C&ESAR 2016

Computer & Electronics
Security Applications
Rendez-vous

Internet des Objets
Vous avez dit sécurité ?

21-23 novembre 2016
Rennes - France

http://www.cesar-conference.org

C&ESAR 2016 :
Internet des Objets vous avez dit sécurité ?

Pour sa 23ème édition la conférence C&ESAR aborde l'Internet des Objets. En préparant le sujet, le comité de programme a hésité à vous proposer ce thème dès cette année. Il y avait le risque de la facilité de rebondir sur une thématique très en vogue. Disposerions-nous de la matière suffisante avec un niveau d'expertise en sécurité satisfaisant pour les attendus de la conférence C&ESAR ?
Ces interrogations ont rapidement été rattrapées par l'accélération de l'actualité. Les récentes cyberattaques sur OVH et sur DYN lancées massivement depuis des objets connectés démontrent qu'il ne s'agit pas de suivre une mode, mais bien d'anticiper les agressions à venir.
L'analyse des soumissions a permis de constater que la sécurité était une préoccupation très présente dans les développements des objets connectés. Le programme de la conférence illustre cette prise en compte de la sécurité en amont des projets. La sécurité tend à devenir un service à valeur ajoutée.
Le comité de programme s'est beaucoup interrogé sur le contour de l'Internet des Objets. Chaque grand domaine d'application, communication, transport, énergie, domotique, robotique, finances, services à la personne… a sa propre approche. Quel modèle de sécurité faut-il envisager ? Le début de la conférence doit permettre d'éclairer la communauté sur cette diversité de vue sans omettre de considérer que ces modèles peuvent s'interpénétrer. Ne parle-t-on pas déjà du concept de Bring Your Own IoT ?
Le programme de C&ESAR 2016 s'attache à faire un point de l'ensemble de ces questions selon différents axes :
- les domaines d'application de l'Internet des Objets ;
- un panorama des problématiques spécifiques de sécurité ;
- un point de situation sur quelques normes et standards pour objets connectés ;
- des illustrations des menaces et des moyens pour réduire le risque ;
- des perspectives de solutions.

Les comités d'organisation et de programme tiennent à remercier chaleureusement tous les acteurs qui ont encore une fois rendu possible notre rendez-vous annuel : les conférenciers, les organisateurs, et tous nos fidèles partenaires sans qui cette manifestation ne pourrait avoir lieu. En leur nom, nous vous souhaitons une excellente conférence.

Pour le comité de programme
Benoît MARTIN (DGA-MI)

Comité de programme

Erwan AGBRALL	DGA-MI, MINDEF
José ARAUJO	ANSSI
Boris BALACHEFF	HP Labs
Christophe BIDAN	Centrale-Supélec
David BOUCART	DGA-MI, MINDEF
Yves CORREC	ARCSI
Frédéric CUPPENS	Télécom Bretagne
Jean-Luc DANGER	Télécom ParisTec
Herve DEBAR	Télécom SudParis
Ivan FONTARENSKY	Airbus Defense & Space
Patrick HEBRARD	DCNS
Ludovic JACQUIN	HP-E
Eric JAEGER	DGSIC
Benoît MARTIN	DGA-MI, MINDEF
Ludovic PIETRE-CAMBACEDES	EDF
Eric WIATROWSKI	Orange

Partenaires

Site officiel : http://www.cesar-conference.org

Tables des matières

Internet des Objets et sécrité : une mission impossible ?

Damien Cauquil

Digital Security - Paris,France
damien.cauquil@digitalsecurity.fr
https://www.digitalsecurity.fr/

Résumé Le marché des objets connectés explose depuis presque deux ans et l'on devrait atteindre les 6,4 milliards d'objets connectés d'ici à la fin de l'année 2016. Les objets connectés commencent désormais à faire partie de notre quotidien : ils s'invitent sur les infrastructures vitales ; la domotique assure des fonctions variées dans les locaux publics et privés ; les objets connectés médicaux suivent notre activité, notre santé et manipulent des données confidentielles ; les drones sont devenus monnaie courante et posent des problèmes de sécurité des personnes et des installations ; etc. La sécurité est devenue un enjeu majeur pur l'Internet des Objets.
Au travers d'une revue de plusieurs évaluations de securit effectuées par le CERT UBIK de Digital Security, cette communication abordera les nouvelles menaces auxquelles sont exposées les objets connectés, montrera par l'exemple les erreurs les plus fréquentes et présentera une synthèse des bonnes pratiques de sécurisation de ces objets.

Keywords: Internet des Objets, sécurité, vulnérabilité, Retex, bonnes pratiques, règlementation, données personnelles

1 Sécurité des objets connectés

Les objets connectés reposent sur un ensemble de technologies et de services pour leur fonctionnement : une architecture matérielle, un ou plusieurs micrologiciels, un ou plusieurs modules de communication et potentiellement des services hébergés sur Internet. Cet ensemble constitue l'écosystème d'un objet connecté, auquel doivent s'appliquer les différentes mesures de sécurité. La communication présente ces écosystèmes et les menaces applicables.

2 Base matérielle

La base matérielle est l'élément principal de l'objet connecté, habituellement une carte électronique ou informatique embarquant l'essentiel des composants permettant d'exécuter la base logicielle. Il peut s'agir de systèmes rudimentaires comme de micro-ordinateurs embarques, selon les fonctions que doit remplir l'objet ou sa durée de vie. C'est un élément important car des fonctionnalités offertes par cette dernière peut dépendre la sécurité de l'objet.

Il existe trois familles de bases matérielles très utilisées dans la conception d'objets connectés :

- les microcontrôleurs ;
- les System-on-Chip ;
- les micro-ordinateurs de poche.

Ces différentes familles permettent d'exécuter des logiciels se présentant sous différentes formes, qui seront abordées dans la section dédiée aux bases logicielles (cf. 3).

2.1 Les microcontrôleurs

Un microcontrôleur est un circuit intégré qui possède un processeur, de la mémoire et des interfaces d'entrées-sorties. Ces circuits consomment moins que des processeurs, travaillent à des fréquences plus faibles que ces derniers et possèdent une capacité de stockage limitée. Quelques exemples de familles de microcontrôleurs rencontrées dans des objets connectés :

- la famille des MSP430 de Texas Instruments ;
- la famille des STM32 et STM8 de STMicroelectronics ;
- la famille des PIC de de Microchip.

Les microcontrôleurs travaillent de manière très proche du système dans lequel ils sont intégrés, et permettent notamment de piloter des actionneurs ou de récupérer des informations de capteurs. Ils ne sont généralement pas capables d'offrir des interfaces de haut-niveau comme le protocole Bluetooth Low Energy, mais sont tout à fait en mesure de s'interfacer avec d'autres composants au travers de protocoles de communication propres au domaine électronique, tels que SPI, I2C ou UART (Universal Asynchronous Receiver Transmitter). Il est ainsi possible de réaliser des fonctions avancées grâce à l'utilisation conjointe d'un microcontrôleur et d'un composant spécialisé.

Le microcontrôleur est programmé et débogué via des interfaces et protocoles dédies, qui peuvent être propriétaires. Cela permet aux développeurs de tester leur code sur un système réel, tout en assurant l'identification de bogues et leur traçabilité.

Le microcontrôleur possède une mémoire morte, généralement de la mémoire Flash, dans laquelle est stocké le code principal qui sera exécuté. Il possède aussi de la mémoire vive, requise par le code exécuté pour effectuer ses calculs et stocker des données de manières temporaires. Le code exécutable stocke en mémoire Flash est important, car il peut constituer une propriété intellectuelle à protéger, c'est pourquoi certains microcontrôleurs implémentent des mécanismes de protection contre l'extraction du contenu de la mémoire.

De la même manière, les interfaces de débogage utilisées lors du développement du micrologiciel propre au microcontrôleur peuvent ^être désactivées afin d'empêcher toute compromission des données stockées par ce dernier.

2.2 Les System-on-Chip (SoC).

Les System-on-Chip ou SoC, ou systèmes sur une puce, se présentent sous la forme de circuits intégrés contenant un système complet permet de réaliser la ou les fonctions attendues. Ils fournissent généralement un ou plusieurs processeurs, de la mémoire, des périphériques d'interface mais aussi des périphériques dédiés. La communication entre les différents processeurs et périphériques est réalisée via des bus de communication internes.

