application sous "USERDATA/root/data/", par exemple "USERDATA/root/data/-com.netgear.android". Sauf indication contraire, les chemins sont relatifs à ce dossier.

### 4.3.1 Amazon Alexa

L'application de Amazon Alexa (*com.amazon.dee.app*) et les traces qui y sont contenues ont déjà été étudiées de manière approfondie (Chung et al., 2017; Rajewski, 2016, 2017). L'analyse de cette application n'a donc pas été jugée nécessaire.

### 4.3.2 QBee

La caméra QBee peut être contrôlée par l'application "QBee Camera", mais aussi par l'application "Swisscom Home App". Les deux ont été installées et ont été analysés afin de comparer les traces disponibles.

Au travers de l'interface des deux applications il est possible d'avoir accès et de télécharger les images et vidéos enregistrés par la caméra dans les derniers sept jours.

Dans les données des deux applications aucune trace concernant les activités de la caméra n'a pu être trouvée. Cependant des traces concernant les comptes utilisés pour la connexion ont été identifiées et sont présentées ci-après.

### **QBee Camera**

**App Identifier** com.vestiacom.qbeecamera

**App Version** 1.0.5

L'application ne stocke pas en cache des fichiers ou des logs concernant les évènements de la caméra, tels que des détections de mouvement. Le seul artefact d'intérêt de l'application est constitué par le fichier *com.vestiacom.qbeecamera\_preferences.xml*.

**shared\_preferences/com.vestiacom.qbeecamera\_preferences.xml** Ce fichier XML contient les réglages de l'application QBee. L'attribut "name" des tags ainsi que leur contenu sont composés par des chaînes en base64. Il s'agit d'un fichier de configuration chiffré avec une version modifiée de la bibliothèque "Secure-preferences"<sup>5</sup> disponible pour Android. Afin de pouvoir déchiffrer le fichier, le code source de la bibliothèque est étudié et comparé à la version décompilée de l'application de QBee, comme décrit en détail à l'annexe A.5. Sur la base des observations faites, un script python permettant de déchiffrer le fichier est écrit et intégré dans un plug-in pour Autopsy<sup>6</sup>. Déchiffré, le fichier contient le nom d'utilisateur et le mot de passe du compte utilisé pour effectuer l'accès.

Une partie des réglages non chiffrés sont présents dans le fichier *com.vestiacom.qbeecamera.preferences.xml*, mais le contenu n'est pas d'intérêt.

### **Swisscom Home App**

**App Identifier** com.swisscom.internetbox

**App Version** 10.5.2

L'application *Swisscom Home App* ne garde pas en cache de données ou de logs concernant les évènements déclenchés par la caméra. Par contre comme dans le cas de l'application *QBee Camera*, un fichier chiffré se trouve dans les préférences de l'application, dont le nom est *com.swisscom.internetbox\_preferences.xml*. Egalement, une partie de réglages non chiffrés sont

---

5. <https://github.com/scottyab/secure-preferences>

6. [https://github.com/fservida/iot\\_autopsy\\_plugins/tree/master/qbee](https://github.com/fservida/iot_autopsy_plugins/tree/master/qbee)

présents dans le fichier *com.swisscom.internetbox.preferences.xml*, mais n'ont pas d'intérêt forensique.

**shared\_preferences/com.swisscom.internetbox\_preferences.xml** Le fichier contient la majeure partie des réglages de l'application. Le fichier peut être déchiffré avec la même procédure que pour le fichier de configuration de l'application *QBee Camera*; la comparaison des applications décompilées montre que le code de chiffrement utilisé est le même. On y retrouve les identifiants pour l'accès au service VoIP de l'Internet Box de Swisscom depuis son réseau local et les identifiants pour un compte @swisscom.com. Le plug-in de Autopsy développé pour l'application *QBee Camera* procède aussi au déchiffrement de ce fichier s'il est présent dans le système de fichier analysé.

### 4.3.3 iSmartAlarm

**App Identifier** iSA.common

**App Version** 2.0.8

L'interface de l'application permet d'accéder à la totalité<sup>7</sup> des logs concernant les événements des senseurs, les changements d'états de la base et l'utilisateur responsable, ainsi que les alarmes déclenchées. Les événements déclenchés par la Skill pour Amazon Alexa associé au compte sont enregistrés de manière équivalente à un changement d'état effectué depuis l'application. Il n'y a pas moyen de les différencier.

L'analyse des données de l'application de iSmartAlarm a permis de trouver plusieurs fichiers et dossiers d'intérêt. Ceux-ci sont présentés ci-dessous.

**shared\_preferences/iSmartAlermData.xml** Il s'agit du fichier de configuration de l'application, sous forme XML. On y retrouve le nom d'utilisateur ainsi que le mot de passe en clair utilisé pour la connexion au service. De plus l'ID du client Message Queue Telemetry Transport (MQTT) ainsi que les *topics* sont à disposition dans le fichier.

**shared\_preferences/MQTT\_Message\_Service.xml.bak** Il s'agit d'un fichier effacé. Le contenu n'est présent que sur Autopsy, UFED reconnaît le fichier, mais n'arrive pas à retrouver le contenu. Selon l'extraction il contient différentes données, sur la première, il contient des

---

7. Depuis l'installation du système

informations par rapport à un dispositif UPnP détecté sur le réseau (notamment une imprimante), tandis que sur l'extraction effectuée dans le cadre du scénario pour le challenge forensique du DFRWS il contient une multitude de réglages réseau, vraisemblablement concernant le service MQTT.

**databases/iSmartAlarm.DB** Cette base de données SQLite contient les informations concernant les dispositifs et les évènements enregistrés par la base *CubeOne*. Les changements d'état (Arm, Disarm, Home...), ainsi que l'utilisateur les ayant déclenchés sont également disponibles. Trois tables intéressantes sont identifiées : *TB\_IPUDairy*, *TB\_SensorDairy* et *TB\_userDairy*. Les entrées dans les trois tables ont été corrélées temporellement avec les évènements connus afin de pouvoir parser correctement les différentes activités. Ceci a été utilisé afin de créer un plug-in pour Autopsy pouvant effectuer le parsing des évènements. Une explication détaillée du contenu des tables ainsi que du développement du plug-in est fournie à l'annexe A.6 Ce plug-in permet également d'automatiser l'extraction des identifiants depuis le fichier de configuration.

**USERDATA/root/media/0/iSmartAlarm/Log/** Ce dossier contient plusieurs fichiers log concernant l'application iSmartAlarm (fichiers \*.txt). On retrouve des informations de log concernant l'utilisation de l'application. En particulier, l'application enregistre le ESSID et le BSSID du réseau WiFi auquel le téléphone est connecté pendant l'utilisation.

**USERDATA/root/media/0/iSmartAlarm/Image/Logo/0792245315.png** Il s'agit de l'image du profil de l'utilisateur. Le nom du fichier est son numéro de téléphone associé, au format national.

**Logs de diagnostic** L'application de iSmartAlarm est la seule qui met à disposition une manière d'envoyer des logs de diagnostic au service technique. Afin de comprendre le fonctionnement du système d'envoi des logs, une capture réseau du trafic du téléphone et du CubeOne est effectuée pendant l'envoi (ismart\_diagnostics.pcap). On peut ainsi faire les observations suivantes : sur le réseau local l'envoi des logs est fait à travers du téléphone, le trafic est en clair et la requête ne nécessite pas d'authentification. L'annexe A.8 présente en détail le processus. Les logs ne sont pas enregistrés dans le téléphone. On retrouve dans ces logs les requêtes qui ont été faites au serveur cloud de la part du CubeOne. Les changements d'états (Arm, Disarm, Home...) effectués au travers de l'application, ainsi que les changements d'état dans des senseurs

tels que ceux de contact sont enregistrés. Pour les changements faits au travers de l’application aucune informations sur l’utilisateur n’est détectée au travers de ces logs. Cependant la structure des logs n’est pas entièrement comprise ; une partie importante est composée de données binaires qui n’ont pas pu être décodées.

Ces logs couvrent toute la période depuis le dernier démarrage du système, et sont effacés dès que le CubeOne est redémarré. Un changement de connexion réseau, tel que débrancher et rebrancher le câble Ethernet, n’entraîne pas la perte de ces logs, même lors d’un changement des réglages réseau (subnet, adresse IP...).

Un script python a été écrit afin d’automatiser la collecte et le parsing d’une partie de ces logs.

#### 4.3.4 Nest

**App Identifier** com.nest.android

**App Version** 5.19.1.2

Les traces identifiées pour cette application correspondent à celles retrouvées par Rajewski (2016, 2017). Le script mis à disposition sur GitHub<sup>8</sup> a été utilisé pour vérifier s’il était toujours possible d’extraire des portions de vidéo à partir des bases de données SQLite "frames\_database". Cela est toujours possible, un extrait se trouve à la figure 4.3.

L’application permet d’accéder à la totalité des logs concernant la détection de fumée, ainsi qu’au flux vidéo et les évènements enregistrés par la caméra dans la période couverte par le plan de Nest Aware<sup>9</sup>.

#### 4.3.5 Arlo

**App Identifier** com.netgear.android

**App Version** 2.4.10\_20077

L’application de Arlo permet d’accéder aux vidéos enregistrés par la caméra dans la période couverte par le plan associé au compte Arlo<sup>10</sup>.

Plusieurs traces sont trouvées dans les données de l’application de Arlo, elles sont décrites ci-après.

---

8. <https://github.com/bicyclemicycle/NestVideoConverter> (Rajewski, 2017)

9. Pendant ce travail le plan actif était la version d’essai avec historique de 5 jours

10. Dans le cas d’un plan gratuit, comme celui utilisé pour ce travail, cela correspond à 7 jours d’historique



FIGURE 4.3 – Frame du vidéo extrait de frames\_database

**shared\_preferences/Phoenix.xml** Il s'agit du fichier XML de configuration contenant le nom d'utilisateur et l'ID du compte de l'utilisateur.

**files/default.realm** Cette base de données de l'application au format "Realm"<sup>11</sup> contient sous forme JSON les informations par rapport à l'utilisateur (DatabaseModelAccountData), le plan activé (DatabaseModelServiceModelData), les dispositifs connectés au compte (DatabaseModelDevicesData), la configuration de chaque dispositif (DatabaseModelDeviceCapabilitiesData), ainsi que l'état actuel des caméras (DatabaseModelCamerasData).

Les informations de configuration des dispositifs ne contiennent pas d'informations intéressantes. Par contre dans les informations relatives aux dispositifs associés au compte on retrouve plusieurs informations d'intérêt. Pour tout dispositif, le nom et prénom du propriétaire depuis la dernière réinitialisation est enregistré. Pour la caméra on retrouve trois liens<sup>12</sup> vers la plateforme de stockage S3 de Amazon, depuis lesquels il est possible de télécharger la dernière image enregistrée par le dispositif. Les liens sont valides pour 24 h depuis la dernière utilisation de l'application. Enfin les données par rapport à l'utilisateur contiennent l'adresse email, le userID et un token comme informations intéressantes (cf. Figure 4.4).

---

11. <https://realm.io/>

12. PresignedLastImageURL, presignedSnapshotUrl, presignedFullFrameSnapshotUrl

```

jpinkman2018@gmail.com.json
1 {
2   "userId": "A79GZN-316-31881729",
3   "email": "jpinkman2018@gmail.com",
4   "token": "2_590DKCJDtpWHT00W2mNT-WJN95AXLKMes4ql2RK7BIBk9wDFWBC_ZzpS_BJDCieSgSrcvtwzH4eTIgIhW9Em9aAqleZ7bd00U
dxmrV0IOeB_Xb-L0i33S1y6-_XrweGsKtD3-1D-mG72M_uC-r0z3rdBWQJQNve-lVs8A1de_F1e",
5   "paymentId": "35656366",
6   "accountStatus": "registered",
7   "serialNumber": "4RD37B75A1EC9",
8   "countryCode": "CH",
9   "tocUpdate": false,
10  "policyUpdate": false,
11  "appStore": {},
12  "validEmail": true,
13  "arlo": true,
14  "dateCreated": 1521034216242
15 }
16
17
18
19 }

jpinkman2018@gmail.com.json 20:1
LF  UTF-8  JSON  0 files

```

FIGURE 4.4 – JSON contenant les informations sur l’utilisateur pour le Arlo

**cache/cams, cache/http et cache/thumbs** Ces trois dossiers contiennent des images enregistrées par la caméra. Il s’agit des vignettes qui ont été téléchargées sur le téléphone lors de la visualisation des événements dans l’application sur le téléphone. La comparaison entre la première extraction et la deuxième confirme que ces images sont présentes seulement si l’utilisateur a utilisé l’application. Les images ne sont pas téléchargées en arrière-plan.

Un plug-in pour Autopsy pour automatiser l’extraction des données du Arlo était en développement, mais n’a pas pu être complété.

### 4.3.6 Wink

**App Identifier** com.quirky.android.wink.wink

**App Version** 6.8.0.18

L’application Wink permet d’avoir accès à tous les événements enregistrés par les dispositifs de Nest et Arlo depuis l’association du compte relatif avec le compte Wink. Ceux-ci sont stockés sur les serveurs de Wink et les limitations temporelles associées aux plans de Nest ou Arlo ne s’appliquent donc pas.

Les traces liées à l’application du Wink-Hub ont déjà été étudiées par Rajewski (2016). Cependant, une partie des fichiers et le type de données contenus a changé ; les traces observées sont donc illustrées ci-dessous.

**shared\_preferences/com.quirky.android.wink.wink\_preferences.xml** Ce fichier XML contient le nom du compte utilisateur.

**shared\_preferences/user.xml** Ce second fichier XML contient des informations sur l'utilisateur en format JSON, un encodage HTML. On y retrouve le nom, prénom et email ; des champs oauth2, password et old\_password sont aussi présents, mais sont vides.

**shared\_preferences/wink\_local\_pref\_\*.xml** Ce troisième fichier XML contient la liste des dispositifs connectés, séparés par type (déTECTeur de fumée, caméra, hub...). N'y apparaissent seulement l'information du type de dispositif et son ID dans la base de données SQLite.

**database/persistenceDB** Cette base de données SQLite présente dans la table "Elements" tous les dispositifs et groupes liés au compte, les informations du compte ainsi que les activités enregistrées par les dispositifs. A chaque entrée, des données de type JSON détaillées sont associées dans la colonne "json". Le format de ces données correspond à celui observé par Rajewski (2016) ; les données sur les activités, qui n'avaient pas été présentées, sont aussi d'intérêt. Elles contiennent un horodatage de l'évènement, le dispositif l'ayant généré et son ID, le type du dispositif, ainsi que l'évènement même, sous la clé "action->reading". Un exemple d'évènement de détection de fumée est donné à la figure 4.5.

**cache/image\_manager\_disk\_cache/\*.0** Ce dossier contient plusieurs fichiers avec extension .0, il s'agit d'images qui ont été gardées en cache. Parmi celles-ci se trouvent les images des caméras de surveillance Nest et Arlo.

Pour faciliter l'extraction et l'analyse des données contenues dans la base de données SQLite, un script a été développé en python<sup>13</sup> qui extrait les informations principales sur les activités depuis la base de données.

A la différence de ce qui a été observé par Rajewski (2016) le fichier *wink\_preferences.xml* contenant le SSID du WiFi et son mot de passe n'est plus présent.

La base de données n'est mise à jour que lors de l'utilisation de l'application. Aucune modification en arrière-plan n'est effectuée.

Une base de données de type Realm est identifiée (*files/default.realm*), mais contient seulement les informations par rapport aux dispositifs supportés par le Wink Hub, et non ceux qui ont

---

13. [https://github.com/fservida/iot\\_autopsy\\_plugins/blob/master/wink/wink\\_db.py](https://github.com/fservida/iot_autopsy_plugins/blob/master/wink/wink_db.py)

```
wink_activity_smoke_detect.json
1 {
2   "action": {
3     "action_automation_mode": null,
4     "action_id": null,
5     "action_name": null,
6     "action_type": null,
7     "object_id": null,
8     "object_name": null,
9     "object_type": null,
10    "reading": {
11      "smoke_detected": true
12    },
13    "target_actor_email": null,
14    "target_actor_first_name": null,
15    "target_actor_last_name": null
16  },
17  "activity_id": "89475671-3036-46fb-917e-30df68b336f9",
18  "category": "reading",
19  "context": {
20    "created_at": 1526546166.417,
21    "object": {
22      "object_id": "212474",
23      "object_name": "SuperLab Kitchen Nest Protect (LabSmoker)",
24      "object_type": "smoke_detector"
25    },
26    "tags": [],
27    "icon_id": null,
28    "name": null,
29    "object_id": null,
30    "object_type": null,
31    "subscription": null
32  }
33},
34
35
36
37
38
39
40 }
```

wink.activity\_smoke\_detect.json 36:16 LF UTF-8 JSON 0 files

FIGURE 4.5 – Wink - Détection de Fumée

été appariés au compte.

### 4.3.7 iOS

Au vu de la possibilité d’effectuer des extractions physiques avec le smartphone Android, l’analyse d’iOS, l’extraction duquel ne donne accès qu’à un nombre limité de données a été faite seulement dans un deuxième temps sur un iPhone 6. Celui-ci a été débridé afin de pouvoir effectuer une extraction du système de fichiers par SSH. Les résultats sont présentés à l’annexe A.10.