Les SoC sont très utilisés dans les objets connectés, car ils offrent une solution clef en main permettant, par exemple, de piloter des actionneurs tout en offrant une connectivité Bluetooth Low Energy. Il en existe pour les différents protocoles de communication radiofréquence qui ont actuellement le vent en poupe, comme les technologies Sigfox ou LoRa. Il est aussi à noter que certains SoC sont basés sur

des processeurs permettant d'exécuter un système d'exploitation complet comme Linux ou Windows, tels que certains SoC d'Altera ou de Broadcom par exemple.

A l'instar des microcontrôleurs, des interfaces de programmation et de débogage sont présentes et utilisées pour la conception des micrologiciels à destination de ces composants, tout comme des interfaces de communication permettant l'interfaçage avec des circuits intégrés externes (SPI, I2C, UART).

De la même manière, ils proposent aussi des mécanismes de protection de la mémoire afin d'assurer l'intégrité du micrologiciel et protéger la propriété intellectuelle.

2.3 Les micro-ordinateurs de poche

Les micro-ordinateurs de poche sont des systèmes complets comprenant un processeur, de la mémoire vive et un support de stockage (carte SD ou micro-SD) sur lequel un système d'exploitation est présent. Ils fonctionnent exactement comme un ordinateur classique, mais possèdent un encombrement réduit, ainsi que des entrées-sorties et périphériques annexes selon les modèles.

Un des premiers micro-ordinateurs de poche popularise fut le Raspberry Pi, qui a évolué au fur et à mesure des années pour atteindre actuellement une taille minimale avec le Raspberry Pi Zero (cf. figure 1 et 2). Les fonctionnalités offertes par le Raspberry Pi ont évolué avec ce dernier, et comprennent notamment un adaptateur WiFi et Bluetooth 4.0, des ports USB et ethernet et des entrées-sorties permettant de connecter des cartes d'extension ou encore des caméras et écrans.

Fig. 1. Raspberry Pi 3, par la fondation Raspberry Pi

Fig. 2. Raspberry Pi Zero, par la fondation Raspberry Pi

L'Edison d'Intel (figure 3) est un micro-ordinateur base sur un processeur Atom d'Intel, se présentant comme un module intégrable intégrant une connectique WiFi et Bluetooth, un connecteur pour carte SD et de la mémoire Flash intégrée.

Fig. 3. Intel Edison, par Intel

Les micro-ordinateurs de poche reposent sur des systèmes d'exploitation connus, et ne proposent pas de fonctionnalités propres à la protection de la propriété intellectuelle. Seuls des mécanismes de *SecureBoot* s'ils sont présents permettent d'assurer la confidentialité du logiciel exécuté sur ces plateformes.

2.4 Exemples de vulnérabilités affectant les bases matérielles

Plusieurs vulnérabilités sont couramment observées lors de l'analyse de bases matérielles, et notamment :

- Les indications présentes sur le circuit imprime de l'objet connecté dévoilent des interfaces de débogage et la manière de s'y connecter ;
- des erreurs de conception électroniques induisant des faiblesses matérielles;
- la non-activation de mécanismes de protection de micrologiciel expose ce dernier ainsi que des secrets pouvant être stockés en mémoire (clefs de chiffrement, mots de passe, etc.).

Circuit imprimé Le circuit imprimé sur lequel sont soudés des composants comporte ce que l'on nomme couramment un masque de soie (ou « silkscreen ») qui est constitué de l'ensemble des éléments visuels sérigraphiés sur ledit circuit. La sérigraphie sert à indiquer l'emplacement des différents composants, mais dans certains cas à fournir des informations sur des points de tests utilisés notamment pour le débogage.

Certains circuits imprimés possèdent des indications explicites, indiquant de manière précise le rôle d'un point de test ou d'un emplacement de connecteur et donnant un avantage à un attaquant : il pourra alors s'interfacer électroniquement et avoir potentiellement accès à des fonctionnalités réservées aux développeurs ou à la maintenance. La figure 4 montre un exemple de ce marquage explicite : les points de test TX et RX étant utilisés dans le cadre d'une interface UART permettant l'accès à une console.

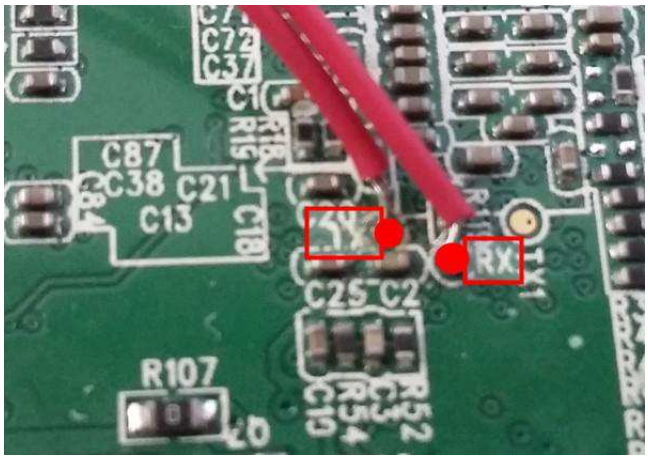


Fig. 4. Points de tests sur une camera connectée (interface UART)

Erreurs de conception électronique Cette autre forme de vulnérabilité est particulière, car ne touchant pas directement à l'aspect connecté de l'objet ni à la sécurité informatique au sens strict, mais elle est malheureusement relativement courante. Les erreurs de conception de circuit imprimé sont légion, mais certaines d'entre elles peuvent avoir un impact considérable, comme le cadenas connecté d'une célèbre marque présenté ci-après.

Ce cadenas connecté possède un unique circuit imprimé auquel est relié un moteur, ce dernier permettant de déverrouiller et verrouiller mécaniquement le cadenas. Ce moteur est alimenté par deux fils, soudés sur le circuit imprimé et encadrés par ses bornes d'alimentation (3V). L'agencement de ces fils n'a pas été pensé de manière sécurisée, car il suffit de faire contact entre la borne d'alimentation Vcc et le fil juste à côte pour déclencher le moteur et forcer le déverrouillage du cadenas ! L'utilisation d'un petit crochet suffit pour effectuer cette manœuvre, sans que cela laisse de traces. La figure 5 montre l'agencement des différents broches et fils, le trait vertical rouge marquant le contact à effectuer pour forcer l'ouverture.

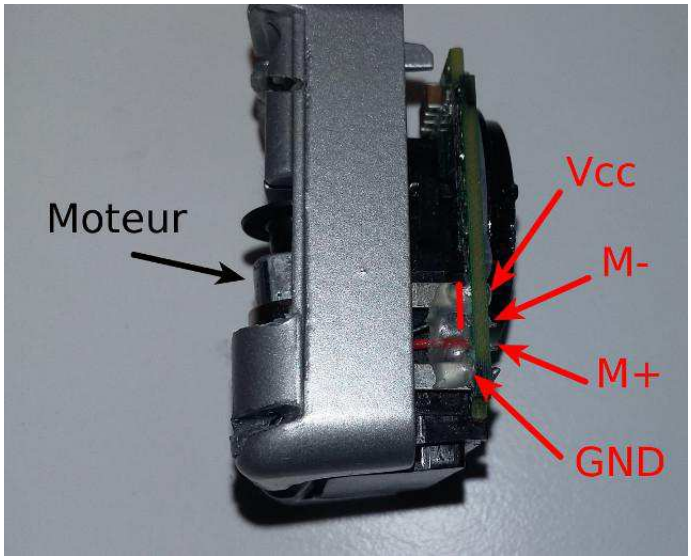


Fig. 5. Vulnérabilité physique d'un cadenas connecté

Non-activation des mécanismes de protection La plupart des SoCs et microcontrôleurs à destination des industriels fournissent un ou plusieurs mécanismes permettant d'empêcher l'extraction du code exécutable qui a été inséré dans la mémoire de ces derniers, et cela afin d'empêcher toute fuite de propriété intellectuelle ou contrefaçon**.** Ces mécanismes sont activés grâce à l'utilisation de fusibles électroniques, dont certains sont définitifs.

Il est ainsi possible de désactiver les interfaces de débogage, de limiter l'accès à la mémoire Flash, voire pour certains de ne protéger que certaines portions de mémoire. Ces restrictions sont implémentées dans les puces, et le seul moyen de les contourner consiste à attaquer physiquement la puce, ce qui nécessite des moyens et des connaissances qui ne sont pas à la portée de tous.

Cependant, de nombreux équipements connectés sont mis en vente bien que ces protections n'aient pas et activées. Dans d'autres cas plus rares, la protection de la mémoire est activée mais les fonctionnalités de débogage permettent d'extraire le contenu de la mémoire Flash grâce aux registres du processeur, comme la technique décrite dans [INCLUDESECURITY-DUMP], par exemple.

3 Base logicielle

La base logicielle est tributaire de la base matérielle et agit à la fois comme fournisseur des services de base de l'objet connecté, superviseur de l'état de l'objet et barrière de sécurité. Cette base logicielle peut être sujette à des erreurs de conception ou de développement, pouvant aboutir à une compromission de la sécurité.

Là encore on distingue différents types de bases logicielles :

- les micrologiciels monolithiques, à destination des microcontrôleurs et SoC ;
- les micrologiciels hétérogènes, à destination de certains SoC et micro-ordinateurs de poche

Un micrologiciel est un logiciel embarqué constitué d'instructions et de structures de données à destination de microcontrôleurs ou de microprocesseurs. Il est stocké en mémoire de manière persistante, et lancé à chaque démarrage de la base matérielle.

3.1 Les micrologiciels monolithiques.

Ces micrologiciels se présentent sous la forme d'un exécutable unique exécuté par un processeur, qui effectue tout un ensemble de tâches. Il peut dans certains rares cas être en mesure de gérer des tâches multiples, en particulier sur des bases matérielles destinées au traitement en temps réel. Ce type de micrologiciel ne repose pas sur un système d'exploitation, il est le seul système existant et actif sur la plateforme matérielle.

L'exploitation de vulnérabilités applicatives présentes dans les micrologiciels monolithiques n'est pas triviale, car elle dépend d'une part de l'architecture du processeur propre à la base matérielle et d'autre part sur le type de vulnérabilité en question. Techniquement parlant, un attaquant peut prendre le contrôle du flux d'exécution du micrologiciel, mais difficilement en tirer facilement avantage.

3.2 Les micrologiciels hétérogènes

Ces micrologiciels se présentent sous la forme d'un ensemble d'exécutables exécutés par un système d'exploitation, afin de réaliser une ou plusieurs tâches. Ces exécutables sont stockés en mémoire, à l'aide d'un système de fichiers. Ce dernier peut être connu (comme CRAMFS par exemple) ou propriétaire.

Des vulnérabilités applicatives peuvent être présentes dans un ou plusieurs des exécutables, et être exploitées afin d'obtenir un accès privilégié au système d'exploitation : console à distance, élévation de privilèges, lancement d'un service, etc.

3.3 Exemples de vulnérabilités affectant les bases logicielles

Différentes vulnérabilités sont couramment observées lors de l'analyse de micrologiciels, et notamment :

- Des erreurs d'implémentations aboutissant à des débordements de zone mémoire (*buffer overflows*) ;
- des implémentations cryptographiques non-standard possédant des vulnérabilités ;
- des mots de passe ou clefs de chiffrement par défaut ;
- des implémentations de protocoles ne respectant pas les standards définis.

Débordement de zone mémoire (*buffer overflow*) Durant l'étude d'une caméra connectée, nous avons identifié une portion de code reposant sur une fonction non-sécurisée et sujette à un débordement de zone mémoire (*buffer overflow*). Celle-ci est utilisée dans le traitement de données en provenance d'un service réseau distant, et la portion de code en question peut être facilement atteinte par un attaquant communiquant avec ledit service.

Le code en question est illustre par la figure 6. Un attaquant peut prendre le contrôle de la caméra à l'aide de cette vulnérabilité, et ainsi contrôler le système Linux sous-jacent.

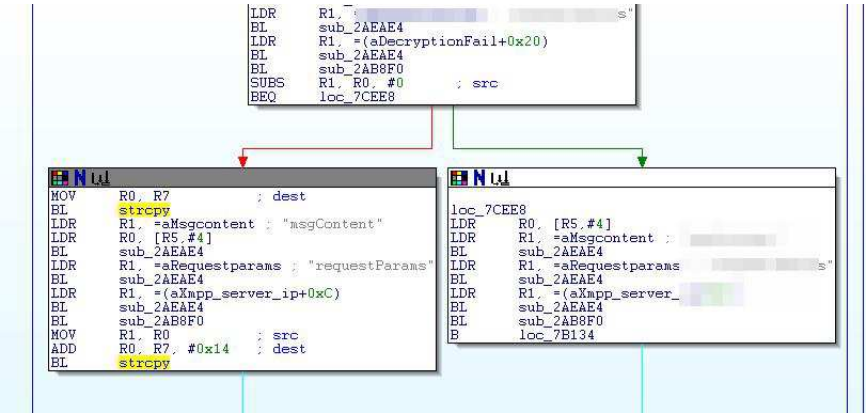


Fig. 6. Débordement de zone mémoire via un appel à strcpy

La mise en œuvre d'un cycle de développement sécurisé permettrait d'assurer le respect des bonnes pratiques de développement et de tester avant livraison du code la sécurité de celui-ci.
Algorithme de chiffrement « maison » Lors de l'étude de la serrure connectée Okidokeys [DC24-BLELOCKS], Anthony Rose et Ben Ramsey (Merculite Secu

rity) ont identifié une faiblesse cryptographique permettant de forcer l'ouverture de celle-ci et de la rendre par la suite inutilisable par les utilisateurs autorisés.
Le constructeur de la serrure affirmait notamment utiliser des technologies de chiffrement hautement sécurisées, similaires aux standards employés par les banques et l'armée (AES 256 bits et authentification 3D Secure), couplées avec des solutions cryptographiques brevetées et prouvées.
La serrure connectée en question utilisait des jetons à usage unique (OTP) pour l'ouverture et la fermeture de cette dernière, mais les deux chercheurs ont démontré que la modification d'un seul octet d'un jeton valide déjà utilisé suffisait à forcer l'ouverture de la serrure et la désynchroniser.
Il semblerait qu'une partie de l'algorithme de chiffrement repose sur une opération de type ou exclusif (XOR), qui influe sur le résultat du déchiffrement et provoque le comportement observé.

L'utilisation d'un algorithme standardisé et éprouvé aurait permis d'éviter cette vulnérabilité.

Non-respect des standards dans l'implémentation de protocoles Un fabricant d'un System-on-Chip dédié au protocole ZigBee (IEEE 802.15.4) fournit un kit de développement standard à ses clients afin que ces derniers puissent développer des micrologiciels pour le SoC en question.

Le protocole ZigBee permet de connecter différents équipements au sein d'un réseau WAN, en sécurisant les communications avec un chiffrement AES. Chaque équipement peut avoir un ou plusieurs rôles parmi les suivants :

- Trust Center (TC) : l'équipement ayant ce rôle assure la gestion de la sécurité (distribution des clefs et authentification des équipements) ;
- Coordinateur : cet équipement unique dans un réseau assure la gestion globale du réseau WAN et des équipements qui y sont connectés (il a aussi habituellement le rôle de TC) ;
- Routeur : l'équipement ayant ce rôle peut relayer des informations en provenance d'autres équipements, permettant la création de réseaux maillés ;
- Equipement final : l'équipement ayant ce rôle est connecté au réseau et communique avec d'autres équipements connectés.

Lors de l'ajout d'un équipement à un réseau ZigBee existant, ce dernier envoie une demande d'association afin que le coordinateur puisse lui attribuer une adresse au sein du réseau. Le coordinateur doit être placé en mode association pour accepter le nouvel équipement. Si un chiffrement est imposé, trois cas sont envisagés :

- l'équipement possède déjà une clef de chiffrement réseau (NWK) préprogrammée et aucune négociation de clef n'est réalisée ;
- l'équipement ne possède pas de clef de chiffrement réseau mais possède une clef de chiffrement de transport (TK), auquel cas le Trust Center lui communique la clef réseau chiffrée avec la clef de transport ;
- l'équipement ne possède aucune clef de chiffrement, et le Trust Center lui communique la clef de chiffrement réseau de manière non-chiffrée.

Cette étape d'ajout d'équipement n'est réalisée qu'une seule fois, le coordinateur et l'équipement mémorisant la clef de chiffrement réseau une fois l'association effectuée.