## 4.4 Analyse des Vulnérabilités, Points d’entrée & Firmware

La recherche des vulnérabilités et des points d’entrée a été effectuée à partir des analyses réseau (connexions et portes ouvertes identifiées), des applications décompilées, ainsi des images système quand celles-ci étaient disponibles. Faute de temps à disposition, seules les applications *QBee Camera*, *Swisscom Internet Box* et *iSmartAlarm* ont été décompilés. Le choix a été dicté par le résultat des analyses précédentes, il s’agissait en effet des plateformes avec le plus de potentiel en ce qui concernait les vulnérabilités, qui étaient donc prioritaires.

**MITM** Pour chacun des dispositifs les ports interceptés sont choisis en fonction des ports contactés vers les services cloud, identifiés dans la section 4.2. Deux logiciels sont utilisés, notamment *mitmproxy*<sup>14</sup> et *SSL Split*<sup>15</sup>. Seul le CubeOne de iSmartAlarm semble être vulnérable à une attaque de type MITM, cela étant cependant dépendant de la connexion et du logiciel utilisé. Lors de l’analyse des connexions interceptées du CubeOne aucune information sensible n’est dévoilée.

**Décompilation** La décompilation de l’application *QBee Camera* a permis de confirmer les points d’entrée retrouvés dans la section 4.2, ainsi que d’en découvrir des nouveaux relatifs aux services cloud. En effet on retrouve, dans le code de l’application, l’interface *com.vestiacom.qbee-camera.http.RetrofitInterface* qui contient les chemins possibles à contacter et la méthode à utiliser. De plus on a ainsi pu déchiffrer les réglages de *QBee Camera* et de *Swisscom Home App* (cf. Annexe A.5). La décompilation de l’application *iSmartAlarm* a par contre permis de trouver l’URL auquel l’application se connecte pour vérifier la présence d’un nouveau firmware (cf. A.7). L’analyse de la méthode appelée pour l’analyse réseau a permis de s’authentifier auprès de cette URL et pouvoir télécharger la version actuelle du firmware utilisé par le CubeOne.

**iSmartAlarm Firmware** Le CubeOne de iSmartAlarm produit un fichier log chiffré dès qu’une clé USB est insérée dans le port disponible sur le CubeOne. Le firmware téléchargé a été extrait avec binwalk, et une recherche de chaînes est faite sur l’exécutable de iSmartAlarm, comme fait par Ching, 2017. Dans la version actuelle du firmware, le chiffrement du fichier log se base désormais sur du chiffrement à clé publique. Seule la clé publique de iSmartAlarm est disponible sur le système, et il est donc impossible d’obtenir le fichier déchiffré. La version du firmware du CubeOne de iSmartAlarm était vulnérable à *CVE-2011-2716* selon les observations de Pidhirney (2017). La version de busybox identifié sur le firmware *2.2.4.10* est *1.12.1*; il s’agit d’une version encore vulnérable. Cependant l’exploitation de la vulnérabilité pour exécuter des commandes n’a abouti à aucun résultat positif.

Aucune autre information intéressante n’a été retrouvée dans le fichier du firmware.

---

14. Cortesi, Hils, Kriechbaumer et Contributors, 2010–.

15. Roethlisberger et Contributors, 2009–.

## 4.4.1 Vulnérabilités Retrouvées

L'analyse des vulnérabilités sur la base de l'analyse réseau, et des données sur le smartphone a permis d'identifier quatre vulnérabilités potentielles qui sont décrites ci-après. Plus de détails sont disponibles à l'annexe A.11.

### QBee Multi-Sensor Camera - Traffic en Clair

Sur le réseau local, les applications *QBee Camera* et *Swisscom Home App* communiquent avec la caméra QBee en clair. Un attaquant sur le réseau local, capable d'observer ce trafic pourra réutiliser les cookies "JSESSIONID, LD\_ID et GC\_ID" afin de s'authentifier auprès de la caméra et en changer les réglages, potentiellement en la désactivant. Cela est possible par exemple par l'activation du mode privacy ou la désactivation de la détection de mouvement.

### QBee Multi-Sensor Camera - Stockage du Mot de Passe

La version pour Android de l'application *QBee Camera* stocke le mot de passe de manière chiffrée dans les réglages<sup>16</sup>, à côté de la clé AES pour son déchiffrement.

### iSmartAlarm - Diagnostique du CubeOne

Les logs de diagnostic de la station de base de iSmartAlarm "CubeOne" sont accessibles de manière non authentifiée. Un attaquant sur le réseau local pourrait donc en obtenir une copie. Les logs contiennent des informations qui peuvent être sensibles, tels que les actions enregistrées par les senseurs.

### iSmartAlarm - Stockage du Mot de Passe

La version pour Android de l'application *iSmartAlarm* stocke le mot de passe de l'utilisateur en clair dans le fichier de configuration.

---

16. com.vestiacom.qbeecamera\_preferences.xml

## 4.5 Accès root et analyse physique

If you try and take a cat apart to see how it works, the first thing you have on your hands is a non-working cat

---

Douglas Adams

Dans cette section les résultats de l'analyse physique du dispositif sont présentés. L'accès aux dispositifs a été obtenu soit via USB, soit via une connexion aux ports de débogage série. Pour la connexion aux ports série un BusPirate en modalité *Bridge UART* a été utilisé au début. Lors de l'analyse du WinkHub, il a été remarqué que le BusPirate n'arrivait pas à transmettre les données envoyées au dispositif (touches pressées) ; la raison n'est pas connue. L'analyse du Arlo et du CubeOne a donc été faite à nouveau en utilisant un adaptateur FTDI USB → Série. Les résultats ci-dessus sont relatifs à cette deuxième analyse.

Pour la station de base du Arlo, le CubeOne et le Wink, les connexions série étaient présentes, cependant les broches de connexion n'étaient pas soudées. Celles-ci ont donc été soudés, et le type des différentes broches (TX,RX,GND,3V) a été identifié à l'aide d'un multimètre. La vitesse de connexion a été trouvée par force brute, en essayant toutes les vitesses standard jusqu'à l'obtention d'un output cohérent.



FIGURE 4.6 – Intérieur de la station de base du Arlo et du CubeOne de iSmartAlarm

### 4.5.1 Nest Camera

La caméra Nest se présente à l'ordinateur lors d'une connexion via USB comme dispositif de stockage de masse (cf. image A.19). Le contenu du stockage est composé par les applications utilisées pour la configuration de la caméra. Aucun fichier log n'est identifié.

### 4.5.2 Nest Protect

Le Nest Protect se présente aussi à l'ordinateur comme stockage USB (cf. image A.20). Lorsque le dispositif est connecté, deux fichiers *debug\_info* et *Technical Info.plist* sont présents. *Technical Info.plist* contient des informations détaillées sur le dispositif, tel que ses adresses MAC ou numéro de série, mais aussi l'ID de l'utilisateur associé. *debug\_info* est par contre un fichier en format binaire. Sa structure n'a pas été comprise et aucune donnée n'a donc pu être extraite.

La collecte de ces logs diagnostiques depuis le Nest Protect présente une limitation importante ; en effet la connexion à l'ordinateur est sporadique. Pendant les essais, le dispositif s'est connecté à l'ordinateur une seule fois, pendant laquelle les fichiers ont été extraits, tout autre essai de connexion, dans les mêmes conditions, ont échoué. A ce jour, les conditions exactes qui permettent la connexion ne sont pas connues.

### 4.5.3 Netgear Arlo

L'accès physique au Netgear Arlo se fait au travers du port série avec protocole UART ; la vitesse de connexion est de 115'200 bps (cf. Annexe A.13).

**Démarrage Standard** Lors d'un démarrage "standard" seul le log du bootloader (CFE) est affiché, dès que le système Linux est chargé l'output termine.

**Console du Bootloader** La combinaison de touches *CTRL+C* permet l'interruption du processus de démarrage ; ainsi une console du bootloader est présentée à l'utilisateur. A partir de cette console il est possible de lancer le système, afficher le contenu du NVRAM, configurer le réseau et faire une copie de la mémoire vers un serveur TFTP. D'autres options sont disponibles, mais ne sont pas pertinentes voir dangereuses dans le cadre de l'analyse (effacer le contenu de la mémoire flash par exemple).

**Extraction des réglages dans le NVRAM** La commande *nvram show* a été utilisée pour afficher tous les réglages sauvegardés dans le NVRAM du Arlo. Dans les réglages on retrouve le nom du réseau WiFi, des clés PSK chiffrées ainsi que des réglages par rapport à d'autres réseaux WiFi inactifs, mot de passe pour UART, serveurs FTP et autres. D'intérêt le réglage *telnethd\_enable=0*, qui sera utiles pour obtenir un accès administrateur.

**Extraction de la mémoire** Avec la commande *save* il a été possible extraire la mémoire du Arlo (0x0000'0000 à 0x1000'0000) vers un serveur TFTP (Figure 4.7).

```
CFE> save 172.21.20.30:arlo.img 0x00000000 0x10000000
268435456 bytes written to 172.21.20.30:arlo.img
*** command status = 0
CFE>
```

FIGURE 4.7 – Arlo - Collecte mémoire vers serveur TFTP

Avec binwalk il n'a pas été possible d'en extraire le firmware, seule une multitude de fichiers compressés sont identifiés, ainsi que des chaînes de caractères et des certificats. L'extraction de ceux-ci engendre une décompression de plusieurs gigaoctets, les détections semblent donc être des faux positifs. Par contre lors d'une analyse du texte contenu dans la première image extraite, la chaîne *eth1\_wpa\_psk=Ird/QQpB7AQjHbREHR3J1YAM0tDRibEz* a été identifiée par *Thibault Soubiran*. Il s'agit du mot de passe du réseau WiFi du Arlo. Aucune autre information pertinente n'a été trouvée.

**Réseau Interne WiFi** Même si trouvé dans la première image, le mot de passe décrit peu avant a été identifié seulement vers la fin du travail. À ce stade une réinitialisation avec les paramètres d'usine de la base du Arlo avait été effectuée en préparation pour le challenge du DFRWS et le nom du réseau avait changé de *NETGEAR76* à *NTGR\_VMB\_1526289002*<sup>17</sup>. Cependant, le mot de passe était toujours valide, et il a été possible de se connecter au réseau privé du Arlo.

Le réseau interne utilise le sous-réseau *172.14.1.0/24*<sup>18</sup> et la base a l'adresse IP 172.14.1.1 ; l'adressage DHCP commence à .100. Depuis ce réseau l'accès à Internet est impossible.

La connexion à *http://172.14.1.1:80/* retourne une page d'erreur 404, d'autres pages communes telles que */admin* ou */dashboard* ont été testées ; de plus une liste de chemins possible a

17. 1526289002 correspond à l'horodatage auquel la base a été initialisée

18. De note, ce réseau ne fait pas partie des réseaux privés selon le RFC-1918, le bloc est de propriété d'AT&T

été extraite depuis les chaînes de caractères dans l'image de mémoire, et utilisée pour rechercher automatiquement de possibles points d'entrée.

Cependant, aucune page valide n'a pu être trouvée. Une recherche par force brute serait possible, mais n'a pas été conduite dans le cadre de cette recherche.

Dans le trafic échangé entre la caméra et la station de base<sup>19</sup> on retrouve les informations techniques sur la caméra sous forme JSON, ainsi que le flux vidéo.

Le réglage *telnetd\_enable* du NVRAM peut être changé pour activer le service telnet et obtenir une console administrateur (cf. A.13). Ainsi il est possible d'avoir accès aux logs de la station de base, dans lesquels on retrouve des informations concernant les évènements de détection de mouvement déclenchés par la caméra.

TABLE 4.8 – Ports ouverts - Station de Base Arlo - Réseau Interne

Port	Service selon nmap	Service identifié
23/tcp	telnet	telnet <sup>20</sup>
67/tcp	dhcps	dhcp server
80/tcp	http	http server
4000/tcp	remoteanything	communication TCP avec cameras

#### 4.5.4 iSmartAlarm CubeOne

Comme pour le Arlo l'accès physique au CubeOne se fait au travers du port série avec protocole UART ; la vitesse de connexion est de 57'600 bps.

**Démarrage Standard** Lors d'un démarrage standard les logs du bootloader (UBoot 3.5.3.0) ainsi que du système Linux (2.6.21) sont affichés à l'écran. Une fois que le système a terminé le démarrage, aucune console interactive n'est présentée, il est seulement possible d'observer les logs de l'application de iSmartAlarm. Les logs du bootloader et du système contiennent des informations sur les adresses de chargement des différents composants. Ces adresses ont été utilisées pour le choix des régions de mémoire à extraire.

19. Capturé avec une WiFi Pineapple et déchiffré avec airdecap-ng

20. Seulement si *telnetd\_enable=1* dans les réglages du NVRAM

TABLE 4.9 – iSmartAlarm - Régions de Mémoire (Taille 8Mio)

Offset	Identifié dans les Logs	Contenu Identifié	N. Image	Nom Fichier
0x0000'0000	-	Système et Texte	1	ismart.img
0x8000'0000	Kernel Load Address	Système	2	ismart_80.img
0xbc00'0000	Boot Image Location	-	3	ismart_bc.img

**Extraction de la mémoire** La pression rapide de la touche 4 au démarrage permet de rentrer dans la console du bootloader. Depuis cette console trois différentes régions de mémoire sont exportées (cf. tableau 4.9 et annexe A.14). La taille choisie des régions correspond à la taille du chip de mémoire SPI présent, notamment 8 Mio.

L’analyse avec binwalk a permis d’extraire à partir de la première et la deuxième image une copie du système (Figure 4.8). Dans la première image, plusieurs régions de mémoire contenant des chaînes de caractères en clair sont présentes. Cependant dans ces régions aucune information pertinente n’a été identifiée.

Le dossier contenant le système extrait de la première et deuxième image ont été comparés entre eux et avec le dossier extrait du firmware téléchargé depuis le cloud de iSmartAlarm avec *diff*; aucune différence n’est remarquée. Une exploration manuelle ne relève non plus aucun fichier autre que les binaires et les configurations par défaut.

Aucun fichier ni fragment pertinent (logs, configuration des senseurs ou autre) n’a été trouvé dans le reste de l’espace des images.

DECIMAL	HEXADECIMAL	DESCRIPTION
3960	0xF78	uImage header, header size: 64 bytes, header CRC: 0x1A086BAF, created: 2013-04-28 09:57:43, image size: 107896 bytes, Data Address: 0x80200000, Entry Point: 0x80200000, data CRC: 0x8FDE24EE, OS: Linux, CPU: MIPS, image type: Standalone Program, compression type: none, image name: "SPI Flash Image"
2184976	0x215710	U-Boot version string, "U-Boot 1.1.3 (Apr 28 2013 - 17:57:40)"
2185568	0x215960	CRC32 polynomial table, little endian
3145728	0x300000	LZMA compressed data, properties: 0x5D, dictionary size: 33554432 bytes, uncompressed size: 6126127 bytes

FIGURE 4.8 – CubeOne - Analyse avec binwalk de ismart\_80.img

### 4.5.5 WinkHub

L'accès physique au WinkHub se fait aussi au moyen du port série avec protocole UART ; la vitesse de connexion est de 115'200 bps.

Il n'y a pas moyen d'interrompre le processus de démarrage avec une simple combinaison de touches. La technique utilisée comporte la mise à la terre du pin 29 de la NAND (Carrier, 2015), effectivement désactivant cette dernière ; lorsque le bootloader n'arrive pas à lire la NAND, il arrête le démarrage et présente à l'utilisateur une console.

L'accès au bootloader ne peut pas être gardé actif : si le délai de démarrage du système n'est pas égal à zéro, lors d'un démarrage où l'on ne connecte pas le port série à un ordinateur, le WinkHub se bloque au stade du bootloader. Au vu du fait qu'il était possible d'obtenir accès root au système, et donc une image complète du système de fichiers, il n'a pas été jugé nécessaire d'essayer d'effectuer des extractions de mémoire depuis le bootloader. Celles-ci auraient posé les mêmes limites que les extractions de mémoire depuis le iSmartAlarm, et il n'aurait pas eu la garantie d'extraire la portion correcte de mémoire.

#### Accès Root et Extraction du Système de Fichiers

L'accès root au WinkHub est obtenu selon la procédure décrite par Carrier (2015), Clone-Num3 (2015), le serveur SSH est activé et configuré pour l'accès avec clé publique. A la suite du redémarrage, il a été possible de se connecter en tant qu'utilisateur root au WinkHub ; la commande *tar* a ensuite été utilisée afin d'extraire une archive complète du système de fichiers.

Les dossiers et fichiers suivants d'intérêt ont été identifiés :

**/database** Il s'agit du dossier contenant les données persistantes.

**/database/apron.db** Base de données SQLite contenant les informations par rapport aux dispositifs ZigBee, Z-Wave et Lutron.

**/database/wpa\_supplicant.conf** Identifiants du réseau WiFi en clair.

**/database/error\_log\*** Log des erreurs, ils pourraient selon les cas contenir des informations intéressantes. Cependant dans le cas d'espèce aucune information pertinente n'est retrouvée.

**database\_default/db\_backup** Dossier de sauvegarde de */database*, on retrouve des anciennes versions de *apron.db*, ainsi que des anciens log d'erreurs.

**/tmp** Dossier temporaire, le contenu est effacé lors du redémarrage du système.

**/tmp/all.log\*** Fichiers logs génériques, comme pour les fichiers d’erreurs des informations pertinentes pourraient être retrouvées si des dispositifs communiquent avec le WinkHub ; cependant dans le cas étudié, on ne retrouve pas des informations d’intérêt.