Malheureusement, le kit de développement fourni par le fabricant possède une vulnérabilité permettant d'effectuer une demande d'association bien que le coordinateur ne soit pas place en mode association, il est alors possible d'intercepter la clef de chiffrement réseau (NWK) et de s'ajouter à ce dernier. Une fois l'accès au réseau obtenu, il est possible de prendre le contrôle des équipements qui y sont connectés. Il se trouve que ce SoC est utilisé dans plusieurs solutions d'ampoules

connectées, qui sont donc toutes vulnérables car basées sur le même kit de développement.

Des tests fonctionnels et de sécurité auraient permis de mettre à jour cette faiblesse, connue depuis 2009, et de la corriger.

4 Couche de communication

La couche de communication est l'élément liant l'objet à son monde extérieur, et implique l'emploi de protocoles de communication avec ou sans fil. Cette couche peut être source de problèmes de sécurité qu'elle repose sur des protocoles de communication sans-fil (*802.15.4, Bluetooth Low Energy* ou protocoles propriétaires) ou classiques (TCP/IP, UDP, MQTT).

On distingue deux facteurs de vulnérabilités dans les protocoles de communication : celles qui sont inhérentes au protocole (des faiblesses de conception) et celles qui sont induites par une mauvaise utilisation du protocole de communication. Nous aborderons ces deux types de vulnérabilités dans la suite de la communication.

Seuls les protocoles de communication radio-fréquence seront abordés dans cette communication, car ce sont les plus utilisés dans le domaine de l'Internet des Objets, dont notamment :

- *Bluetooth Smart ;*
- *Enhanced ShockBurst ;*
- *SIGFOX ;*
- *LoRa ;*
- *LoRaWAN ;*
- *ZigBee ;*

4.1 ***Bluetooth Smart (Bluetooth Low Energy)***.

Le protocole *Bluetooth* Smart fait partie de la norme *Bluetooth* depuis sa version 4.0, et vise à permettre l'établissement de canaux de communication entre différents équipements en assurant une consommation d'énergie minimale et une simplicité d'utilisation. Ce protocole est supporté par la grande majorité des équipements mobiles actuels (ordinateurs portables et ordiphones), ce qui en fait un protocole de choix pour la communication avec des objets connectés. Les équipements médicaux connectes conçus ces dernières années reposent de plus en plus sur ce protocole de communication.

Ce protocole fournit des mécanismes de sécurité permettant d'assurer le chiffrement des communications et de créer des liens de confiance entre équipements (on parle d'appairage). Ces fonctionnalités permettent d'empêcher les attaques de l'homme du milieu (*Man-in-the-Middle*) et les écoutes passives.

4.2 Enhanced ShockBurst

Le protocole *Enhanced ShockBurs*t a été conçu par Nordic Semiconductor, et est intégré par défaut dans différents System-on-Chip produits par ce dernier. Il s'agit d'un protocole utilisant la bande ISM de 2.4GHz, peu gourmand en énergie et qui permet d'assurer le transport d'information à faible portée. Il est très utilisé dans la conception de souris et claviers sans-fil par exemple, mais on le retrouve aussi dans d'autres équipements de contrôle d'accès par exemple.

Ce protocole de communication n'offre aucune protection particulière en termes de confidentialité, et peut être écouté à l'aide d'une astuce découverte par un chercheur en sécurité (Travis Goodspeed, [GOODSPEED]).

4.3 SIGFOX

Le protocole de communication SIGFOX est d'origine toulousaine, créé par la société éponyme, et fait partie de ce que l'on nomme couramment les LPWAN (*Low Power Wide-Area Network*, les réseaux longue portée pour équipements à faible puissance). Le réseau SIGFOX est déployé dans pas moins de 24 pays.

Ce protocole ne fournit aucune confidentialité des données : aucun chiffrement n'est en place et il est difficile d'implémenter de l'AES vu que la charge utile est de 12 octets (il en faudrait 16 pour réaliser un chiffrement AES solide).

4.4 *LoRa* et *LoRaWAN*

Le protocole de communication LoRa est la technologie sur laquelle repose le protocole *LoRaWAN*, un réseau LPWAN géré par la LoRa Alliance. Il s'agit d'un réseau concurrent de SIGFOX, utilisant une technologie différente et plus résiliente.

Le protocole *LoRaWAN* utilise du chiffrement AES par défaut, assurant la confidentialité des communications. Contrairement aux idées reçues, le protocole *LoRa* utilisé pour faire de la communication de machine à machine n'emploie pas de chiffrement, à l'instar du protocole SIGFOX.

4.5 ZigBee

Le protocole de communication *ZigBee* est basé sur la norme IEEE 802.15.4 et permet la création de réseaux personnels sans-fil à base d'équipements de faible puissance. Cette technologie est utilisée en domotique, dans certaines installations industrielles ainsi que dans les smart buildings.

La norme *ZigBee* fournit tous les mécanismes de sécurité nécessaires pour assurer l'intégrité et la confidentialité des échanges. Néanmoins, des différences d'implémentations sont observées selon les fabricants de modules compatibles, à cause des nombreuses versions de la norme qui utilisent des spécifications différentes.

4.6 Exemples de vulnérabilités couramment observées sur la couche de communication

La couche de communication constitue généralement le point faible des objets connectés : l'accès aux données transmises est aisé et conditionné à l'utilisation de matériel adéquat. Depuis quelques années, la radio logicielle est devenu abordable grâce à divers équipements peu coûteux comme le *HackRF One*, le *Yard Stick One*, ou encore le *Blade RF*.

La radio logicielle, ou SDR, est définie comme le traitement numérique de signaux radios, permettant de démoduler, décoder, ré-encoder et moduler des protocoles de communication à l'aide de logiciels dédies. De cette manière, il est possible de réaliser des attaques automatisées sur des protocoles radio complexes, à faible coût.

Il existe aussi des équipements spécialisés pour certains protocoles de communication difficile à intercepter, comme le *Bluetooth Smart* ou encore le protocole *Enhanced ShockBurst*.

Les vulnérabilités rencontrées lors de l'analyse des communications entre objets connectes incluent notamment :

- La transmission en clair de données sensibles ;
- l'absence d'authentification des équipements ;
- l'absence de contrôles d'intégrité ;
- un chiffrement faible ou pouvant être cassé.

Transmission de données sensibles Durant l'analyse d'un cadenas connecté, nous avons intercepté l'ensemble des communications entre l'application mobile Android et le cadenas en question, et avons découvert que celui-ci transmettait en clair l'ensemble des informations.

Lors de la première utilisation du cadenas, l'application mobile demande à l'utilisateur de saisir le code de 8 chiffres associé au cadenas, par défaut 12345678. Une fois ce code saisi et validé par l'application, l'utilisateur peut le modifier. Le nouveau code est transmis au cadenas de manière transparente, qui le stocke en mémoire et s'en servira pour de futures authentifications.

Le souci, c'est que la communication entre l'ordiphone et le cadenas se base sur le protocole *Bluetooth Smart* sans utiliser les mécanismes de sécurité assurant le chiffrement. Un attaquant écoutant la communication peut ainsi découvrir le code du cadenas, transmis en clair par l'application mobile à chaque utilisation, comme le montre la figure 7.

Connected			
write	fff0	fff3	05 .v .i .r .t .u 00 00 00 00 00 00 00 00
read	180f	2a19	.Y
write	1805	2a2b	bb .p 8a 1f
read	fff0	fff3	05 .v .i .r .t .u 00 00 00 00 00 00 00 00
write	ffd0	ffd6	00 12 34 56 78 00 00 00 00
notification	ffd0	ffd7	01 ff
read	180a	2a26	05 .B 01 01 20 16 03 06 00 00
read	ffd0	ffd8	03
notification	ffd0	ffda	00
read	ffd0	ffda	00
write	ffd0	ffd9	01

Fig. 7. Code confidentiel transmis en clair

L'utilisation d'appairage tel que défini dans les spécifications *Bluetooth Smart* permet d'assurer le chiffrement des communications et une protection contre l'interception passive de données confidentielles.

Absence d'authentification des équipements L'analyse d'un glucomètre connecté communiquant via *Bluetooth Smart* nous a permis de mettre en évidence l'absence d'authentification des équipements. En effet, l'application mobile répertorie le ou les glucomètres d'un patient et les identifie par leurs adresses matérielles *Bluetooth*. Aucune authentification n'est prévue par l'application, et il est relativement aisé de simuler un équipement possédant les mêmes caractéristiques.