**/tmp/database/apron\_\*.db** Copies historiques de apron.db

**/var/lib/database/apron.db** Copie additionnelle de apron.db, persiste entre redémarrages.

Ces fichiers contiennent des informations concernant les dispositifs auxquels le WinkHub s’est directement connecté. Dans le cas étudié, les informations retrouvées concernaient seulement deux lampes ZigBee qui avaient été connectées lors de l’utilisation précédente par le Prof. Casey.



## 5 Discussion

Les différents résultats présentés précédemment ont permis de répondre aux questions de recherche initiales.

### Quelles traces sont disponibles sur les dispositifs IdO ?

Entre les dispositifs étudiés, des traces d'intérêt ont été trouvées sur trois dispositifs, notamment le CubeOne de iSmartAlarm, la station de base du Arlo et le WinkHub. Il a pu être observé que le CubeOne garde en mémoire temporaire des logs concernant une partie des évènements enregistrés par les senseurs ainsi que les commandes envoyées par l'utilisateur ; cela permet par exemple de déterminer à quel instant un port a été ouvert, à travers des informations sur le senseur de contact, ou le moment auquel l'alarme a été désactivée. Sur la station de base de Arlo les réglages de la station et des dispositifs connectés, ainsi que les logs de système contenant des informations sur les évènements déclenchés par la caméra sont trouvés. Enfin sur le WinkHub les informations concernant les dispositifs directement connectés au WinkHub sont identifiées, notamment les ampoules ZigBee et leur état ; Cependant sur le WinkHub aucune information n'a été retrouvée concernant les caméras de Arlo et de Nest ainsi que le Nest Protect. Les traces retrouvées sur celui-ci se limitent donc aux dispositifs avec lesquels il a eu une interaction directe.

On observe qu'entre les différentes sources d'informations retrouvées sur les dispositifs, une partie importante ne persiste pas entre les redémarrages. Cela est probablement dû à la faible capacité de stockage des dispositifs, qui donc exploitent la mémoire flash pour le système et les réglages, tandis que les logs et autres informations sont stockés sur un chip de mémoire vive.

## Quelles traces sont générées par leurs applications mobiles ?

L'analyse des traces laissées par les applications pour smartphone a permis de retrouver des traces de nature très différente.

On retrouve dans les caches des applications de Arlo, Wink et Nest des vignettes des images enregistrés par les caméras.

Dans les caches des applications, on peut retrouver des informations sur les évènements qui se sont déroulés ; le cas des données de Nest en est un exemple, dans les fichiers JSON en cache on retrouve des informations par rapport aux évènements enregistrés par la caméra. Les bases de données des applications, dans le cas étudié spécialement celle de iSmartAlarm, contiennent aussi des informations sur les évènements enregistrés par les senseurs ou sur les commandes envoyées par l'utilisateur, ainsi que sur celui-ci.

Un dernier type de trace d'intérêt identifié est celui des identifiants stockés dans les applications. Les applications pour smartphone sont par nature connectées, on ne veut pas rentrer à chaque fois un mot de passe pour utiliser l'application ; il est donc nécessaire de stocker dans les réglages les identifiants pour les services cloud. De l'application de iSmartAlarm qui stocke le mot de passe en clair à l'application de Arlo, qui stocke dans la base de données un token d'accès, des identifiants de connexions sont toujours présents.

## Quelles informations ces traces peuvent fournir à un enquêteur lors de l'investigation d'un crime ?

Lors de l'investigation d'un crime, les traces retrouvées, sur les dispositifs ainsi que sur les smartphones, peuvent être utilisées pour reconstruire l'activité, estimer le nombre des participants, ainsi que leur identité. Les informations temporelles peuvent être aussi inférées sur la base des horodatages enregistrés dans les logs ou dans les bases de données. Par contre, il est important de ne pas se fier aux horodatages de la création des fichiers, car ceux-ci sont dépendant du moment d'utilisation de l'application et non de l'évènement. L'exploitation des identifiants pour l'accès aux services cloud est aussi importante ; armé de ces identifiants, un policier pourrait les utiliser pour se connecter aux services cloud et télécharger l'entièreté des données à disposition dans l'application. Dans le cas des applications comme *iSmartAlarm* ou *QBee Camera* il serait simplement possible d'installer l'application sur un deuxième téléphone et d'effectuer le login. Dans le cas du Netgear Arlo ou d'autres applications utilisant un token, un script pourrait être

écrit permettant de réutiliser le token lors des requêtes (voir l'exemple à l'annexe A.9).

Particulièrement intéressant est le cas de l'application du WinkHub. Celui-ci agissant comme agrégateur d'informations, il est possible de retrouver les logs des dispositifs connectés. Ces logs, comme remarqué lors de l'analyse, sont stockés sur les serveurs de Wink. Il est donc possible d'avoir accès aux évènements enregistrés sans dépendre de la limite imposée par les souscriptions des comptes connectés ; ainsi on pourrait avoir accès à des données sur des évènements auxquelles on n'aurait pas accès depuis l'application officielle du dispositif.

**Exemple** On peut prendre l'exemple fictif d'un cambriolage avec homicide, où l'on a à disposition les traces des dispositifs étudiés dans cette recherche.

Lors de l'arrivée sur la scène, la police remarque que le système d'alarme est composé par des dispositifs de la gamme iSmartAlarm, sur place elle branche la connexion Ethernet de la base à un ordinateur d'analyse et extrait les logs de diagnostic. Avec ceux-ci la police peut déjà obtenir des informations sur les évènements d'ouverture de la porte d'entrée et du déclenchement des senseurs de mouvement, et ainsi inférer une possible plage horaire de commission du crime. Avec les logs ils se rendent compte que l'alarme a été désactivée avant le crime.

Suite à l'obtention du téléphone de la victime, une extraction physique est effectuée. La police pourrait utiliser les vignettes pour obtenir des informations sur le nombre de personnes impliquées dans le cas, ainsi que leur identité ; Dans la base de données de iSmartAlarm on pourra trouver des logs supplémentaires, ainsi le pseudonyme de qui a désactivé l'alarme. Le token d'accès du Netgear Arlo pourra ensuite être utilisé pour télécharger toutes les vidéos enregistrées par la caméra de surveillance, même si les vignettes ne sont pas présentes sur le téléphone.

## **Quelles vulnérabilités sont présentes dans ces dispositifs ?**

L'analyse des dispositifs a aussi mis en évidence plusieurs vulnérabilités sur ces dispositifs, notamment il est possible de désactiver la caméra QBee si l'on intercepte le trafic réseau, ou la possibilité d'obtenir les logs de diagnostic du iSmartAlarm dès qu'on se trouve sur le réseau. Deux vulnérabilités dans les applications pour smartphones ont aussi été identifiées, concernant le stockage des mots de passe dans les réglages.

Dans le cas des vulnérabilités sur les dispositifs, on observe qu'elles sont dues à l'utilisation de protocoles non protégés (HTTP - QBee) ou non standardisés, et mal sécurisées (iSmartAlarm).

Les dispositifs de Nest et Arlo communiquent avec des protocoles sécurisés, standardisés et ouverts (HTTPS, STUN, dTLS ou Weave).

Ainsi ce n'est pas parce qu'un dispositif est un dispositif IdO qu'il ne doit pas être sécurisé. Les protocoles sécurisés existent, mais malheureusement seulement une partie des producteurs décident de les utiliser, pour des raisons de performance (les dispositifs ont une puissance limitée de calcul) ainsi que d'un faux sentiment de sécurité. Contactée par rapport à la vulnérabilité de QBee par exemple, l'équipe de Swisscom a répondu que l'utilisation de HTTP à la place de HTTPS était voulue, car cela permettait d'avoir une interface plus réactive ; de plus, selon eux la vulnérabilité n'avait pas raison d'être, car elle ne pouvait être exploitée que depuis le réseau local, et donc par l'utilisateur même. Cela est loin d'être le cas, un réseau pourrait être facilement attaqué si WPS est activé<sup>1</sup> ou si le mot de passe choisi par l'utilisateur est faible.

## **Leur connaissance peut-elle être utile à des fins forensiques ?**

Si d'une part les vulnérabilités sont intéressantes pour les criminels qui veulent les exploiter, leur connaissance est aussi très importante à des fins forensiques. En effet la connaissance des vulnérabilités d'un dispositif IdO peut aider à expliquer comment les auteurs pourraient avoir exploité le dispositif pour faciliter la commission de leur crime (Vulnérabilité DoS du QBee). D'autre part des vulnérabilités telles que celle des logs de diagnostic du iSmartAlarm, ou les mots de passe sur les smartphones peuvent être utiles pour acquérir des informations supplémentaires, soit directement depuis le dispositif, soit depuis les services cloud connectés.

On peut donc observer que les dispositifs IdO génèrent une grande quantité de données. On retrouve des traces dans les logs, les bases de données, les fichiers en cache et les réglages des applications. La plupart des traces observées se retrouvent sur les appareils mobiles, ce qui n'est pas surprenant, étant donné que ce sont les applications mobiles qui "consomment" les données générées par les dispositifs IdO. Cependant, des traces ont pu être extraites des dispositifs IdO. Au vu de la faible capacité de stockage de ces dispositifs, la majeure partie des traces liées aux évènements réside dans la mémoire temporaire, et est donc perdue lors d'un redémarrage. L'accès à ces données demande donc un accès au dispositif via une vulnérabilité, afin de ne pas devoir le redémarrer. Les données des réglages des dispositifs sont aussi intéressantes, comme

---

1. « Wifi Protected Setup vulnerable », 2012.

observé dans l'extraction du WinkHub, les réglages peuvent inclure les identifiants pour l'accès au WiFi, qui pourraient être utilisés pour se connecter à d'autres sources de données sur le réseau.

La méthodologie appliquée répond bien aux besoins d'analyse, l'étude initial du trafic réseau permet de choisir comment cibler les passages suivants, et a été cruciale pour la découverte de deux des vulnérabilités. Cependant, la méthodologie présentée ne constitue pas une méthodologie linéaire, lors des différentes étapes d'analyse il a été nécessaire de revoir les données récoltées en amont, ou en récolter des nouvelles afin de pouvoir être le plus exhaustif possible. Par exemple, lors de l'analyse des applications sur smartphone, il a été remarqué que seule l'application de iSmartAlarm avait un système d'envoi des informations diagnostiques. Il a donc fallu retourner analyser de manière approfondie l'échange sur le réseau entre le téléphone et le CubeOne, ce qui a permis de dévoiler une des vulnérabilités, ainsi qu'obtenir l'accès aux informations diagnostiques.

Les traces concernant les dispositifs IdO et leurs applications peuvent donc être de forte aide dans le cadre d'une investigation. Cependant selon le dispositif ces traces sont très variables. Il est donc nécessaire lors d'une investigation où des nouveaux dispositifs sont concernés de pouvoir être capable d'en extraire et d'analyser les traces, que cela soit depuis le dispositif même, ou les données de l'application.

Pour cela, la figure d'un *Digital/Multimedia Forensic Advisor*, ayant des connaissances étendues et transversales dans les différents domaines traités dans ce travail est d'importance vitale pour une police qui ne veut pas juste limiter les investigations aux traces physiques et à celles numériques que les producteurs des logiciels d'analyse considèrent comme "standard". De manière similaire au *Forensic Advisor* tel que décrit par Bitzer et al. (2018) son rôle sera de conseiller les investigateurs afin de définir les priorités d'analyse ; cependant il aura aussi le rôle plus pratique de développement et mise à jour d'outils d'analyse adaptés aux dispositifs que la police pourrait rencontrer. Lorsque les connaissances nécessaires pour l'analyse des dispositifs lui échappent, il devrait pouvoir se coordonner avec des spécialistes du domaine.

Si dans ce travail on se limitait à analyser les données extraites de manière automatique par les logiciels, la majorité des traces pertinentes n'auraient pas pu être obtenues. L'analyse d'une partie des données pourrait se faire de manière manuelle, mais la taille des données et leur encodage rendraient ce travail extrêmement long. C'est pourquoi il est important de pouvoir développer des outils ou des plug-ins pour extraire, de manière automatisée, les données des dispositifs IdO auxquels on fait face.

Comme présenté dans la figure 5.1, cette méthodologie permet, pour des nouveaux dispositifs, de répondre aux besoins de la première étape du processus forensique, notamment l'étape de recherche/détection (*survey*). Lors d'une enquête l'investigateur saura donc quels types de traces pourraient être retrouvées et où, ce qui va pouvoir orienter les choix d'investigation. Les outils ainsi que les connaissances développées lors de l'analyse pourront ensuite être réutilisés dans les étapes de préservation, examination et analyse.

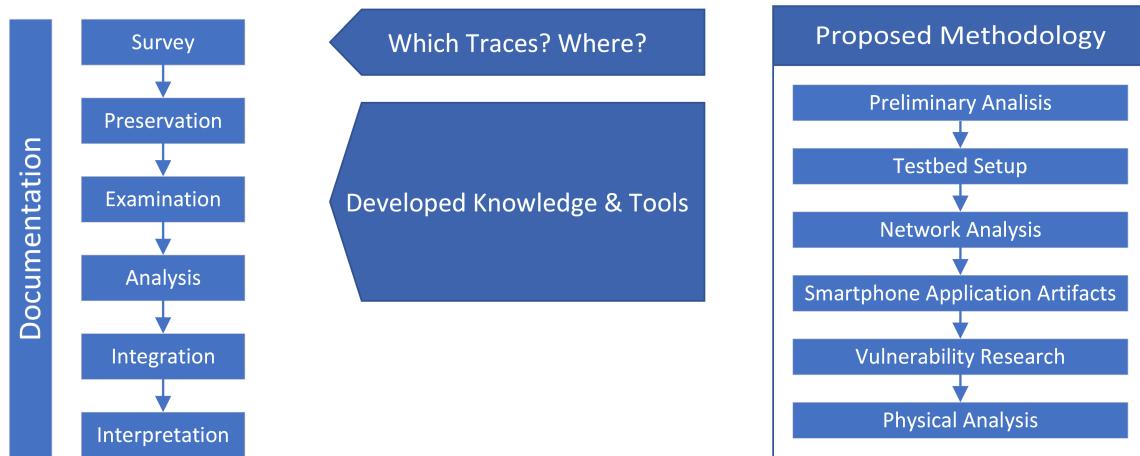


FIGURE 5.1 – Application de la méthodologie proposée dans l'investigation numérique des IdO

## 5.1 Limites & Développements Futurs

**Extraction Physique** L'extraction physique des images de mémoire, faite depuis le bootloader, présente comme problématique le choix des plages de mémoire correctes à extraire ; à ce stade en effet la mémoire est présentée de manière brute, et les plages correspondant aux différents dispositifs ne sont pas toujours bien documentées. Dans ce travail les extractions ont été faites sur la base d'un mélange d'informations obtenues depuis la documentation des microprocesseurs, ainsi que des informations sur les adresses de chargement des images dans les log de boot.

De plus dans les dispositifs étudiés la mémoire flash était composée par des chips de type NAND ou SPI, pour lesquels nous n'avions ni l'expertise ni les instruments pour effectuer des images par "Chip-Off".

A l'avenir il pourrait être intéressant d'étudier de manière plus approfondie ce type d'extraction.

**Applications Smartphone - Evènements en arrière-plan** Une limite importante de l’analyse des artefacts générés par les applications sur les smartphones est leur création. En effet les traces concernant des évènements, que ça soit des logs, des vignettes, ou des entrées dans les bases de données sont mis à jour lors de l’utilisation de l’application. Lors de l’analyse d’un téléphone bloqué, il ne sera pas possible d’utiliser l’application ; par conséquent les données et les fichiers de l’application retrouvés correspondront à ceux mis à jour lors de la dernière utilisation de l’application.

L’utilisation des identifiants stockés dans les réglages (mots de passe ou token) peut être une alternative. Avec ceux-ci il serait possible d’obtenir directement depuis le service cloud les informations d’intérêt. Cependant, cela pourrait poser un problème de légalité selon le pays, nécessitant avant leur utilisation, l’obtention d’un mandat supplémentaire, même si l’on n’a pas besoin d’effectuer une demande formelle au fournisseur de service.

**Charge de travail** Si d’une part cette méthodologie permet d’avoir une vue assez complète des traces disponibles sur un dispositif, cela n’est que grâce à un nombre très élevé de temps investi dans l’analyse et décodage des traces présentes ainsi que dans la recherche de possibles techniques d’extraction pour le dispositif en cause. La demande de temps élevé est donc une limitation importante de ce travail.

Dans le cas où un forensicien se trouve dans une investigation face à des dispositifs qui n’ont pas pu être préalablement étudiés, il est conseillé de procéder comme ci-dessous :

- Prélèvement du dispositif IdO
- Extraction du/des téléphones qu’y sont vraisemblablement liés et analyse des artefacts des applications.
- Si les connaissances et le matériel sont disponibles : extraction d’une image physique du dispositif.

**Protocoles** L’analyse des communications des dispositifs s’est concentrée sur les données transmises par WiFi ou réseau câblé. Cependant ces dispositifs utilisent d’autres protocoles tels que 801.15.4/ZigBee (p.ex. les dispositifs Nest) ou RF (iSmartAlarm). Ceux-ci n’ont pas été étudiés à cause d’un manque d’expertise et de matériel, mais leur analyse pourrait aussi être d’intérêt, surtout pour la connaissance de vulnérabilités potentielles, comme montré par Pidhirney (2017).