Nous avons donc créé un tel équipement simulé, lui avons attribué la même adresse matérielle et avons réussi à forcer l'application mobile à s'y connecter. Il nous a ensuite été possible de fournir des mesures erronées à cette dernière, qui les a importées dans le journal des mesures stocké sur l'ordiphone. Sachant que cette application peut être connectée à des services de télésanté (*eHealth*), il est alors possible d'impacter les diagnostiques et les prescriptions qui en découlent, cela pouvant avoir un impact non-négligeable sur la santé du patient.

L'authentification des équipements est pourtant un mécanisme fourni par la norme *Bluetooth Smart*, englobé dans la fonctionnalité d'appairage. En effet, cette dernière permet de déployer sur les deux équipements appairés une clef de chiffrement unique et des informations sur l'identité de ces équipements. Ces derniers ont alors les moyens de valider leurs identités lors de l'établissement d'une connexion, et de refuser de se connecter à un équipement ne pouvant pas prouver son identité.

5 Informatique en nuage

La plupart des objets connectés communiquent au final avec un ou plusieurs services héberges sur Internet, qui manipulent et stockent les données transmises par ces objets et permettent leur mise à disposition aux utilisateurs légitimes. Cette centralisation du stockage de données est, en cas d'intrusion, une opportunité pour un attaquant d'accéder à l'ensemble des informations des objets connectés utilisant ce service.

Les services hébergés sur Internet peuvent être de types variés et reposer sur des protocoles différents, cependant on distingue deux grandes familles de protocoles :

- les protocoles relatifs aux interfaces de programmation avancées, offrant l'accès à des données propres à l'utilisateur et stockées sur du long terme ;
- les protocoles de communication en temps-réel, utilisés par les applications et équipements pour la transmission de données et la notification à très faible latence.

Ces services sont utilisés à la fois par les applications mobiles ou de bureau à destination des utilisateurs, et par certains équipements ayant accès à Internet ou par la passerelle si ces derniers ne peuvent pas directement communiquer avec l'environnement extérieur.

Des mécanismes de sécurité existent sur l'ensemble de ces protocoles permettant d'assurer la confidentialité, comme le protocole TLS qui permet d'assurer la confidentialité des échanges lorsqu'il est correctement mis en œuvre.

5.1 Contournement de chiffrement TLS

Lors d'une récente analyse de caméra connectée, nous avons déterminé que celle-ci communiquait avec un serveur de contrôle via le protocole XMPP (*eX-tensible Messaging and Presence Protocol*), et que le chiffrement par TLS était requis. Il nous a tout de même été possible de désactiver ce chiffrement en abaissant le niveau de sécurité.

Pour ce faire, nous avons utilisé l'outil striptls[1] et avons forcé le client XMPP à annoncer au serveur qu'il ne supportait pas de chiffrement TLS : ce dernier a accepté une connexion non-sécurisée tandis que le client n'avait aucun moyen de vérifier l'identité du serveur. Ni le client, ni le serveur n'implémentait de vérification du niveau de chiffrement, permettant à un pirate d'intercepter les identifiants de connexion.

5.2 Enumération des ressources connectées

Le protocole XMPP est utilisé pour faire communiquer des équipements en temps-réel, et le serveur XMPP sert de point de relais central. Le service d'annuaire était actif sur ce serveur, et a donc permis d'énumérer tous les équipements qui y étaient connectés, sans aucune restriction, comme le montre la figure 8.

[1] https ://github.com/tintinweb/striptls

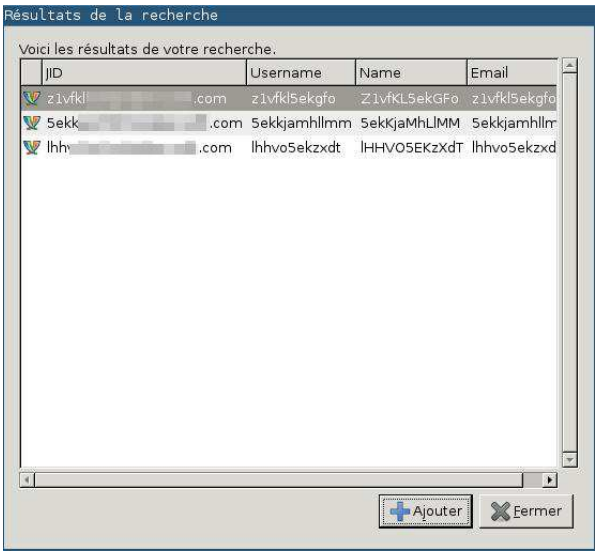


Fig. 8. Découverte des équipements par le service d'annuaire XMPP

6 Synthèse des erreurs les plus courantes

Cette synthèse est basée sur une vingtaine d'analyses de solutions connectées effectuées au sein du laboratoire Recherche & Développement de *Digital Security*.

Les erreurs et vulnérabilités les plus courantes peuvent être classées dans les catégories suivantes, de la plus courante à la moins courante :

1. Authentification faible ou inexistante
2. Mots de passe par défaut
3. Chiffrement faible ou inexistant
4. Fonctionnalités de débogage activées
5. Absence de fonctionnalité de mise à jour
6. Absence de chiffrement/signature des mises à jour
7. Mauvaise gestion des erreurs
8. Absence de protection physique
9. Absence de fonctionnalité de remise à zéro

6.1 Difficulté et coût de correction

Certaines erreurs de conception induisent une difficulté et un coût importants, telle que l'absence de fonctionnalité de mise à jour. En effet, si un équipement a et conçu sans possibilité de mettre à jour son micrologiciel, alors le seul moyen de corriger un problème qui l'impacterait consisterait à :

- rappeler tous les équipements impactes a n de mettre à jour leurs micrologiciels (via une interface de programmation) ;
- remplacer les équipements impactes s'il est impossible de les mettre à jour par programmation.

Un exemple qui a marqué les esprits concerne la société Chrysler. Suite à une attaque trouvée par les chercheurs en sécurité Charlie Miller et Chris Valasek et exposée à la conférence mondiale de sécurité informatique DEF CON en 2015, cette dernière a rappelé pas moins de 1,4 millions de véhicules pour la mise à jour d'un composant interne impliqué dans la chaîne d'exploitation que les deux chercheurs ont mis en œuvre.
En parallèle, les voitures connectées fabriquées par Tesla ont elles aussi et ciblées par des chercheurs en sécurité et plusieurs vulnérabilités ont et identifiées. Pourtant, aucun rappel de véhicules n'a été nécessaire, les concepteurs de la voiture ayant mis au point un système de mise à jour over-the-air, c'est-à-dire qui utilise une connectivité sans-fil pour transmettre automatiquement le nouveau micrologiciel et corriger les vulnérabilités.

Dans le cas du glucomètre connecté présenté dans cette communication, et c'est vrai pour une grande majorité de solutions médicales connectées, les principales vulnérabilités viennent du fait que les concepteurs ont porté un protocole de communication existant utilisé pour la communication par interface série sur un protocole de communication sans-fil, nonobstant les implications en terme de sécurité des échanges. Ce type d'erreur est tout de même coûteux car le modèle testé n'a pas de fonctionnalités de mise à jour, et un rappel est donc nécessaire pour corriger les problèmes rencontrés.

6.2 Impact de ces vulnérabilités sur la réputation des entreprises

Le monde des chercheurs en sécurité est actuellement centré sur les solutions connectées, et notamment les véhicules autonomes, les systèmes de contrôle d'accès, les équipements médicaux et industriels. Ces solutions connectées, contrairement à un pèse-personne connecté ou à une smartwatch ont la particularité d'avoir des répercussions dans le monde réel : on peut envisager de prendre le contrôle de ces équipements et de porter atteinte à la sécurité physique et l'intégrité de personnes. Les entreprises commercialisant ces types de solutions et dont les produits seraient vulnérables ont leur réputation en jeu sur des marchés porteurs à forte croissance : il s'agit de ne plus faire l'autruche et de considérer la sécurité comme un des arguments essentiels tant pour la vente que pour la sécurité des biens et des personnes.

7 Repenser la sécurité à l'ère des objets connectés

Les solutions connectées et leur interaction avec les systèmes d'information nous amènent à repenser la sécurité de ces derniers en prenant en compte les risques qu'elles introduisent. On distingue notamment les différents cas d'usage de ces solutions connectées :

- les solutions connectées intégrées au système d'information et déployées en entreprise, comme par exemple les systèmes de contrôle d'accès ;
- les solutions connectées intégrées au système d'information et déployées en dehors de l'entreprise, comme par exemple les *smart meters* ou les capteurs connectés ;

- les solutions connectées utilisant le système d'information comme moyen d'accéder à Internet, comme des centrales d'alarme ou encore des ampoules connectées ;

7.1 De nouveaux vecteurs d'attaque

L'introduction de nouveaux vecteurs d'attaque par des solutions connectées a un impact non négligeable sur le risque encouru par les systèmes d'information avec lesquels ces solutions sont interfacées. La variété des vecteurs d'attaque et le risque associé dépendent des configurations de ces solutions.

Les passerelles Une grande majorité de solutions connectées reposent sur des passerelles matérielles pour faire le lien entre les objets constitutifs de la solution et un réseau traditionnel pour accéder à Internet. Il s'agit dans le cas présent d'un élément de la solution connectée qui utilise le système d'information afin que cette dernière puisse accéder aux environnements hébergés sur Internet, et assurer leur bon fonctionnement.

Une passerelle est donc capable de communiquer avec un ou plusieurs objets connectés via un protocole de communication adapté *(ZigBee, Bluetooth Smart, ShockBurst,* etc.) mais aussi avec les équipements réseaux traditionnels afin d'obtenir un accès Internet (WiFi ou ethernet la plupart du temps). De plus, elle permet le passage d'information de l'un à l'autre, en appliquant ses propres contrôles de sécurité, s'il y en a.

L'ajout d'équipement tiers dans un système d'information et la gestion de la sécurité relative à cet équipement est une problématique bien connue des responsables sécurité, tout comme les technologies sur lesquelles reposent la communication entre l'équipement et le réseau existant. Mais ce n'est pas le cas des protocoles de communication employés par la passerelle pour communiquer avec les équipements connectés : ceux-ci peuvent être documentés voire propriétaires, et ne sont que très rarement indiquées par les constructeurs. Cela fait partie de la magie de l'Internet des Objets : on crée et fournit des solutions connectées permettant de réaliser une ou plusieurs fonctions, en cachant les aspects techniques.

Prenons en exemple une solution d'ampoules connectées. Ces ampoules communiquent avec une passerelle mais aussi entre elles (via un réseau maillé), passerelle qui est alimentée et connectée au réseau d'entreprise via le WiFi. La passerelle a été placée dans un réseau à part (VLAN), séparé du réseau d'entreprise, et à accès à Internet. Le réseau WiFi qu'elle utilise est sécurisé par un mot de passe unique, utilisé seulement pour ce réseau. Les bonnes pratiques sont respectées.

En réalité, les ampoules communiquent entre elles et avec la passerelle grâce au protocole ZigBee, dont la sécurité repose sur une clef de chiffrement unique. Le constructeur a mis cela en place pour une facilité d'utilisation : n'importe quelle ampoule compatible du commerce peut être installée et ajoutée rapidement à celles déjà supervisées par la passerelle. Mais un attaquant, qui a au préalable acheté le même modèle d'ampoule et réussi à extraire la clef de chiffrement, peut aisément se connecter au réseau d'ampoules (en considérant qu'il est à sa portée) et communiquer avec la passerelle et les ampoules.

De là, plusieurs scenarios sont possibles :

- l'attaquant prend le contrôle des ampoules et les pilote à distance, sans passer par Internet ;
- l'attaquant attaque la passerelle et réussit à extraire le code protégeant l'accès au réseau sans-fil WiFi ;

— l'attaquant compromet la passerelle et rebondit sur le réseau auquel cette passerelle a accès.

Il n'y a aucune surveillance en place sur le réseau d'ampoules reposant sur le protocole ZigBee, ce qui implique aucune traçabilité de l'attaque. Si durant l'attaque la passerelle est compromise et sert de relais, il est probable que le trafic malveillant soit repéré et la passerelle incriminée. Sachant que cette dernière est cloisonnée du réseau de l'entreprise, il est peu probable que l'attaquant puisse s'en prendre aux systèmes internes. Le pire scenario possible reste donc la prise de contrôle des ampoules : le risque peut être important selon le rôle que jouent ces ampoules.

Les équipements sur site client (CPE) L'utilisation par un client d'une solution connectée raccordée à l'infrastructure d'un opérateur expose à la fois le client et l'opérateur. Cette problématique est bien connue et se retrouve notamment dans le cadre des offres triple-play des fournisseurs d'accès à Internet par exemple, qui fournissent des « box » pour leurs clients qui seront raccordées à son réseau de manière presque automatique.

Cependant, l'utilisation de réseaux novateurs pour relier ces équipements aux réseaux des opérateurs et le fait que ces derniers soient placés chez le client augmentent les risques : un attaquant peut tout à fait arriver à prendre le contrôle de l'équipement placé chez le client voire accéder aux systèmes communiquant avec ces équipements, en particulier dans le cas de protocoles de communication sans-fil. L'utilisation de réseaux bas-débit à longue portée, tels SIGFOX ou LoRaWAN, permet le déploiement chez les clients de terminaux capables de transmettre de l'information en quantité limitée à des centres de collecte, voire d'en recevoir selon la technologie choisie. L'inconvénient des réseaux sans-fil est qu'en l'absence de chiffrement et de signature des données échangées par ce biais, il est possible d'une part d'injecter des données à destination de l'un ou l'autre des équipements communiquant, mais aussi d'intercepter les données échangées. Ces données étant traitées par la suite par un ou plusieurs centres de collecte, il est possible qu'un attaquant puisse en tirer profit.
Prenons le cas des compteurs connectés, ces équipements installés chez les clients de fournisseurs d'énergie électrique visant à permettre une meilleure régulation de la production d'énergie. Ces compteurs connectés remontent des informations au fournisseur lui permettant d'une part d'assurer la facturation, et d'autre part d'estimer les besoins en énergie du client afin d'adapter la production. Il est aussi envisageable de penser que le compteur connecté puisse être administré de manière distante par le fournisseur d'énergie, permettant ainsi de modifier des options liées à l'abonnement ou des limites en place. Dans ce contexte, si un attaquant est en capacité d'intercepter des échanges entre le compteur et le fournisseur, voire même d'envoyer des données qu'il maîtrise et qui sont considérées comme valides à l'un ou l'autre, il est alors possible que ce dernier puisse d'une part altérer le comportement du compteur, et d'autre part fournir des données erronées au fournisseur. Si l'attaquant est en plus en mesure d'usurper l'identité de plusieurs compteurs, il peut mener une attaque plus massive et impacter la production d'énergie. Le tout en manipulant simplement des protocoles de communication sans-fil, à l'aide d'équipements spécialisés.
Dans le cadre des réseaux bas-débit à longue portée, la surveillance est possible de la part des opérateurs si des attaques de grande ampleur sont réalisées à l'encontre des fournisseurs de services, mais reste très difficile à mettre en œuvre lorsque celles-ci ciblent un ou plusieurs équipements en place chez des clients, de manière localisée.

7.2 Un environnement inhabituel

L'installation d'équipements en environnement hostile impose de prendre un ensemble de précautions fondamentales afin de réduire les risques associés à ce type de configuration.

En premier lieu, l'équipement connecté étant généralement de petite taille, le vol de ce dernier par une personne malintentionnée est un risque à envisager. Il est donc primordial que cet équipement soit peu coûteux, afin d'être facilement remplacé et de limiter le risque de vol. L'équipement peut stocker en mémoire des clefs de chiffrement, qui selon les bases matérielles peuvent être plus ou moins compliquées à extraire. Ces clefs de chiffrement permettent de communiquer avec d'autres équipements, voire d'envoyer des messages compréhensibles par des opérateurs de réseaux bas-débit à longue portée, il est donc primordial de les protéger.

Dans certains cas, ces clefs de chiffrement sont des clefs maîtresses, c'est-à-dire qu'elles sont uniques et déployées dans tous les équipements devant communiquer entre eux : la connaissance d'une clef maîtresse permet de compromettre la sécurité de l'ensemble des équipements connectés, et d'avoir un impact conséquent.

C'est le cycle de *provisionning* de ces équipements qui est à concevoir et mettre en place de façon à apporter un maximum de sécurité, dans un premier temps. En effet, de la même façon que l'on cloisonne les réseaux et que l'on utilise des mots de passe dédiés à différents comptes et usages, les clefs de chiffrement installées dans les équipements connectés doivent être propres à l'équipement. De cette manière, seul l'équipement attaqué est impacté par l'extraction des clefs de chiffrement, et un attaquant pourra dans le pire des cas usurper l'identité de ce seul équipement, limitant de fait le risque encouru.