## 6 Conclusion

Ce travail a pour but d'identifier les traces générées par les dispositifs IdO et leurs applications pour smartphone. Dans un même temps, la sécurité des dispositifs a été évaluée à la recherche de vulnérabilités exploitables par un criminel, mais aussi par la police lors d'une investigation. Une méthodologie d'analyse transversale, comprenant l'analyse du réseau et des applications pour smartphones, la recherche de vulnérabilités et l'analyse physique des dispositifs, était proposée et a été appliquée à l'analyse.

Ce travail a permis de démontrer que, si la majorité des traces sont constituées par les artefacts des applications pour smartphones, des traces pertinentes sont également présentes sur les dispositifs. Celles-ci peuvent être obtenues soit par l'exploitation de vulnérabilités, soit avec des extractions physiques. La méthodologie choisie s'est révélée adéquate pour l'analyse de ces dispositifs, et permet de couvrir de manière étendue l'ensemble des types de traces pouvant être détectées.

Concernant la sécurité, celle-ci est devenue une préoccupation majeure pour de nombreux fabricants, tandis que pour certains constructeurs cet aspect n'est que d'une importance secondaire. Cependant les vulnérabilités découvertes ne sont pas seulement utiles à des potentiels criminels, mais peuvent être exploitées aussi à des fins forensiques.

Afin de mettre en œuvre cette méthodologie, on remarque l'importance de la figure du *Digital/Multimedia Forensic Advisor* ayant des connaissances transversales dans les domaines de ce travail. Il pourra ainsi conseiller les investigateurs sur les priorités de recherche et s'occuper de l'analyse de nouveaux dispositifs auxquels une enquête pourrait être confrontée si nécessaire en collaborant avec des experts du domaine.

A l'avenir, il serait intéressant d'étudier de manière plus approfondie les dispositifs d'un point de vue de l'analyse physique. Les traces présentes sur les dispositifs iOS, qui en raison du temps n'ont pu être analysés de manière approfondie pourraient elles-aussi constituer des pistes d'analyse future.



# Bibliographie

- Akatyev, N. & James, J. I. (2017). Evidence identification in IoT networks based on threat assessment. *Future Generation Computer Systems*. doi :doi.org/10.1016/j.future.2017.10.012
- Alexander-Bown, S. (2013–). Secure-preferences. Récupérée à partir de <https://github.com/scottyab/secure-preferences>
- Amar, Y., Haddadi, H., Mortier, R., Brown, A., Colley, J. A. & Crabtree, A. (2018). An Analysis of Home IoT Network Traffic and Behaviour. *CoRR, abs/1803.05368*. arXiv : 1803.05368. Récupérée à partir de <http://arxiv.org/abs/1803.05368>
- AOL. (2012–). Moloch (Version 1.1.1). Récupérée à partir de <https://molo.ch/>
- Ashton, K. (2009, juin 22). That 'Internet of Things' Thing. Récupérée 1 avril 2018, à partir de <http://www.rfidjournal.com/articles/view?4986>
- Barnard-Wills, D., Marinos, L. & Portesi, S. (2014, décembre 1). *Threat Landscape and Good Practice Guide for Smart Home and Converged Media*. European Union Agency for Network et Information Security (ENISA). doi :10.2824/33134
- Bitzer, S., Heudt, L., Barret, A., George, L., Dijk, K. V., Gason, F. & Renard, B. (2018). The introduction of forensic advisors in Belgium and their role in the criminal justice system. *Science & Justice*, 58(3), 177-184. doi :10.1016/j.scijus.2017.11.002
- Carrier, M. (2015, août 25). Hacking the Winkhub. Récupérée à partir de <https://mattcarrier.com/post/hacking-the-winkhub-part-1/#Resources>
- Ching, Y. (2017, novembre 30). Public Disclosure : Firmware Vulnerabilities in iSmartAlarm CubeOne. Récupérée 25 juin 2018, à partir de <https://poppopretn.com/2017/11/30/public-disclosure-firmware-vulnerabilities-in-ismartalarm-cubeone/>

- Chung, H., Park, J. & Lee, S. (2017). Digital forensic approaches for Amazon Alexa ecosystem. *Digital Investigation*, 22, S15-S25. doi :10.1016/j.diin.2017.06.010
- Cisco. (p. d.). Cisco IoT Threat Defense. Récupérée 6 juillet 2018, à partir de <https://www.cisco.com/c/en/us/solutions/security/iot-threat-defense/index.html>
- Cisco. (2014a, septembre 26). Network Address Translation on a Stick. Récupérée 2 juillet 2018, à partir de <https://www.cisco.com/c/en/us/support/docs/ip/network-address-translation-nat/6505-nat-on-stick.html>
- Cisco. (2014b, juin 4). *The Internet of Things Reference Model*. Cisco. Récupérée à partir de [http://cdn.iotwf.com/resources/71/IoT\\_Reference\\_Model\\_White\\_Paper\\_June\\_4\\_2014.pdf](http://cdn.iotwf.com/resources/71/IoT_Reference_Model_White_Paper_June_4_2014.pdf)
- CloneNum3. (2015, mai 12). Rooting ANY firmware level... the hard way. Récupérée 24 juin 2018, à partir de <http://www.rootwink.com/viewtopic.php?f=6&t=67>
- Cluster of European Research Project on the Internet of Things. (2010, avril 12). *Vision and challenges for realising the Internet of things*. Cluster of European Research Projects on the Internet of Things, European Commision. Récupérée à partir de <https://publications.europa.eu/en/publication-detail/-/publication/ed079554-72c3-4b4e-98f3-34d2780c28fc/language-en>
- Combs, G. & Contributors. (2006–). Wireshark (Version 2.4.3). Récupérée à partir de <https://www.wireshark.org>
- Cortesi, A., Hils, M., Kriechbaumer, T. & Contributors. (2010–). mitmproxy (Version 3.3.0). Récupérée à partir de <https://mitmproxy.org/>
- Dorsemaine, B., Gaulier, J.-P., Wary, J.-P., Kheir, N. & Urien, P. (2016). A new approach to investigate IoT threats based on a four layer model. In *2016 13th Int. Conf. New Technol. Distrib. Syst.* 18 juillet 2016 (p. 1-6). doi :10.1109/NOTERE.2016.7745830
- Elastic. (2010–). Elasticsearch (Version 6.2.3). Récupérée à partir de <https://www.elastic.co/products/elasticsearch>
- Elastic. (2014–). Kibana (Version 6.2.3). Récupérée à partir de <https://www.elastic.co/products/kibana>

- Evans, D. (2011, avril). *The Internet of Things How the Next Evolution of the Internet Is Changing Everything*. Cisco Internet Business Solutions Group (IBSG). Récupérée à partir de [https://www.cisco.com/c/dam/en\\_us/about/ac79/docs/innov/IoT\\_IBSG\\_0411FINAL.pdf](https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf)
- FCC. (2014). FCC ID 2ACAJ-WINK22 home automation gateway by Wink Labs, Inc. Récupérée 30 juin 2018, à partir de <https://fccid.io/2ACAJ-WINK22>
- FCC. (2015a). FCC ID SENDWS3A Contact Sensor by iSmart Alarm, Inc. Récupérée 30 juin 2018, à partir de <https://fccid.io/SENDWS3A>
- FCC. (2015b). FCC ID SENIPU3A CubeOne by iSmart Alarm, Inc. Récupérée 30 juin 2018, à partir de <https://fccid.io/SENIPU3A>
- FCC. (2015c). FCC ID SENPIR3A Motion sensor by iSmart Alarm, Inc. Récupérée 30 juin 2018, à partir de <https://fccid.io/SENPIR3A>
- FCC. (2015d). FCC ID SENRC3A Remote tag by iSmart Alarm, Inc. Récupérée 30 juin 2018, à partir de <https://fccid.io/SENRC3A>
- FCC. (2015e). FCC ID ZQANC11 device by Nest Labs Inc. Récupérée 30 juin 2018, à partir de <https://fccid.io/ZQANC11>
- FCC. (2015f). FCC ID ZQAS30 Wireless Protect by Nest Labs Inc. Récupérée 30 juin 2018, à partir de <https://fccid.io/ZQAS30>
- FCC. (2016a). FCC ID H8N-QBMSCFXL Multi - Sensor Camera by Askey Computer Corp. Récupérée 30 juin 2018, à partir de <https://fccid.io/H8N-QBMSCFXL>
- FCC. (2016b). FCC ID PY316200349 Arlo Pro by Netgear Incorporated. Récupérée 30 juin 2018, à partir de <https://fccid.io/PY316200349>
- FCC. (2016c). FCC ID PY316200350 arlo Base Station by Netgear Incorporated. Récupérée 30 juin 2018, à partir de <https://fccid.io/PY316200350>
- Holub, A. & Colford, P. (2017, janvier 5). The Future is Here – Assaulting the Internet with Mirai. Récupérée à partir de <https://umbrella.cisco.com/blog/blog/2017/01/05/future-assaulting-internet-mirai/>

- Hyde, J. & Moran, B. (2017). Alexa, are you Skynet ? In *SANS DFIR Summit 2017*, 22-23 juin 2017. Récupérée à partir de <https://www.sans.org/summit-archives/file/summit-archive-1498230402.pdf>
- iFixit. (2018, janvier 26). Nest Cam Motherboard Replacement. Récupérée 30 juin 2018, à partir de <https://www.ifixit.com/Guide/Nest+Cam+Motherboard+Replacement/61721>
- Kebande, V. R. & Ray, I. (2016). A Generic Digital Forensic Investigation Framework for Internet of Things (IoT). In *2016 IEEE 4th Int. Conf. Futur. Internet Things Cloud*, 22-24 août 2016 (p. 356-362). doi :10.1109/FiCloud.2016.57
- Kolias, C., Kambourakis, G., Stavrou, A. & Voas, J. (2017). DDoS in the IoT : Mirai and Other Botnets. *Computer (Long. Beach. Calif)*. 50(7), 80-84. doi :10.1109/MC.2017.201
- Loung, T. (2018, juin 23). Thermostats, Locks and Lights : Digital Tools of Domestic Abuse. *New York Times*. Récupérée à partir de <https://www.nytimes.com/2018/06/23/technology/smart-home-devices-domestic-abuse.html>
- Lyon, G. & Contributors. (1997–). Nmap (Version 7.60). Récupérée à partir de <https://nmap.org/>
- Micaksica. (2017, janvier 2). Exploring the Amazon Echo Dot, Part 1 : Intercepting firmware updates. Récupérée à partir de <https://medium.com/@micaksica/exploring-the-amazon-echo-dot-part-1-intercepting-firmware-updates-c7e0f9408b59>
- NETRESEC. (2007–). NetworkMiner (Version 2.2). Récupérée à partir de <http://www.netresec.com/?page=CapLoader>
- NewSky Security. (2016a, septembre 14). Brute Force Vulnerability in Netgear ARLO. Récupérée 25 juin 2018, à partir de <https://blog.newskysecurity.com/brute-force-vulnerability-in-netgear-arlo-f561c3bc1f3d>
- NewSky Security. (2016b, septembre 1). Factory Reset Vulnerability in Netgear ARLO. Récupérée 25 juin 2018, à partir de <https://blog.newskysecurity.com/factory-reset-vulnerability-in-netgear-arlo-414c68b17cb6>
- Open Web Application Security Project. (2016, mai 18). Top IoT Vulnerabilities. Récupérée 17 juin 2018, à partir de [https://www.owasp.org/index.php/Top\\_IoT\\_Vulnerabilities](https://www.owasp.org/index.php/Top_IoT_Vulnerabilities)

- Open Web Application Security Project. (2018, mars 9). IoT Attack Surface Areas Project. Récupérée 17 juin 2018, à partir de [https://www.owasp.org/index.php/OWASP\\_Internet\\_of\\_Things\\_Project#tab=IoT\\_Attack\\_Surface\\_Areas](https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project#tab=IoT_Attack_Surface_Areas)
- Oriwoh, E., Jazani, D., Epiphaniou, G. & Sant, P. (2013). Internet of Things Forensics : Challenges and Approaches. In *Proc. 9th IEEE Int. Conf. Collab. Comput. Networking, Appl. Work.* 20-23 octobre 2013. doi :10.4108/icst.collaboratecom.2013.254159
- Pidhirney, D. (2017, mars 4). DIY Smart Home Security ? Meh.. Récupérée 23 juin 2018, à partir de <http://blog.seekintoo.com/diy-smart-home-security-meh.html>
- Pollitt, M., Casey, E., Jaquet-Chiffelle, D.-O. & Gladyshev, P. (2018, janvier). *A Framework for Harmonizing Forensic Science Practices and Digital/Multimedia Evidence*. The Organization of Scientific Area Committees for Forensic Science (OSAC). doi :10.29325/OSAC.TS.0002
- Poole, N. (2013). Nest Protect Teardown. Récupérée 30 juin 2018, à partir de <https://learn.sparkfun.com/tutorials/nest-protect-teardown>
- Rajewski, J. T. (2016). Internet of Things Forensics. In *Enfuse 2016*, 23 mai 2016-26 mai 2015. Récupérée à partir de [https://www.jonrajewski.com/data/Presentations/EnFuse2016/Enfuse\\_2016\\_Internet\\_of%20Things\\_Rajewski.pdf](https://www.jonrajewski.com/data/Presentations/EnFuse2016/Enfuse_2016_Internet_of%20Things_Rajewski.pdf)
- Rajewski, J. T. (2017). Internet of Things Forensics. In *Enfuse 2017*, 22-25 mai 2017. Récupérée à partir de [https://www.jonrajewski.com/wp-content/uploads//2017/07/Enfuse\\_2017\\_Rajewski\\_IOT\\_Forensics.pdf](https://www.jonrajewski.com/wp-content/uploads//2017/07/Enfuse_2017_Rajewski_IOT_Forensics.pdf)
- Roethlisberger, D. & Contributors. (2009–). SSLsplit (Version 0.5.2). Récupérée à partir de <https://www.roe.ch/SSLsplit>
- Sachidananda, V., Siboni, S., Shabtai, A., Toh, J., Bhairav, S. & Elovici, Y. (2017). Let the Cat Out of the Bag : A Holistic Approach Towards Security Analysis of the Internet of Things. In *Proceedings of the 3rd ACM International Workshop on IoT Privacy, Trust, and Security*, 2 avril 2017 (p. 3-10). IoTPTS '17. doi :10.1145/3055245.3055251
- Shnайдман, И. (2017, juillet 10). Burglar and Hacker – When Physical Security Is Compromised by IoT Vulnerabilities. Récupérée 25 juin 2018, à partir de <https://dojo.bullguard.com/dojo>

by-bullguard/blog/burglar-hacker-when-a-physical-security-is-compromised-by-iot-vulnerabilities/

Skylot. (2013–). JADX (Version 0.7.1). Récupérée à partir de <https://github.com/skylot/jadx>

The Tcpdump Group. (1999–). tcpdump (Version 4.9.2). Récupérée à partir de <https://www.tcpdump.org>

Wifi Protected Setup vulnerable. (2012). *Computer Fraud & Security*, 2012(1), 3. doi :10.1016/S1361-3723(12)70004-0

Zawoad, S. & Hasan, R. (2015). FAIoT : Towards Building a Forensics Aware Eco System for the Internet of Things. In *2015 IEEE Int. Conf. Serv. Comput.* 27 juin-2 juillet 2015 (p. 279-284). doi :10.1109/SCC.2015.46

# A Annexes

## A.1 Analyse Préliminaire des Dispositifs

Les informations présentées dans le tableau A.1 correspondent à ce qui a été observé sur la documentation du FCC (2014, 2015a, 2015b, 2015c, 2015d, 2015e, 2015f, 2016a, 2016b, 2016c) ainsi que sur d'autres sources sur Internet (iFixit, 2018 ; Poole, 2013).

## A.2 Raspberry Pi

Dans le cadre de ce travail, un Raspberry Pi 3 a été principalement utilisé. Les logiciels nécessaires à fournir les services réseau, ainsi que pour la collecte de données, ont été testés d'abord dans une machine virtuelle Debian, pour simuler l'environnement du Raspberry (la distribution utilisée, Raspbian, est dérivée de Debian) et s'assurer du bon fonctionnement de la configuration. Une interface très simple (cf. figure A.1) a été développée pour monitorer l'état des dispositifs ainsi que permettre une utilisation aisée du WPS.

Pour garantir la reproductibilité de la configuration sur d'autres dispositifs des *Playbooks* pour Ansible<sup>1</sup> sont disponibles<sup>2</sup>. Cependant certains éléments, tels que mitmproxy ou SSLSplit, sont à compiler directement depuis le code source, car les versions disponibles dans le repository de Raspbian ne sont pas à jour.

Si le Raspberry Pi 3 possède un chip WiFi intégré, pour un problème de compatibilité il n'est pas possible d'ajouter des nouveaux dispositifs avec WPS. Afin de pouvoir connecter la caméra QBee au réseau, il a donc été nécessaire utiliser un Raspberry Pi 2 avec un adaptateur WiFi USB. Une fois le dispositif connecté au réseau, il est possible d'utiliser le Raspberry Pi 3 à nouveau.