De plus, l'utilisation de composants dédiés au stockage des données sensibles comme les *Secure Elements* permet d'éviter toute extraction d'information sensible et d'assurer la confidentialité des éléments stockés par ces derniers. La combinaison de clefs propres à un équipement et de stockage sécurisé limite grandement les risques liés au vol et à l'usurpation.

7.3 Traçabilité et journalisation

Enfin, les équipements connectés actuellement conçus et employés ne permettent pas pour la plupart le suivi de leur utilisation ou des erreurs rencontrées, contrairement aux équipements traditionnels placés au sein d'un système d'information. Il y a un manque flagrant de journalisation permettant d'assurer une traçabilité des actions et des utilisateurs.

Cette lacune est compréhensible, du fait des capacités de stockage réduites de ces solutions et équipements connectés, bien qu'elles n'en soient pas pour autant dépourvues. Globalement, il s'agit d'une fonctionnalité souvent oubliée lors des phases de conception mais qui pourrait être très utile dans le cadre d'investigations liées à ces solutions connectées. C'est une pratique courante dans le monde de l'informatique (tous les services et systèmes produisent des journaux contenant des traces d'événements), mais très peu répandue dans le domaine de l'Internet des Objets, à tort.

La faible capacité de stockage n'empêche en aucun cas la conservation d'informations liées aux événements propres à un équipement connecté, du moins d'un certain nombre d'entre eux. De plus, les mécanismes de communication sans-fil peuvent être employés, s'ils sont correctement sécurises, pour transmettre de manière régulière des informations de journalisation à des équipements tiers dédiés à cet usage, et ce afin de faciliter l'identification des origines d'un incident, et de pouvoir imputer celui-ci à un équipement ou une personne.

Enfin, la possibilité de surveiller un ou plusieurs protocoles de communication sans-fil permettrait de détecter et dans une certaine mesure de prévenir des incidents de sécurité, à l'instar des systèmes de détection et de prévention qui sont omniprésents dans les systèmes d'information actuels. Là encore, des efforts sont à faire pour mettre au point ces systèmes et envisager d'avoir une vision suffisante des interactions entre équipements pour être capable de déceler des attaques.

8 Des obligations légales et réglementaires à prendre en compte

L'objet connecté peut être destiné à faciliter la vie de ses utilisateurs, en particulier dans le domaine de la santé et de la *mesure de soi*. Cet usage particulier amené l'objet connecté à conserver des informations personnelles, dont l'usage est encadré en France par la loi informatique et libertés (voire dans certains cas par le code de la sante publique) qui impose de garantir la sécurité des données. La résistance à l'environnement, la durée de vie et les contraintes de coût peuvent impacter considérablement la sécurité d'un objet connecté installe. Il convient donc de gérer les risques tout au long du cycle de vie des objets et d'adopter des mesures adaptées à chaque étape de la vie d'un objet.

8.1 Du bon usage du chiffrement

Afin d'assurer la confidentialité des données stockées par un équipement connecté, l'utilisation de chiffrement est obligatoire. L'emploi d'un algorithme sûr et éprouvé est de rigueur, tel que le standard choisi par *le National Institute of Standard and Technology* americain et l'Agence Nationale pour la Sécurité des Systèmes d'Information : **AES** avec une clef de taille 128 bits au minimum, comme recommandé dans l'annexe B1 du Référentiel Général de Sécurité [RGS-ANNEXE-B1] établi par l'ANSSI.

Cependant, l'algorithme de chiffrement ne fait pas tout, le choix et le stockage de la clef sont des éléments aussi importants : s'ils sont mal choisis, la confidentialité des données peut être remise en question. En ce qui concerne la génération de la clef, il est recommandé d'utiliser des générateurs pseudo-aléatoires cryptographiques, dont certains ont été implémentés pour des microcontrôleurs. Quant au stockage, l'utilisation d'un *Secure Element* est de mise, afin d'offrir une solide protection contre l'extraction.

8.2 Transitivité

Les équipements médicaux connectes ont vocation à être prêtés aux patients, et donc à stocker des données relatives à différents patients successivement. Il est donc impératif pour ce type d'équipement d'offrir une fonctionnalité assurant l'effacement complet sécurisé des données stockées, afin de pouvoir le transmettre à un autre patient sans mettre en danger les données relatives au patient précédent.

Cette fonctionnalité d'effacement de données, ou de remise à zéro, est essentielle pour assurer la transitivité. Les standards d'effacement de données existent, et doivent être suivis scrupuleusement pour assurer une suppression efficace. Les mémoires de stockage actuelles reposant en grande partie sur la technologie Flash, l'effacement complet des données est fastidieux. C'est pourquoi l'utilisation de cartes de stockage séparées est recommandée afin de permettre la réutilisation d'un équipement médical par un autre patient dans les délais les plus courts, le nombre d'équipements connectés étant généralement limité.

8.3 Anonymisation

Les données de l'équipement transmises à des applications tierces hébergées sur Internet doivent être anonymisées, de manière à ne pas pouvoir faire le lien entre l'identité du patient et ces données. Le patient se voit ainsi attribuer un numéro unique l'identifiant, utilisé à la fois sur l'équipement connecté et sur les services en ligne, offrant la garantie au patient que personne ne puisse faire le lien entre son identité (nom, prénom, date de naissance) et les pathologies pour lesquelles il est traité.

Cette anonymisation est une condition nécessaire pour pouvoir manipuler des données de santé conformément aux dispositions de la loi Informatique et Liberté (article 8), comme l'indique le guide à destination des professionnels de la santé [CNIL-GUIDEPROSANTE].

9 Conclusion

Les solutions connectées, de par leur complexité et les variétés de technologies sur lesquelles elles reposent, requièrent une rigueur particulière dans les phases de conception, d'implémentation et de diffusion afin d'assurer la sécurité des données et réseaux auxquelles elles ont accès.
Les différentes technologies de communication sans-fil couplées à la dispersion physique des équipements connectés amènent de nouveaux vecteurs d'attaque à prendre en considération lors de la conception des solutions connectées ainsi que lors de leur utilisation. Certaines solutions connectées peuvent être assimilées à des boîtes noires dont les technologies sont propriétaires, peu testées ; les implications en termes de sécurité sont donc impossibles à évaluer et les protections mises en place moins efficaces. Une meilleure connaissance des technologies et plus de transparence de la part des fournisseurs de solutions amélioreraient grandement la sécurité des systèmes et réseaux.

De plus, un manque flagrant de traçabilité et de possibilité de surveillance a été observé dans la plupart des solutions connectées, rendant de fait très difficile les investigations suite à un incident de sécurité. Ces derniers étant en constante augmentation, l'instauration de mécanismes de journalisation et de gestion des erreurs permettrait de détecter de potentiels incidents, et dans la mesure du possible les prévenir.

De nouvelles normes et référentiels de sécurité sont en cours d'élaboration par divers organismes ([AFNOR], et [NIST] notamment), qui prennent en compte les spécificités de l'Internet des Objets afin de pouvoir garantir un certain niveau de sécurité en ce qui concerne les solutions connectées. Ces normes et certifications en sont à leurs balbutiements et devraient s'affiner dans les années à venir pour finalement être une base permettant l'appréciation de la sécurité des équipements et solutions connectées, afin de sécuriser au mieux ces dernières lors de leur intégration au système d'information.