---

1. <https://www.ansible.com/>

2. <https://github.com/fservida/pi-pineapple>

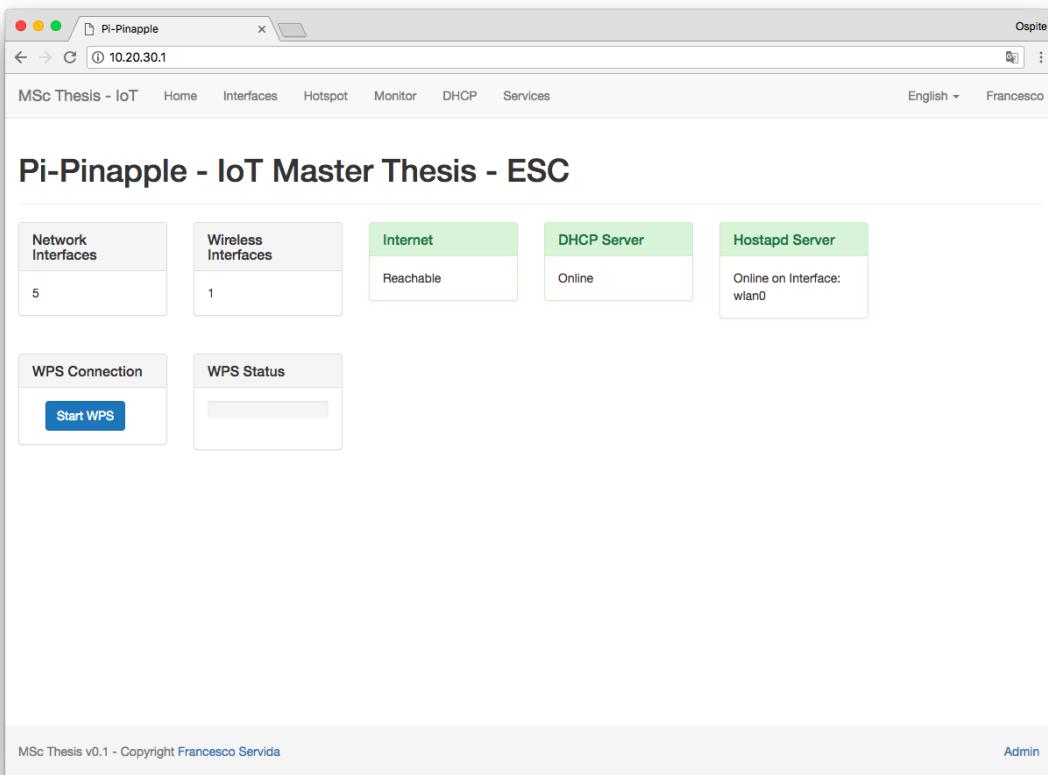


FIGURE A.1 – Raspberry Pi - Page d’administration (Pi-Pineapple)

**TFTP** Lors de l’extraction via TFTP de l’image depuis le Arlo pour le challenge du DFRWS, il a été remarqué que la connexion vers le serveur TFTP à distance échouait. Un serveur TFTP a donc été installé en local sur la Raspberry Pi. L’image en local a pris beaucoup moins de temps que celle à distance (~5 minutes contre ~3 heures), cela même si à distance il n’y avait pas de limitation au niveau de la bande passante.

## A.3 QBee - Connexions Réseau

**Connexions "Keep Alive"** Les connexions sur le port 7402 sont gardées actives de manière continue, dès qu’une connexion est fermée, une autre est ouverte peu après. Elles consistent dans un échange continu de paquets TCP avec 40 octets des données sous le format "970a002000000000 + 32 octets". Ce pattern est interrompu de temps en temps par des paquets avec une structure de données différentes "970a00 + 1 byte variable + 00000000 + nombre variable de octets", ou encore par des paquets du serveur qui ne sont pas en réponse à un de la caméra. À la suite de ces interruptions, les données envoyées peuvent varier fortement, comme on le voit dans l’image A.3.

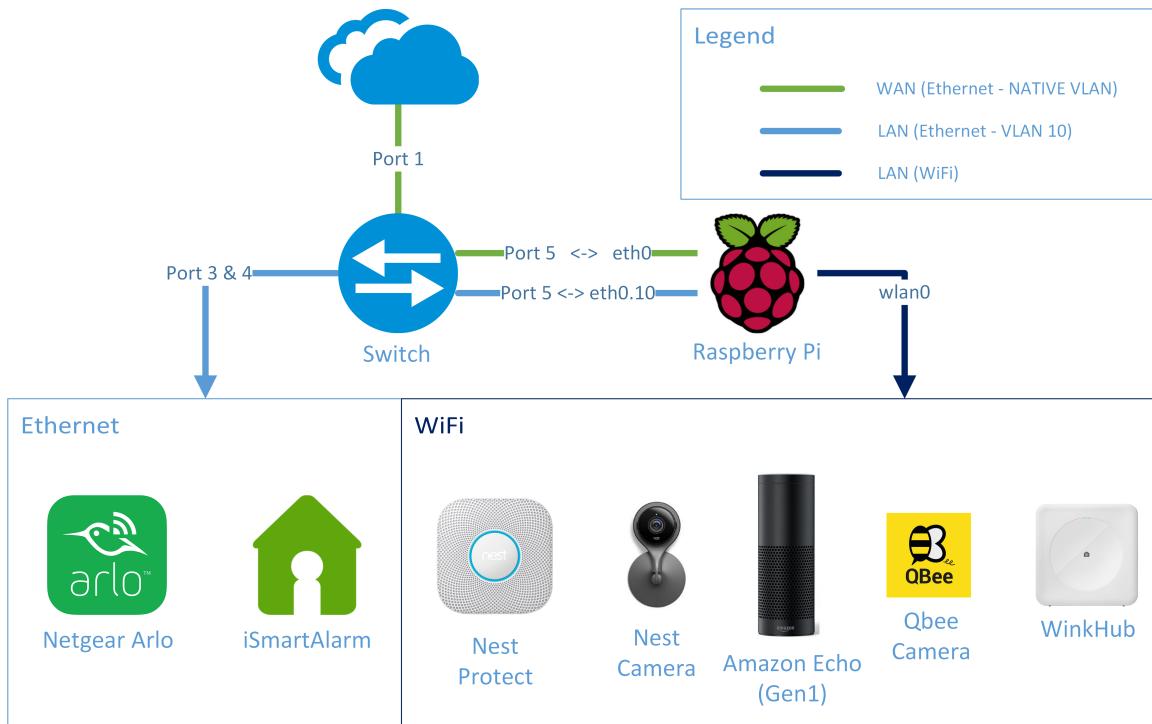


FIGURE A.2 – Topologie physique du réseau

Il s'agit donc en toute probabilité de la connexion keepalive entre la caméra et le serveur, qui permet donc au serveur d'envoyer des données ou commandes à une caméra derrière un pare-feu.

**UPnP** Avec Linux IGD il a été possible de voir que la caméra QBee essaie d'ouvrir une porte dans le pare-feu via UPnP (cf. figure A.4). La porte ouverte sur le pare-feu, ainsi que celle sur le dispositif sont comprises entre 15800 et 15810, plusieurs tentatives avec numéro de port croissant sont faites. Cependant toutes les tentatives échouent ; en effet si la porte peut être ouverte dans le pare-feu du Raspberry-Pi, elle reste fermée dans le pare-feu de l'université. L'ouverture étant échouée, le dispositif n'ouvre pas le port en local, et aucun trafic ne peut être observé. On ne peut donc pas savoir quel type de trafic et quel niveau de protection sont présents. Afin de tester correctement ce type de trafic, il serait nécessaire de modifier la mise en place pour que le routeur (ici le Raspberry Pi) soit directement connecté à Internet.

Il s'agit cependant du seul dispositif ouvrant des portes dans le pare-feu, et qui pourraient donc être attaquables depuis l'extérieur.

```

970a0020000000000014de9803d4ea5fd2663c575f03e7643a5b8aaaf770e3360697d5a0fa3404e56cc
970a00200000000000c1b8b7080fcfe63be96f853cd8422ddeeda084cfe899396293b85d047752600
970a002000000000005d2ff448b3af2f11fc61c88f9edd6d360b9b9867b399f58e350dc3d190e75c94
970a00200000000007a474b9edd2d2492041991d7812eb62837601129927780a20eec22ee736d90b8
970a00200000000000203f14867c6be57f5249b94e2d167d2201cbf037bda9925f9aac0909272eff5c
970a00200000000000d2ccb3371767edb75ff5a600b601f16eafea47143c9f384bcbed3d8d0ec531af
970a002000000000a69f26c3c57d3b5f03f05e49caa3928d929b8adf7a8caf16f8025fd335d7f73
970a00200000000000d18e41cdef095c8de6c9c3c60be4061f4577bae9ebd4206fccd3b8078e9ddef4
970a002000000000828a5001dc99bb2ac7d1a7e930d7babe4a0aead6c589e7d169c0f8157dbf60fa
970a00800000000000aaa3c84648bda1f284b6797aa9974b1c871af4b89a4781783004c920960f7399d5cf37aa70af6ee4ee36
714a96cdc54033375b2d906fe16741d28e3bb6e1e3336d1eb1a2ae64110676dc7595bdf236ba17ae66362eb5c5df701
970a0020000000000087ad21e43a08adfeceb8e2d4d74c5661a0a88ee3f443e66861f0aee344304b95
970a10200000000003d361367fde40ee615e5a0d3fd27811f9e4a9b77047294d7a58f487c3c50be7c1d8162df29a3339d314438:
4a726afcc6559bda87790031cb8e4d92072f67d8fdae5e0831a386caf9409719127bdd6565b57da09d285481c19d537b0b75d9:
a518faabc7a2110667ad85e4e5f94b2a8242e7a599015434af331bfa47205c7c1f37507abcc1cbc9dc15f958ba4a72f6b47a89t
9ca329da50a01ccda6ba34664def0d96701c2ec324add44f8db195b3554b963389f55f6e5ce6c7f2276de4be859e8fe36cd72b:
a42b7c6cc52f6be1661bb0b670b3acc9561b67825c336a49b92d99bb3fc57b422949a7d757707bc359962a01b4124924327a4:
0c4a8ea665cfc992d5a4193ea11b9205e9f9f696d4cee951538fef9f77be67754cc48d77a310e50fcdec410891d8fb982510:
660267c97bb90c8e54cef7c2e0fd9a9f65cd7e82f14f3f7197816e7ff43507d1a10a4af8178efa9f6c6438ce22d32025816bcb:

```

FIGURE A.3 – Extrait des connexions d'une session "Keep-Alive" de la caméra QBee

```

20:17:10 raspberrypi systemd[1]: Started UPNP Daemon (IGD).
20:17:10 raspberrypi upnpd[704]: upnpd[704]: 0x76fe74b0 UPnP SDK Successfully Initialized.
20:17:10 raspberrypi upnpd[704]: 0x76fe74b0 UPnP SDK Successfully Initialized.
20:17:10 raspberrypi upnpd[704]: upnpd[704]: 0x76fe74b0 Successfully set the Web Server Root Directory.
20:17:10 raspberrypi upnpd[704]: 0x76fe74b0 Successfully set the Web Server Root Directory.
20:17:10 raspberrypi upnpd[704]: upnpd[704]: 0x76fe74b0 IGD root device successfully registered.
20:17:10 raspberrypi upnpd[704]: upnpd[704]: 0x76fe74b0 IGD root device successfully registered.
20:17:11 raspberrypi upnpd[704]: upnpd[704]: 0x76fe74b0 Advertisements Sent. Listening for requests ...
20:17:11 raspberrypi upnpd[704]: upnpd[704]: 0x76fe74b0 Advertisements Sent. Listening for requests ...
23:48:04 raspberrypi upnpd[704]: upnpd[704]: 0x735d3470 AddPortMap: DevUDN: uuid:75802409-bccb-40e7-8e6c-
23:48:04 raspberrypi upnpd[704]: 0x735d3470 AddPortMap: DevUDN: uuid:75802409-bccb-40e7-8e6c-fa095ecce13e
23:48:08 raspberrypi upnpd[704]: upnpd[704]: 0x735d3470 DeletePortMap: Proto:TCP Port:15800
23:48:08 raspberrypi upnpd[704]: 0x735d3470 DeletePortMap: Proto:TCP Port:15800
23:48:10 raspberrypi upnpd[704]: upnpd[704]: 0x735d3470 AddPortMap: DevUDN: uuid:75802409-bccb-40e7-8e6c-
23:48:10 raspberrypi upnpd[704]: 0x735d3470 AddPortMap: DevUDN: uuid:75802409-bccb-40e7-8e6c-fa095ecce13e
23:48:33 raspberrypi upnpd[704]: upnpd[704]: 0x735d3470 DeletePortMap: Proto:TCP Port:15801
23:48:33 raspberrypi upnpd[704]: upnpd[704]: 0x735d3470 DeletePortMap: Proto:TCP Port:15801
23:48:35 raspberrypi upnpd[704]: upnpd[704]: 0x735d3470 AddPortMap: DevUDN: uuid:75802409-bccb-40e7-8e6c-
23:48:35 raspberrypi upnpd[704]: upnpd[704]: 0x735d3470 AddPortMap: DevUDN: uuid:75802409-bccb-40e7-8e6c-fa095ecce13e
23:48:40 raspberrypi upnpd[704]: upnpd[704]: 0x735d3470 DeletePortMap: Proto:TCP Port:15802

```

FIGURE A.4 – QBee - Tentatives d'ouverture d'une porte dans le pare-feu (Logs de Linux IGD)

## A.4 Arlo - Connexion locale avec Smartphone

Le trafic sur le réseau local est fait directement entre le téléphone et la station de base en utilisant le protocole dTLS. Au vu du fait que le Arlo ne présente pas de ports ouverts, l'établissement de la connexion doit suivre une procédure particulière. Notamment la station de base du Arlo et l'application pour smartphone utilisent le protocole STUN afin d'annoncer l'un à l'autre une porte ouverte sur laquelle transmettre du trafic UDP. L'analyse du trafic avec Wireshark a permis d'inférer le fonctionnement du protocole utilisé, cela est présenté à la figure A.5.

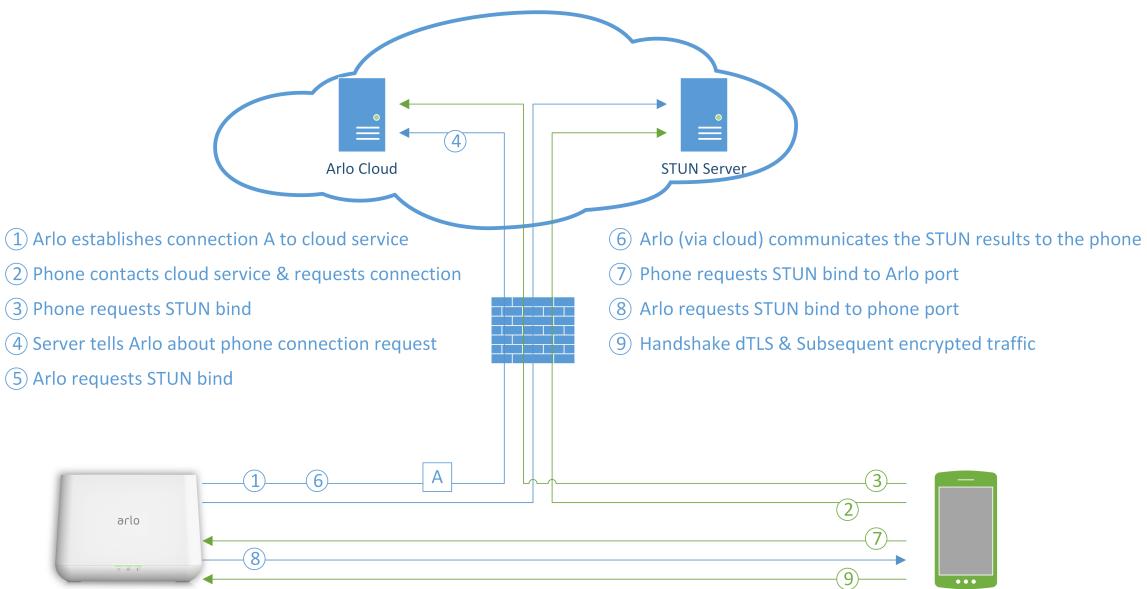


FIGURE A.5 – Arlo - Etablissement de la connexion en locale

## A.5 QBee - Déchiffrement des Réglages

Lors de l’analyse de l’application *QBee Camera* le fichier XML *com.vestiacom.qbee-camera\_preferences.xml* est retrouvé. On observe que l’attribut *name* des tags, ainsi que leur contenu sont encodés en base64. Un décodage du texte en base64 est effectué, mais le contenu est composé par des données binaires.

La structure de données faisait penser à des réglages chiffrés ; une recherche a donc été effectuée pour déterminer quelles bibliothèques existaient pour chiffrer les réglages sur Android, intégrés dans le système ou externes. La bibliothèque *Secure-preferences* (Alexander-Bown, 2013–) a été identifié comme la plus populaire ; la comparaison du fichier des réglages de l’application *QBee Camera* et un fichier d’exemple créé par *Secure-preferences* a montré que la structure correspondait. En même temps l’application *QBee Camera* avait été décompilée avec JADX (Skylot, 2013–). Dans le code source de l’application, l’on a suivi les différentes fonctions afin de trouver la classe gérant les préférences de l’application. La classe identifiée est *com.b.a*.

Le code de *Secure-preferences* a donc été comparé à celui décompilé, comme on peut voir par la comparaison à la figure A.6, le code utilisé par *QBee Camera* correspond à celui de la bibliothèque *Secure-preferences* à la version antérieure à 0.0.4<sup>3</sup>.

Le protocole de chiffrement est AES sans CBC, ainsi le déchiffrement des différents champs

3. <https://github.com/scottyab/secure-preferences/tree/02f3f0ed1ad921d83c404aa203fa6a3a30ab86d1>

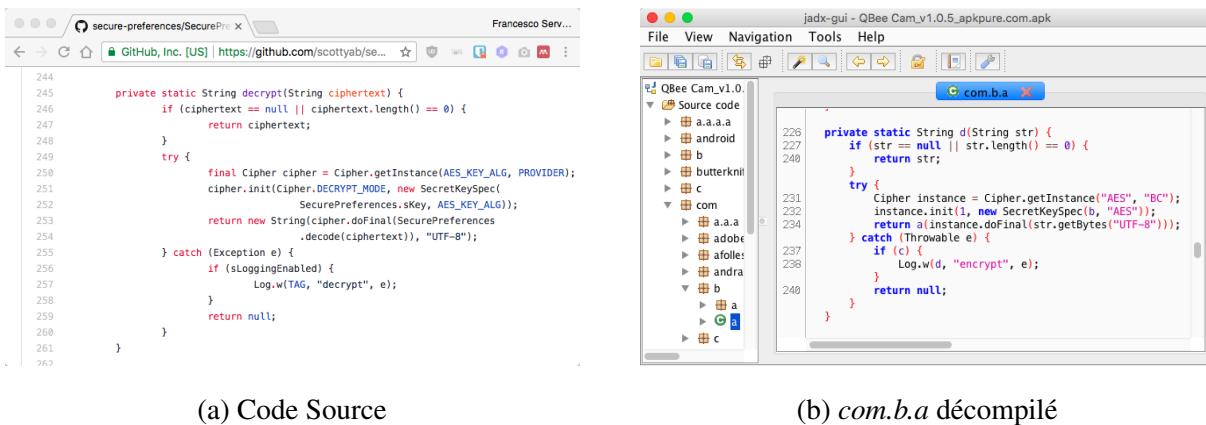


FIGURE A.6 – Comparaison des fonctions de déchiffrement

nécessite seulement la clé AES. Dans la version originale de *Secure-preferences* la clé AES est stockée, encodé en base64, directement dans le fichier XML ; l’application *QBee Camera* utilise une version légèrement modifiée de celle-ci. En effet seulement une partie de la clé est stockée dans le fichier, sous forme d’une chaîne de texte encodé en base64 de 26 caractères. La clé définitive est dérivée à partir de cette chaîne à travers de la fonction présentée à la figure A.7.

Sur la base de ces informations, il a été donc possible d’écrire un script qui puisse parser les fichiers de configuration ainsi chiffrés, extraire les chaînes et retourner un fichier JSON contenant les données déchiffrées.

La bibliothèque utilise l’identifiant unique du dispositif en combinaison avec l’algorithme PKBDF afin de générer une chaîne de caractères unique. Cette chaîne est utilisée comme attribut "name" du tag qui contient la clé AES dans le fichier XML ; l’application peut ainsi stocker et extraire la clé. Dans ce cas on n’a pas à disposition l’identifiant unique, et il n’est donc pas possible d’identifier le tag correct par cette approche ; une approche par brute force a donc été utilisée. Toutes les chaînes de 26 caractères sont extraites et utilisées pour déchiffrer le contenu des différents tags. Si plusieurs clés sont possibles, le fichier JSON contiendra le résultat du déchiffrement avec les différentes clés.

Le script développé s’applique tant au fichier de réglages de *QBee Camera* qu’à celui de *Swisscom Home App*, le code utilisé pour le chiffrement étant effectivement partagé.

Afin de pouvoir exécuter le script avec Autopsy, qui ne supporte que python 2.7 avec une quantité limitée de modules, le script est compilé dans un exécutable Windows avec CXFreeze, et exécuté depuis un plug-in de Autopsy.

Le plug-in pour Autopsy se charge de déchiffrer les fichiers de réglage et d’extraire les

```

private static byte[] c(String str) {
    StringBuilder stringBuilder = new StringBuilder();
    stringBuilder.append(str.substring(0, str.length() / 2));
    stringBuilder.append("a!k@E$2,g86AX&D8vn2]");
    stringBuilder.append(str.substring(str.length() / 2));
    byte[] bArr = null;
    try {
        bArr = MessageDigest.getInstance("SHA256").digest(stringBuilder.toString().getBytes());
    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
    }
    return bArr;
}

```

FIGURE A.7 – Fonction de dérivation de la clé AES

identifiants de l'utilisateur, comme présenté à la figure A.8.

Source File	Username	Secret	Secret Type	Service
com.vestiacom.qbeecamera_preferences.xml	Pinman	Esc_jot_2018	Password	QBee

FIGURE A.8 – QBee - Artefacts personnalisés dans Autopsy

## A.6 iSmartAlarm - Parsing Base de Données

L'application *iSmartAlarm* stocke ces données dans une base de données en format SQLite *iSmartAlarm.db*. Cette base de données présente plusieurs tables ; dans notre cas seul *TB\_IPUDairy*, *TB\_SensorDairy* et *TB\_userDairy* contiennent des données pertinentes. Les données trouvées dans les différentes tables sont corrélées avec les évènements connus, disponibles via l'analyse "en vivo" de l'application ; pour chaque table SQLite un tableau récapitulatif est fourni.

Les horodatages contenus dans ces tables sont particulièrement intéressants pour remarquer la différence entre outils et la nécessité d'en utiliser plusieurs. En effet elles sont stockées en tant qu'horodatage UNIX à 10 ou 13 chiffres ; les colonnes sont par contre définies seulement comme entiers à 8 chiffres. Lors de l'analyse avec UFED Reader cela est strictement imposé par le logiciel, les données subissent donc un overflow et sont interprétées comme nombres négatifs. *DB Browser for SQLite* ou la plus récente visualisation de BD dans Autopsy 4.7.0 n'ont pas ce problème, et interprètent correctement les horodatages stockés (cf. Figure A.9).

	TB_CameraDairy (0)	TB_CountryInfo (10)	TB_IPUDairy (170)	TB_IPUVersionInfo (4)	TB_ISC3Dairy (0)	TB_ISC3VideoInfo (0)	TB_PushMessageInfo (0)	TB_SensorDairy (178)	TB_camera (0)	TB_ipuInfo (0)	TB_isc3Info (0)	TB_logInfo (1)	TB_profilecamera (0)	TB_profileInfo (0)	TB_profileSensor (0)	TB_screenshotsInfo (0)	TB_sensorInfo (0)	TB_updatedevice (0)	TB_userDairy (10)	TBUserInfo (0)
<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/> 000A9474 618134547 14					<input checked="" type="checkbox"/> 000A9474 618134547 14												
<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/> 004D3209D9E4 1522155201 2					<input checked="" type="checkbox"/> 004D3209D9E4 1522155201 2												
<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/> 004D3209D9E4 1521455576 2					<input checked="" type="checkbox"/> 004D3209D9E4 1521455576 2												
<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/> 004D3209D9E4 1521454232 4					<input checked="" type="checkbox"/> 004D3209D9E4 1521454232 4												
<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/> 004D3209D9E4 1521451890 5					<input checked="" type="checkbox"/> 004D3209D9E4 1521451890 5												
<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/> 004D3209D9E4 1521210403 2					<input checked="" type="checkbox"/> 004D3209D9E4 1521210403 2												
<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/> 004D3209D9E4 1521210328 1					<input checked="" type="checkbox"/> 004D3209D9E4 1521210328 1												
<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/> 004D3209D9E4 1521210326 1					<input checked="" type="checkbox"/> 004D3209D9E4 1521210326 1												
<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/> 004D3209D9E4 1521210212 2					<input checked="" type="checkbox"/> 004D3209D9E4 1521210212 2												
<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/> 004D3209D9E4 1521203850 2					<input checked="" type="checkbox"/> 004D3209D9E4 1521203850 2												
<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/> 004D3209D9E4 1521203540 1					<input checked="" type="checkbox"/> 004D3209D9E4 1521203540 1												
<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/> 0006B4E5 -1453707220 6					<input checked="" type="checkbox"/> 0006B4E5 -1453707220 6												
<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/> 0006B4E5 1810316008 5					<input checked="" type="checkbox"/> 0006B4E5 1810316008 5												
<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/> 0006B4E5 1810305684 5					<input checked="" type="checkbox"/> 0006B4E5 1810305684 5												
<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/> 0006B4E5 1809331576 5					<input checked="" type="checkbox"/> 0006B4E5 1809331576 5												
<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/> 0006B4E5 1809321042 5					<input checked="" type="checkbox"/> 0006B4E5 1809321042 5												
<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/> 0006B4E5 1809310380 5					<input checked="" type="checkbox"/> 0006B4E5 1809310380 5												
<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/> 0006B4E5 1736774989 5					<input checked="" type="checkbox"/> 0006B4E5 1736774989 5												
<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/> 0006B4E5 1736764761 5					<input checked="" type="checkbox"/> 0006B4E5 1736764761 5												

Table TB_SensorDairy 178 entries		
sensorID	date	action
000A9474	1521036557331	14
004D3209D9E4	1522155201	2
004D3209D9E4	1521455576	2
004D3209D9E4	1521454232	4
004D3209D9E4	1521451890	5
004D3209D9E4	1521210403	2
004D3209D9E4	1521210328	1
004D3209D9E4	1521210326	1
004D3209D9E4	1521210212	2
004D3209D9E4	1521203850	2
004D3209D9E4	1521203540	1
0006B4E5	1523259682860	6
0006B4E5	1522228738792	5
0006B4E5	1522228728468	5
0006B4E5	1522227754360	5

(a) UFED Reader

(b) Autopsy 4.7.0

FIGURE A.9 – iSmartAlarm - Comparaison horodatages en fonction du logiciel

**TB\_IPUDairy** Contient les informations concernant les changements d'état de la base *CubeOne*<sup>4</sup> ainsi que les déclenchements d'événements d'alarme. Les résultats de la corrélation avec les événements connus sont présentés dans le tableau A.3.

**TB\_SensorDairy** Contient les informations par rapport aux changements d'état des senseurs. Les informations présentées dans le tableau A.4 ont été obtenues par corrélation avec les événements connus, mais une partie aussi par "reverse-engineering" de l'application *iSmartAlarm* décompilé<sup>5</sup> (en gris dans le tableau). Cela permet, lors de l'intégration dans un script, de décoder

4. IPU semble être le nom technique du CubeOne

5. IDs retrouvés dans la classe : andon.isa.setting.Act5\_14\_Sensor\_Logs\_Model

date	▼	action	IPUID	logType	sensorName	operator	sensorType	sensorID	userID	profileId	profileName
Filter		Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
145	1521108904		004D3209D9E4	2		TheBoss (Remote Tag)			2		DISARM
146	1521108878		004D3209D9E4	2		TheBoss (Remote Tag)			1		HOME
147	1521108826	2	004D3209D9E4	5							
148	1521042170		004D3209D9E4	2		TheBoss (Remote Tag)			2		DISARM
149	1521041811	1	004D3209D9E4	5							
150	1521037461		004D3209D9E4	2		skyman			2		DISARM
151	1521037151		004D3209D9E4	2		TheBoss (Remote Tag)			2		DISARM
152	1521037119	1	004D3209D9E4	1	TheBounc...						

FIGURE A.10 – iSmartAlarm - IPUDairy

aussi les informations concernant des évènements qui n'avaient pas été observés.

**TB\_userDairy** Cette table contient les informations concernant les actions de l'utilisateur, selon les observations faites les entrées présentes sont des doublons des entrées avec  $logType=1$  dans la table TB\_SensorDairy. Elles ne sont donc pas considérées dans l'analyse.

**Extraction** Sur la base des observations faites et des tableaux A.3 et A.4 un script python a pu être développé. Celui-ci automatise le processus de parsing des informations depuis la base de données ; l'intégration dans un plug-in pour Autopsy a permis ensuite d'extraire ces informations dans des artefacts personnalisés.

## A.7 iSmartAlarm - Obtention du Firmware

Une partie importante de l'analyse des dispositifs IdO est l'analyse de leur firmware. Cela est utile pour retrouver des failles ou des secrets qui pourraient être stockés. L'on voit l'exemple de Ching (2017).

**Décompilation** Grace à la décompilation de l'application de iSmartAlarm pour Android il a été possible de télécharger la version actuelle du firmware du CubeOne.

Dans *andon.isa.settings.Firmware\_Update* on voit que la méthode *getNewsForIPU* appelle la *getnewst* de la classe *Cloudprotocol* pour vérifier la présence d'un nouveau firmware. A partir de cette méthode les différentes paramètres POST à envoyer et l'URL à laquelle vérifier la version ont pu être déterminés (cf. figure ??). Ainsi il a été possible d'effectuer une requête valide manuellement, et obtenir le lien pour le téléchargement du firmware<sup>6</sup>.

6. [https://d15qxdetlnr1v.cloudfront.net/UpgradeKit/1514393660/IPU3G\\_2.2.4.10\\_OfficialVersion\\_EU\\_NoTel.img](https://d15qxdetlnr1v.cloudfront.net/UpgradeKit/1514393660/IPU3G_2.2.4.10_OfficialVersion_EU_NoTel.img)

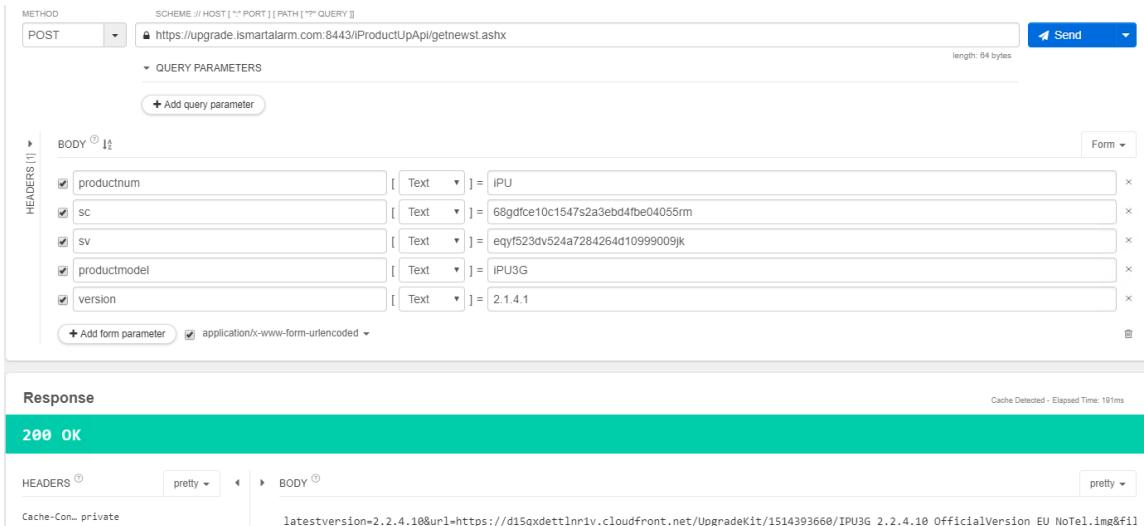
```

public static void getNewsForIPU(int index, Handler handler, Context context, String logInfo, IPU ipu) {
    Log.i(TAG + "getNews", new StringBuilder(String.valueOf(logInfo)).append(" get ipu Firmware URL").toString());
    Message message = new Message();
    message.what = index;
    if (C.cloudProtocol == null) {
        C.cloudProtocol = new CloudProtocol(context, C.getCurrentUser(TAG).getTels(), C.getCurrentUser(TAG).getTels(), C.getCurrentUser(TAG));
    }
    Map<String, String> param = C.cloudProtocol.getnewst(ipu.getIpuid(), ipu.getFirmwareVersion(), ipu.getProductNum(), ipu.getProductModel());
    HttpModel.getHttpModelInstance().httpPostRequest(1000, Url.getnewst, param, new AnonymousClass3(message, handler));
}

public Map<String, String> getnewst(String mac, String version, String productnum, String productmodel, String hardwareversion) {
    this.parameter.clear();
    this.parameter.put("sc", "68gdfce10c1547s2a3ebd4fbe04055rm");
    this.parameter.put("sv", svCode.getnewst());
    this.parameter.put("mac", mac);
    this.parameter.put("version", version);
    this.parameter.put("productnum", productnum);
    this.parameter.put("productmodel", productmodel);
    this.parameter.put("hardwareversion", hardwareversion);
    this.parameter.put("testcode", CommonUtilities.TESTCODE);
    return this.parameter;
}

```

(a) Recherche paramètres et URL



(b) Requête manuelle

FIGURE A.11 – iSmartAlarm - Récherche URL du Firmware

Cependant, lors de l’analyse des données sur iPhone, faite en fin de travail, la même url est retrouvée dans le fichier de configuration de l’application<sup>7</sup>.

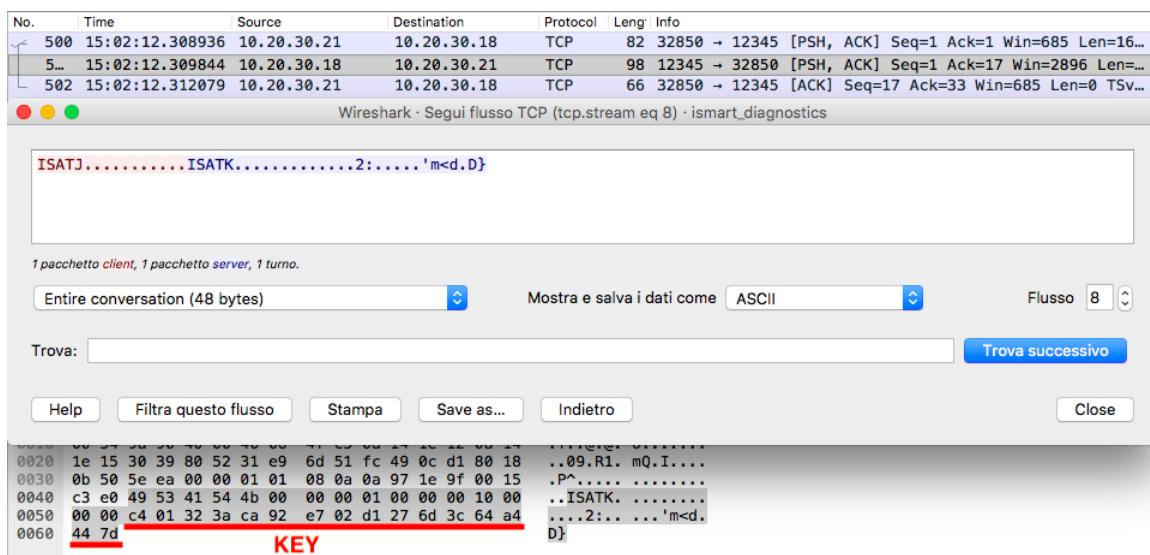
## A.8 iSmartAlarm - Logs de Diagnostique

L’application *iSmartAlarm* fournit à l’utilisateur la possibilité d’envoyer au service technique les logs de diagnostic du dispositif. Il s’agit de la seule application ayant cette fonctionnalité. On a donc étudié le processus de collecte de ces logs. La première remarque fut que les logs sont envoyés directement au service technique, ils ne sont pas cachés sur le smartphone, ni un fichier est fourni pour être mis en annexe d’un email. Lors de la requête des logs via l’application, une capture réseau est effectuée. Celle-ci permet de faire plusieurs observations :

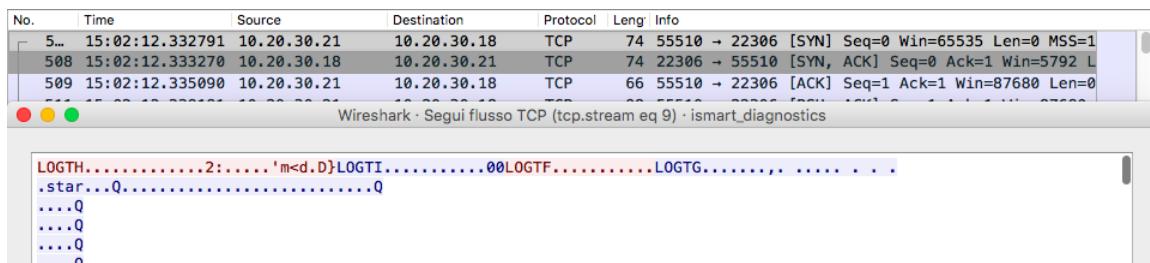
7. com.ismartalarm.ismart.plist

- Le trafic entre le smartphone Android et le CubeOne n'est pas chiffré.
- Le trafic se fait d'abord sur le port 12345 et ensuite transite au port 22306
- Le smartphone ne s'authentifie pas auprès du CubeOne lors de la connexion sur le port 12345 ; la connexion sur le port 12345 permet à l'application d'obtenir une clé pour authentifier la requête successive. Vu que la première requête n'est pas authentifiée, il est possible d'envoyer des requêtes manuelles pour recevoir les logs.

La figure A.12 montre un extrait de ce processus de connexion.



(a) Récupération de la clé sur le port 12345



(b) Authentification sur port 22306 et téléchargement des logs

FIGURE A.12 – iSmartAlarm - Extrait connexion pour les logs de diagnostique

Une fois capturées, ces données diagnostiques se présentent sous forme d'un fichier mélangé de données binaires et de texte. Si d'une part le composant binaire n'a pas pu être décodé, la partie de texte (dont un extrait se trouve à la figure A.13 contient plusieurs informations intéressantes. Les entrés des logs sont séparés par le caractère \$, entre ces entrés on peut retrouver les logs des requêtes effectués par le CubeOne, ainsi que tous les paramètres POST qui ont été utilisés. Dans les premiers 17 caractères, on retrouve l'horodatage UNIX encodé sous forme hexadécimale.

Ces entrées sont corrélées avec les évènements connus, comme dans le cas de la base de données, afin de pouvoir comprendre la structure des logs. Sur la base de ça un script python permettant d'automatiser le processus d'extraction et de parsing a été écrit. Le script ne parse qu'une partie des données, notamment il utilise des REGEX pour parser les évènements générés par le senseur de contact ou les changements d'état du CubeOne (ARM, DISARM...). Cependant les logs des requêtes POST à `/api/SensorStateULoad.htm` et `/api/SStatus_Change.htm` sont potentiellement aussi intéressantes. Leur limitation est que le type de senseur n'est pas connu, seuls l'ID du senseur et l'ID de son état actuel sont connus, mais on ne peut pas différencier entre un senseur de mouvement ou un senseur de contact par exemple. Lors d'une investigation, il serait donc nécessaire de récupérer aussi les identifiants des différents senseurs afin de pouvoir parser correctement ces entrées dans les logs.

```
AF9::APSEND::the receive message is AP auto send, try to get more message$@00000005AFADAF9::A  
d$@00000005AFADAF9::ALARMDOOR:{ "SensorID": "000A8540", "MessageType": "1", "TS": "1526389497550" },  
AFB::APSEND::the receive message is AP auto send, try to get more message$@00000005AFADAFB::A
```

FIGURE A.13 – iSmartAlarm - Fermeture de la porte dans les logs de diagnostic

## A.9 Arlo - Accès aux données sur le Cloud par réutilisation du token

Lors de l'analyse de l'application *Arlo* un token d'authentification a été retrouvé dans la base de données *default.realm*. Si d'une part l'analyse du Cloud ne faisait pas partie du sujet de recherche de ce travail, d'autre part il a été jugé intéressant de voir si le token pouvait être utilisé pour l'accès aux données contenues sur le cloud. Cela en particulier, car une bibliothèque python, *arlo*<sup>8</sup>, existait déjà. Dans son utilisation traditionnelle, la classe principale, *Arlo*, est initialisée avec le nom d'utilisateur et le mot de passe ; une fois cela fait le token d'authentication et l'ID du compte de l'utilisateur sont stockés et utilisés pour authentifier les requêtes. Cet ID et token correspondent à ceux qu'on retrouve lors de l'analyse de l'application *Arlo*.

Une nouvelle classe, *Arlo\_Forensic*, héritant de *Arlo* a donc été implémentée ; En initialisant cette classe directement avec l'ID du compte et le token, on peut sauter l'étape de login et effectuer directement des requêtes.

8. <https://github.com/jeffreydwalter/arlo>

On peut ainsi utiliser ces deux identifiants pour télécharger les données depuis le cloud. Un exemple de l'utilisation de cette nouvelle classe, qui réutilise le code fourni dans le *README* de *Arlo* est donné à la figure A.14.

The screenshot shows the PyCharm IDE interface. The top part displays the code for the *arlo* module, specifically the *Arlo* class which iterates through recordings in a library and downloads them. The bottom part shows a terminal window with a log of video download commands and their results, indicating files were downloaded from 20180531 at various times between 13:57:23 and 09:08:53.

```

    Arlo
    Library = arlo.GetLibrary(seven_days_ago, today)
    # Iterate through the recordings in the library.
    for recording in Library:
        videotimestamp = datetime.datetime.fromtimestamp(int(recording['name']) // 1000).strftime(
            "%Y-%m-%d %H-%M-%S") + '-' + recording['uniqueId'] + '.mp4'
        # The videos produced by Arlo are pretty small, even in their longest, best quality settings,
        # but you should probably prefer the chunked stream (see below).
        ## Download the whole video into memory as a single chunk.
        # video = arlo.GetRecording(recording['presignedContentUrl'])
        # with open('videos/' + videofilename, 'wb') as f:
        #     f.write(stream)
        #     f.close()
        # Or:
        # Get video as a chunked stream; this function returns a generator.
        stream = arlo.GetRecording(recording['presignedContentUrl'])
        with open(videofilename, 'wb') as f:
            for chunk in stream:
                f.write(chunk)
            f.close()
        print('Downloaded video ' + videofilename + ' from ' + recording['createdDate'] + '.')

```

FIGURE A.14 – Arlo - Téléchargement Videos Dernière Semaine

## A.10 iOS - Traces sur iOS 11

L'analyse des traces sur iOS 11 est faite à partir d'une capture du système de fichier d'un iPhone 6 débridé<sup>9</sup>(Archive Tar via SSH). Cet iPhone a été utilisé pour le scénario du DFRWS, et l'application Swisscom Home App n'était pas installée.

L'extraction suite au débridage n'a pas pu être faite qu'à la fin du travail. La recherche de traces est donc seulement un aperçu des traces disponibles. Aucun tentative de compréhension ni décodage approfondi des traces n'est fait.

Les fichiers contenant de possibles traces d'intérêt sont présentés à la table A.5.

De particulier intérêt sur iOS est la possibilité de parsing des notifications de système. Ainsi on a accès à des informations sur des évènements qui n'ont pas été encore ajoutés aux données des différentes applications.

9. Débridage effectué avec Electra

## A.11 Rapports de Vulnérabilités

Les rapports de vulnérabilité sont présentés aux figures A.15, A.16, A.17 et A.18. Un dépôt GIT est disponible à l'adresse [https://github.com/fservida/msc\\_thesis\\_vulnerabilities](https://github.com/fservida/msc_thesis_vulnerabilities); celui-ci contient les différents rapports, *Proof of Concept* et une vidéo de démo. La vidéo de démo pour la vulnérabilité *QBee Multi-Sensor Camera - Trafic en clair* est aussi disponible à l'adresse [https://youtu.be/dd8vt0\\_DJF4](https://youtu.be/dd8vt0_DJF4).

## A.12 Nest - USB Analysis

Lors de la connexion via USB des deux dispositifs Nest le trafic sur l'interface USB a été capturé avec Wireshark afin de déterminer les protocoles de connexion supportés par les dispositifs. Les deux annoncent le support pour la classe de protocoles *Mass Storage* pendant la connexion, comme le montrent les images A.19 et A.20.

## A.13 Arlo - Physical Analysis

La figure A.21 montre les logs complets lors d'un démarrage normal. La figure A.22 montre, de sa part, un extrait des réglages du NVRAM, comme affichés depuis la console du bootloader.

Depuis la console du bootloader, il a été essayé, sans succès, de démarrer le système avec des paramètres de démarrage différent, de manière similaire à ce qui a été fait avec le WinkHub, afin de forcer le kernel à présenter une console de système.

**Accès Root** Depuis la console du bootloader il est possible de changer la valeur des réglages dans le NVRAM. La modification de la valeur associée à *telnetd\_enable* de 0 à 1, permet lors du démarrage du Arlo d'activer un serveur telnet sur le Arlo. Il est donc possible de se connecter via telnet à une console root de busybox. Avec la commande *mount* il est possible identifier */tmp/media/nand/* comme le point de montage de la mémoire NAND. Une archive du dossier est donc produite et sauvegardée sur une clé USB. Une archive complète du système n'a pas pu être obtenue ; toute exécution de la commande tar avait comme résultat une archive incomplète.

Dans le dossier */tmp/media/nand* on retrouve plusieurs données :

**Logs** Le fichiers compressés *syslog-\*\*\*\*.gz* contiennent l'historique des logs de système.

Les logs contiennent plusieurs entrées concernant les évènements déclenchés par la

caméra, notamment sur les détections de mouvement.

**Fichiers de configuration** Dans le dossier `vzdaemon/conf` on retrouve plusieurs fichiers JSON de configuration de la base ainsi que de la camera.

C'est important de remarquer comme l'accès root au Arlo n'a été obtenu qu'en fin de travail (4 juillet), cependant les logs de système archivés couvrent toute la période depuis la réinitialisation aux paramètres par défaut faite le 14 mai. C'est donc envisageable d'utiliser ces nouvelles données dans le cadre du challenge forensique du DFRWS.

## A.14 iSmartAlarm - Extraction Physique

L'extraction physique de la mémoire est faite sur les régions de mémoire spécifiées dans la section 4.5.4. Depuis la console de UBoot la commande `md.b` a été utilisé. Celle-ci affiche à l'écran le contenu de la mémoire sous forme hexadécimale et ASCII (Figure A.23).

Le texte affiché dans la console a été enregistré avec minicom dans un fichier texte. Ensuite le script `uboot-mdb-dump`<sup>10</sup> a été utilisé pour convertir le contenu du fichier texte dans une image binaire.

Le dump de la mémoire procède à la vitesse de connexion série, notamment 57'600 bps ; si on considère l'*overhead* dû à l'affichage de l'offset et du code ASCII, les données sont transmises à ~1Ko/s ; à cette vitesse l'extraction de juste 8Mio, tels que la taille du chip SPI de mémoire, prend autour de 2 h 20. C'est donc pourquoi seul 8Mio depuis les trois régions de mémoire plus promettantes ont été extraites.

A noter que l'utilisation de screen comme émulateur de console pour la capture du résultat de `md.b` résulte dans un fichier qui n'est pas accepté par le script, et parfois incomplet ; l'utilisation de minicom produit des meilleurs résultats.

## A.15 DFRWS

Dans le cadre de ce travail, un scénario IdO pour le challenge forensique du DFRWS a été préparé. La génération des traces a été conduite avec l'aide des étudiants du cours *Appareils Mobiles* donné aux étudiants en Master de l'ESC.

---

10. <https://github.com/gmbnomis/uboot-mdb-dump>

**Scénario - Confidential** Jessie Pinkman est propriétaire d'un laboratoire illégale de production de stupéfiants. Ayant peur d'une attaque d'une bande rivale il équipe son laboratoire avec un système d'alarme (iSmartAlarm) avec senseur de contact sur la porte et détecteur de mouvement et trois différentes cameras, une par local. De plus, par peur d'incendies, il installe un détecteur de fumée dans le local de synthèse. Il installe aussi un Amazon Echo, qui est lié au système d'alarme et un WinkHub, pour pouvoir centraliser la gestion de ses appareils IdO.

Le 15 et 16 mai, deux amis, Donnie Pandana (surnommé Panda) et Sergio Varga, lui rendent visite au laboratoire ; Jessie leur donne donc accès au réseau WiFi. Comme la semaine successive il avait prévu de se faire aider par Donnie pour des travaux, il l'ajoute aux utilisateurs de la maison dans l'application de iSmartAlarm.

Le 17 mai, pendant le travail dans le laboratoire Jessie active et désactive le système d'alarme plusieurs fois, car il doit sortir à plus reprises du laboratoire. Il rentre dans le laboratoire et l'active une dernière fois à 10 h 34 min 17 s pour qu'il se déclenche lors de l'ouverture de la porte. Peu de secondes après juste quand il était en train de recommencer à travailler, à 10 h 34 min 31 s, l'alarme est désactivée par Donnie, à 10 h 34 min 36 s la porte est ouverte et une bande rivale rentre dans le laboratoire en faisant une razzia. Donnie avait aussi profité peu avant de l'accès au WiFi pour intercepter le trafic entre le smartphone de Jessie et la caméra QBee surveillant l'entrée ; ainsi il avait pu désactiver la caméra en utilisant la vulnérabilité du trafic en clair (cf. section 4.4.1). La bande rivale en profite pour essayer de mettre le feu au laboratoire, ce qui déclenche le détecteur de fumée à 10 h 36. Cependant ils fuient en vitesse, et ils ne se rendent pas compte que le feu, qui n'avait pas bien abouti, s'éteint quelque moment après. Un passant, qui avait remarqué du bruit et de la fumée, se rend sur place à 10 h 39. Lorsqu'il remarque ce qui s'est passé, il appelle la police, qui arrive sur scène peu après, à 10 h 45. Lors de l'état de lieux, la police remarque les objets IdO ; un enquêteur IT intégré au groupe procède donc sur les lieux au prélèvement des logs de diagnostic depuis la station de base de iSmartAlarm, et du système de fichiers du WinkHub, ainsi qu'au prélèvement des stations de base de Arlo et de iSmartAlarm. Comme la police remarque que la caméra QBee était désactivée, ils procèdent aussi à une capture du trafic réseau de la caméra. En laboratoire il procède à l'extraction des images de mémoire depuis le bootloader des deux stations de base.

Une fois en possession du smartphone du suspect (Samsung Galaxy Edge S6) il procède aussi à son extraction physique.

**Données collectés** Dans le cadre du scénario les données suivantes ont été collectées :

- Extraction physique du téléphone : la procédure présentée dans la méthodologie est appliquée, deux images sont donc faites ; avant et après une analyse des applications. Comme on s'attendait, seule la deuxième image contient les données par rapport aux évènements d'intérêts, c'est donc celle-ci qu'il faudra donner lors du challenge.
- iSmartAlarm - Logs de diagnostic
- iSmartAlarm - Images de mémoire : seules les images aux offset 0x0000'0000 et 0x8000'0000 ont été extraites. Etant donné que lors de l'analyse aucune donnée n'avait été trouvée dans l'image à l'offset 0xbc00'0000, et du temps qu'une extraction supplémentaire aurait pris, elle n'a pas été jugée nécessaire.
- Arlo - Image de mémoire : Image à l'offset 0x0000'0000 de 256Mio de taille a été extraite (*dfrws\_arlo.img*).
- Arlo - NVRAM : Les réglages du NVRAM sont sauvegardés.
- Arlo - NAND : Une archive du dossier */tmp/media/nand* est extraite.
- Wink - Image système de fichiers : Archive TAR du système de fichiers obtenu par SSH.
- Echo : En utilisant CIFT<sup>11</sup> les données sur le cloud de Amazon, concernant le Amazon Echo, sont collectées.

## Limites

**Samsung** L'on remarque lors d'une rapide analyse des données sur le Samsung que des données non pertinentes au scénario sont présentes dans le dossier *media*, ainsi que dans les applications de Google Drive, Chrome et Arlo. Notamment la réinitialisation du téléphone, contrairement à l'avertissement "Toutes les données seront effacées", ne comprend pas le dossier *media*, qui contient les données de certaines applications, mais aussi les images prises avec le téléphone. De plus faute à la synchronisation automatique de Chrome, qui par erreur avait été activé entre l'instance sur l'ordinateur personnel et le téléphone, une partie des recherches, documents visualisés et données de remplissage automatique ont été synchronisés sur la mémoire du smartphone. Enfin dans la base de données *default.realm* de Arlo, l'on retrouve *Francesco Servida* comme propriétaire de la caméra. Cela est dû au fait que le premier compte lié était à mon nom ; lors de la préparation pour le scénario, la caméra a été effacée du compte et, après avoir réinitialisé la base, liée au compte de Jessie. Cependant la caméra ne présente pas un moyen

---

11. <https://bitbucket.org/cift/pycift> (Chung et al., 2017)

d'être réinitialisé, et il semblerait qu'elle, ou les serveurs cloud, gardent en mémoire le nom et prénom du premier utilisateur.

Il semblerait que toutes les données sensibles à cacher soient dans *media*, pour la publication de l'image il devrait donc être possible d'effacer ce dossier, sans perte de données pertinentes. Les données synchronisées sont toutes stockées de manière non compressée, une substitution par recherche de chaînes est donc envisageable pour les cacher.

**Wink** L'image du système de fichier a été extraite par SSH ; cela demande qu'un accès root soit déjà obtenu. Dans le cas contraire, pour obtenir l'accès root, il est nécessaire de redémarrer le dispositif ce qui entraîne la perte de tous les fichiers temporaires. Dans le cadre du scénario l'accès root avait été obtenu préalablement, et ces fichiers ont donc pu être collectés. Cependant dans une situation réelle cela n'est vraisemblablement pas le cas.

TABLE A.1 – Résumé des chips d'intérêt et portes d'accès physique disponibles identifiés suite à l'analyse préliminaire

Dispositif	Chip	Description	Portes
Nest Camera	Ambarella A5s	ARM SoC	USB
	CT49248DD486C1	2Gb NAND + 2 Gb DRAM	
Nest Protect	Freescale Kinetis K60	ARM MCU	USB
Arlo	BCM53573	ARM SoC	JTAG, Série
	W29N01GVSIAA	1Gb NAND	
iSmartAlarm	Ralink RT5350F	MIPS SoC	JTAG, Série
	mxic MX25L6406E	64Mb SPI ROM	
WinkHub	Freescale i.MX28	ARM CPU	JTAG, Série, USB
	Spansion S34ML01G100TFI000	1Gb NAND	
QBee	Vatics Mozart 385s	SoC	-
	Spansion ML01G100BH100	1Gb NAND	

TABLE A.2 – Endpoint API découverts via analyse réseau sur caméra QBee

Endpoint	Type	Fonction Observé	Cookies Présents
/verify	GET	Heartbeat	JSESSIONID, GC_ID, DST_PORT
/event?service=webdis&transport=stream	GET	Informations Environnementales <sup>a</sup>	JSESSIONID, GC_ID, LD_ID, SERVICE
/live2.sdp	GET	Live Stream RTSP	JSESSIONID, GC_ID, LD_ID,
			CUSTOM_KEEPALIVE_TIMEOUT
/config/get?service=webdis	GET	Obtenir Reglages	JSESSIONID, GC_ID, LD_ID, SERVICE
/config/set?service=webdis	POST	Modification Reglages	JSESSIONID, GC_ID, LD_ID, SERVICE
/stream/begin	POST		
/technical?service=webdis	GET	Informations Techniques	JSESSIONID, GC_ID, LD_ID, SERVICE

<sup>a</sup>. Température, Luminosité, Humidité

TABLE A.3 – iSmartAlarm - IDs de la table TB\_IPUDairy

logType	Type	action	Signification	profileId	profileName
1	Alarm	1	Contact Sensor Alarm	-	-
		2	Motion Sensor Alarm	-	-
2	Profile Change	-	-	0	ARM
		-	-	1	HOME
		-	-	2	DISARM
		-	-	3	PANIC
5	Cube Status	1	Cube Offline	-	-
		2	Cube Online	-	-

TABLE A.4 – iSmartAlarm - IDs de la table TB\_SensorDairy

logType	Type	action	Signification
1	RemoteTag Action	1	Arm
		2	Disarm
		3	Back Home
		4	Left Home
-	-	1	Ouvert (Dispositif Inconnu)
		2	Fermé (Dispositif Inconnu)
		3	Porte Ouverte
		4	Porte Fermée
		5	Détection de Mouvement
		6	Batterie Faible
		7	<i>Nominal Power</i>
		8	Test de Senseurs (model == 0)
		8	Détection de Fumée (model == 1)
		14	Device Added
		15	Device Deleted

TABLE A.5 – Fichiers d'intérêt sur iOS 11

Application	Dossier/Fichier <sup>a</sup>	Contenu
-------------	------------------------------	---------

iSmartAlarm	/Documents	Logs de diagnostic en cache Logs application avec liste dispositifs
	/Library/Preferences/com.ismartalarm.ismart.plist	Données Utilisateur (Mot de Passe en clair)
		Logs Dispositif et Senseurs
Wink	/Library/Caches/com.quirky.wink/fCachedData	Vignettes caméras
QBee	/Library/com.askey.qbeecam/cache.db	Logs des requêtes réseau en cache
	/Library/events/images/	Images enregistrées par la caméra
Nest	/Documents/Nest.sqlite	Liste des dispositifs
		Données sur le geofence et les transitions associées
	/Library/Caches/com.nestlabs.jasper.release/Cache.db	Logs des requêtes réseau en cache
Arlo	-	-

<sup>a.</sup> Tous les dossiers sont relatifs au container de l'application dans /private/var/mobile/Containers/Data/Applications/APP\_ID/

2018-06-08

QBee Camera App - Vulnerability Report

## Vulnerability Report – QBee Camera App (Android)

### Description

The android version of the QBee Camera application stores the encrypted user password in the configuration file alongside the AES key for decryption (com.vestiacom.qbeecamera\_preferences.xml).

### Platform

The following devices and versions have been used during the test:

- Samsung Galaxy Edge 6, Android OS 6.0.1
  - o QBee Camera App – Version 1.0.5

### Impact

CVSS v3 Severity and Metrics:

- Base Score: 6.4
- Vector: AV:P/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H
- Impact Score: 5.9
- Exploitability Score: 0.5

### Steps to reproduce

The vulnerability requires the attacker to have physical access to the user device in order to extract the configuration file from the QBee Camera application.

In the configuration file the user password is stored encrypted. However, the decryption key can be derived by combining a cleartext key with a hardcoded key present in the application APK.

Using the derived key, the attacker is able to decrypt the configuration file and extract the user password.

A proof of concept decryption script along with a configuration file extracted from the test phone is provided (crypto\_dec.py). The script accepts a “com.vestiacom.qbeecamera\_preferences.xml” file as input and outputs a decrypted version of the configuration in JSON format.

### Recommendations

The application should use a token-based authentication system and only store the authorization token in the configuration.

2018-06-08

QBee Multi-Sensor Camera - Vulnerability Report

## Vulnerability Report – QBee Multi-Sensor Camera

### Description

Cleartext transmission of session cookies between the phone application on android (“QBee Camera” or “Swisscom Home App”) and the camera.

An attacker with access to the local network is able to read and reuse the cookies in order to gain access to the camera settings, potentially disabling the camera.

In case the user solely relies on the Swisscom Home App for the camera management the attack can result in a complete denial of service of the camera (DoS).

### Platform

The following devices and versions have been used during the test:

- Samsung Galaxy Edge 6, Android OS 6.0.1
  - o QBee Camera App – Version 1.0.5
  - o Swisscom Home App – Version 10.5.2
- QBee Camera
  - o HW Version: 4.1 - Firmware Version: 4.16.4

### Impact

CVSS v3 Severity and Metrics:

- Base Score: 6.4
- Vector: AV:A/AC:H/PR:N/UI:N/S:U/C:L/I:L/A:H
- Impact Score: 4.7
- Exploitability Score: 1.6

### Steps to reproduce

The vulnerability requires the attacker to have access to the local network and to be able to record the traffic between an authorized phone and the camera.

Once the user opens and uses the application the cookies are transmitted in cleartext over http to the camera.

The attacker can thus parse the cookies and reuse them in a custom script in order to gain access to the camera functions and settings.

A proof of concept script along with a network capture are provided alongside this report (qbee.py, pcap\_parser.py & qbee\_camera\_network.pcap); in the capture file the traffic until 19:51 is generated by the “QBee Camera” application and that after 19:52 by the “Swisscom Home App”.

1. Intercept traffic between an authorized phone and the camera
2. Manually obtain the JSESSIONID, GC\_ID and LD\_ID from the pcap or parse the pcap with “pcap\_parser.py” (eg. “python pcap\_parser.py your\_capture.pcap camera\_ip”)
3. Load the credentials and camera IP in qbee.py
4. Edit and run qbee.py to reflect the wanted status of the camera.

The demo video (qbee\_vulnerability.mp4) included provides more insight on how the attack is carried out.

### Recommendations

Encrypt all traffic between the phone and the camera with https.

Francesco Servida

University of Lausanne

In collaboration with Seculabs SA

CONFIDENTIAL

1/1

FIGURE A.16 – QBee Multi-Sensor Camera - Trafic en clair

2018-06-08

iSmartAlarm App - Vulnerability Report

## Vulnerability Report – iSmartAlarm App (Android)

### Description

The android version of the iSmartAlarm application stores the user password in cleartext in the configuration file (iSmartAlermData.xml).

### Platform

The following devices and versions have been used during the test:

- Samsung Galaxy Edge 6, Android OS 6.0.1
  - o iSmartAlarm App – Version 2.0.8

### Impact

CVSS v3 Severity and Metrics:

- Base Score: 6.4
- Vector: AV:P/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H
- Impact Score: 5.9
- Exploitability Score: 0.5

### Steps to reproduce

The vulnerability requires the attacker to have physical access to the user device in order to extract the configuration file from the iSmartAlarm application.

In the configuration file the user password is stored in cleartext.

### Recommendations

The application should use a token-based authentication system and only store the authorization token in the configuration.

Francesco Servida  
University of Lausanne  
In collaboration with Seculabs SA

CONFIDENTIAL

1/1

FIGURE A.17 – iSmartAlarm - Diagnostique CubeOne

2018-06-08

iSmartAlarm CubeOne - Vulnerability Report

## Vulnerability Report – CubeOne

### Description

An unauthenticated attacker with access to the local network is able to obtain a copy of the diagnostic logs from the CubeOne device.

### Platform

The following devices and versions have been used during the test:

- CubeOne:
  - o Firmware Version - 2.2.4.10

### Impact

CVSS v3 Severity and Metrics:

- Base Score: 4.3
- Vector: AV:A/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N
- Impact Score: 1.4
- Exploitability Score: 2.8

### Steps to reproduce

The vulnerability requires the attacker to have access to the local network on which the device resides.

The attacker is able to obtain a copy of the diagnostics logs from the CubeOne device without authentication needed.

The logs contain sensible information such as the interaction the user had with the cube one or the sensor logs. Other sensible information could be present but has not been studied.

A proof of concept script is provided (ismartalarm.py), the script can be used to obtain the diagnostic logs from the device. Some example data can be parsed using the script.

Refer to the README included as well as the command line help (-h) for information about dependencies and usage of the script.

### Recommendations

The diagnostic logs can contain sensible data, the endpoint to collect them should be authenticated, and the traffic should be encrypted.

Francesco Servida  
University of Lausanne  
In collaboration with Seculabs SA

CONFIDENTIAL

1/1

FIGURE A.18 – iSmartAlarm - Stockage du Mot de Passe

No.	Time	Source	Destinatio	Protocol	Leng	Info
105	10:49:32.265651	16.9.0	host	USB	40	GET DESCRIPTOR Request STRING
106	10:49:32.265846	16.9.0	host	USB	34	GET DESCRIPTOR Response STRING
107	10:49:32.265913	16.9.0	host	USB	40	GET DESCRIPTOR Request STRING
1...	10:49:32.266039	16.9.0	host	USB	58	GET DESCRIPTOR Response STRING
109	10:49:32.273886	16.9.0	host	USBMS	40	GET MAX LUN Response
110	10:49:32.273995	16.9.0	host	USBMS	33	GET MAX LUN Response
111	10:49:32.274081	16.9.1	host	USB	32	URB_BULK out (submitted)
112	10:49:32.274100	16.9.1	host	USBMS	63	
▶ Frame 108: 58 bytes on wire (464 bits), 58 bytes captured (464 bits) on interface 0						
▶ USB URB						
▼ STRING DESCRIPTOR						
bLength: 26						
bDescriptorType: 0x03 (STRING)						
bString: Mass Storage						

FIGURE A.19 – Nest Camera - Connexion USB

No.	Time	Source	Destinatio	Protocol	Leng	Info
13	11:21:16.882171	16.10.0	host	USB	40	GET DESCRIPTOR Request STRING
14	11:21:16.882298	16.10.0	host	USB	34	GET DESCRIPTOR Response STRING
15	11:21:16.882368	16.10.0	host	USB	40	GET DESCRIPTOR Request STRING
16	11:21:16.882541	16.10.0	host	USB	66	GET DESCRIPTOR Response STRING
17	11:21:16.885796	16.10.0	host	USB	40	GET DESCRIPTOR Request CONFIGURATION
18	11:21:16.885969	16.10.0	host	USB	41	GET DESCRIPTOR Response CONFIGURATION
19	11:21:16.886280	16.10.0	host	USB	40	GET DESCRIPTOR Request CONFIGURATION
20	11:21:16.886392	16.10.0	host	USB	64	GET DESCRIPTOR Response CONFIGURATION
▶ Frame 20: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface 0						
▶ USB URB						
▶ CONFIGURATION DESCRIPTOR						
▼ INTERFACE DESCRIPTOR (0.0): class Mass Storage						
bLength: 9						
bDescriptorType: 0x04 (INTERFACE)						
bInterfaceNumber: 0						
bAlternateSetting: 0						

FIGURE A.20 – Nest Protect - Connexion USB

```

26 Decompressing...done
27
28
29 CFE for Foxconn Router VMB4000 version: v1.0.4
30 Build Date: Tue Aug 2 15:04:59 CST 2016
31 Init Arena
32 Init Devs.
33 Boot up from NAND flash...
34 Boot partition size = 262144(0x40000)
35 NFLASH Boot partition size = 524288(0x80000)
36 DDR Clock: 392 MHz
37 Info: DDR frequency set from clkfreq=900,*392*
38 et0: Broadcom BCM47XX 10/100/1000 Mbps Ethernet Controller 9.10.178.4002 (r623328)
39 CPU type 0x0: 900MHz
40 Tot mem: 131072 KBytes
41
42 Device eth0: hwaddr 08-02-8E-FF-75-4E, ipaddr 192.168.1.1, mask 255.255.255.0
43 ..... gateway not set, nameserver not set
44 img_boot=1
45
46 trx_failed=0 at <nflash1.trx2>
47 Got the Linux image 1
48 Loader:raw Filesys:raw Dev:nflash0.os2 File: Options:(null)
49 Loading: ..... 5504736 bytes read
50 Entry at 0x00008000
51 Closing network.
52 Starting program at 0x00008000

```

FIGURE A.21 – Arlo - Boot Log

```

22 CFE> nvram show
23 x_broker_port=443
24 ap_mode_cur=1
25 wlg_wds_mode=1
26 wl_radius_port=1812
27 wla_temp_wep_length_2=0
28 wlan_acl_dev24=
29 wan2_dns=
30 wl0_scb_activity_time=0
31 gui_check_enable=1
32 wlan_acl_dev25=

```

FIGURE A.22 – Arlo - Extrait Réglages NVRAM

```

U-Boot 1.1.3 (Apr 28 2013 - 17:57:40)
RT5350 # md.b 0xffffffff0 0x10
bfffffff0: dd ba dd ba dd ba dd ba dd ba dd ba ..... .

```

FIGURE A.23 – iSmartAlarm - Commande *md.b*