Références

[INCLUDESECURITY-DUMP] Firmware dumping technique for an ARM Cortex-M0 SoC, Include Security : http ://blog.includesecurity.com/2015/11/NordicSemi-ARM-SoC-Firmware-dumping-technique.html

[DC24-BLELOCKS] Picking Bluetooth Low Energy Locks from a Quater Mile Away, Athony Rose & Ben Ramsey, DEF CON 24 : https://media.defcon.org/DEF%20CON%2024/DEF%20CON%2024%20presentations/DEFCON-24-Rose-Ramsey-Picking-Bluetooth-Low-Energy-Locks.pdf

[GOODSPEED] Promiscuity is the nRF24L01+'s Duty, Travis Goodspeed http://travisgoodspeed.blogspot.fr/2011/02/promiscuity-is-nrf24l01s-duty.html

[RGS-ANNEXE-B1] ANSSI Annexe B1 au Référentiel Général de Sécurité https://references.modernisation.gouv.fr/sites/default/ les/RGS_Mecanismes_cryptographiques_v1_20.pdf

[CNIL-GUIDEPROSANTE] CNIL Guide Professionnels Santé https://www.cnil.fr/sites/default/ les/typo/document/CNIL- Guide professionnels de sante.pdf

[AFNOR] AFNOR, Internet of Things : http://norminfo.afnor.org/structure/119570

[NIST] NIST, NIST's Network-of-Things Model Builds Foundation to Help Define the Internet of Things : https://www.nist.gov/news-events/news/2016/07/nists-network-things-model-builds-foundation-help-define-internet-things

Entreprise et IoT un mélange instable

Xavier AGHINA

ORANGE Labs
xavier.aghina@orange.com

Résumé L'utilisateur dispose d'une gamme de petits appareils informatiques, dont l'utilisation fait maintenant partie de sa vie quotidienne et qui forme un environnement pervasif. Cela permet à des objets de se reconnaître et de se localiser automatiquement avec une capacité d'intelligence distribuée. Mais au vu de ces possibilités avancées, qu'en est-il réellement, de la sécurité de ces systèmes qui peuvent être instrumentés et le seront à plus ou moins brève échéance. Les risques majeurs sont d'ajouter ou de voir des centaines d'appareils « non traditionnels » sur son réseau d'entreprise, exécutant des systèmes d'exploitation différents, et utilisant des protocoles propriétaires. Ces appareils ne disposent pas des ressources avancées pour prendre en charge la sécurité, ces risques ne doivent plus être ignorés, ils deviennent un défi pour l'entreprise. Les risques de sécurité associés aux dispositifs IoT ont le potentiel de devenir le risque de sécurité le plus important du réseau : l'entreprise n'a aucune visibilité sur le niveau de protection à mettre en œuvre et ne dispose de la technologie en adéquation pour évaluer la sécurité.

Keywords : Sécurité, Protection, Entreprise, Menaces, Vulnérabilité, Réseaux, Données personnelles

1 Introduction et définition

Mais qu'est-ce que l'IoT ? Il existe de nombreuses façons de décrire l'IoT, plus de 20 groupes professionnels et de recherche ont travaillé pour le caractériser, mais jusqu'à présent, il n'y a pas une définition universellement acceptée (Procter&Gamble -1999, CGEIET, UIT, Gartner, Wikipedia). En revanche, ils décrivent tous essentiellement le concept d'un même univers croissant de dispositifs connectés à Internet qui devraient permettre de créer de nouvelles économies d'affaires, de nouvelles sources de revenus et de nouvelles opportunités commerciales. Ces dispositifs sont également censés apporter un nouveau paradigme impliquant défis et risques de sécurité pour l'entreprise, ces systèmes seront au cœur de nos infrastructures critiques et formeront la base de nos futurs services intelligents.

L'Internet des Objets (IoT) peut être défini comme « un réseau omniprésent qui permet la surveillance et le contrôle de l'environnement physique par la collecte, le traitement et l'analyse des données générées par des capteurs ou des objets intelligents ».

L'informatique ubiquitaire peut être considéré comme la troisième ère de l'histoire de l'informatique, qui succède à l'ère des ordinateurs personnels et à celle des mainframes. A l'ère de l'informatique pervasif, l'utilisateur a à sa disposition une gamme de petits appareils informatiques dont l'utilisation fait partie de sa vie quotidienne.

La capacité de l'intelligence embarquée et distribuée dans le réseau est un élément architectural de base de l'IoT pour trois raisons principales:

- **Collecte de données** : La collecte de données centralisée et la gestion des objets intelligents ne fournissent pas l'évolutivité requise par l'IoT. Par exemple,

la gestion de plusieurs millions de capteurs et actionneurs dans un réseau - Smart Grid - ne peut pas être efficacement faite en utilisant une approche centralisée.

- **Préservation des ressources réseau** : Parce que la bande passante du réseau peut être rare et la collecte de données environnementales à partir d'un point central dans le réseau, mène inévitablement par l'aide d'une capacité réseau plus importante.
- **Fonctionnement en Boucle fermée** : Pour certains cas d'utilisation, l'IoT nécessite des temps de réaction réduits. Par exemple, l'envoi d'une alarme via plusieurs sauts à partir d'un capteur vers un système centralisé (qui gère l'analyse) avant d'envoyer une commande à un actionneur entraînerait des retards inacceptables.

Cette capacité d'intelligence distribuée est connue comme « Fog Computing » : extension du paradigme du cloud. Il est semblable à l'informatique en nuage, mais plus « terre-à-terre ». L'architecture « Fog Computing » étend le cloud dans le monde physique des choses. L'effet de « brouillard informatique » sur le cloud computing et les grands systèmes de données peut varier avec une limitation dans la distribution précise de contenu, aboutissant à la création de mesures pour améliorer la précision.

L'immense intérêt de l'Internet des objets, extension moderne des réseaux de capteurs et d'actionneurs, réside dans ses capacités :

- à être partout (ubiquité)
- à « sentir », apprendre de leur environnement (intelligence)
- à communiquer/informer
- à prendre des décisions, seul

Mais au vu de ces possibilités avancées qu'en est-il, réellement, de la sécurité envisageable sur ces systèmes qui peuvent être tous instrumentés et le seront à plus ou moins brève échéance.

Les risques d'ajouter des centaines d'appareils « non traditionnels » à son réseau - périphériques qui exécutent différents systèmes d'exploitation, - utilisent un certain nombre de protocoles propriétaires et souvent ne disposent pas des ressources avancées pour les configurations de sécurité, ne doivent plus maintenant être ignorés. Ces quelques préoccupations ciblent la conception du réseau et la stratégie d'approvisionnement, avec la notion « d'appairage ».

Verizon 2015 Data Breach Investigations Report [http://www.verizonenterprise.com/verizon-insights-lab/dbir/] fait valoir que toutes les vulnérabilités signalées à ce jour ont été identifiées par les chercheurs. Pourtant, la rapidité d'évolution de l'IoT avec un éventail de menaces plus large que la moyenne pourrait changer à tout moment le périmètre de la sécurité. Les enquêtes auprès des réseaux de professionnels montrent leurs inquiétudes, en identifiant la sécurité comme l'un des plus grands défis de l'IoT.

Sur le réseau d'entreprise, vous avez besoin de savoir « qui » est sur votre réseau afin de le sécuriser, cela semble en apparence assez simple, mais pour de nombreux départements informatiques, trouver et arrêter les dispositifs corrompus est un jeu interminable, rendant l'entreprise exposé aux menaces.

2 Imaginons une journée « connecté »

La journée de Charlotte s'était achevée hier par une séance d'entraînement fractionné pour progresser en running, avec l'aide incontournable de sa montre connectée, lui servant de coach, de GPS pour le parcours, sans oublier le cardiofréquencemètre pour optimiser ses performances.
Pourtant le réveil de Charlotte se fait en douceur comme tous les jours depuis qu'elle utilise des capteurs placés sur le matelas, planifiant le sommeil et les réveils en actionnant la lampe par une diffusion progressive de sons et lumière. Dans le même temps la salle de bain est à la bonne température, avec son programme musical préféré. La cafetière est prête au bon moment pour le petit déjeuner et Charlotte c'est déjà surprise plusieurs fois à reposer la p te à tartiner dans le réfrigérateur connecté, sachant que tout le monde aurait connaissance de ce petit écart. Elle a d'ailleurs perdu 2 kilos depuis que le bracelet et capteurs mesurent ainsi sa vie, non pas qu'elle l'ait voulue activement ou qu'elle pensait en avoir besoin, mais elle a reçu de nombreuses félicitations, notamment de son assurance santé, étant partie intégrante des services financés par l'entreprise.

En prenant sa voiture électrique pour se rendre au travail, celle-ci en calculant le temps de trajet, décale de 15 minutes le début d'une réunion en notifiant les participants. Arrivée au bureau, alors qu'elle appuie sur le bouton de l'ascenseur, son bracelet vibre. Sur l'écran « et si vous preniez l'escalier ? ». De retour de réunion, Charlotte en approchant de l'ordinateur, a ouvert automatiquement la session et peut directement consulter le compte rendu. Après un déjeuner au restaurant d'entreprise calibré sur son nouvel objectif nutritionnel, Charlotte se replonge dans son travail avec intensité tout l'après-midi, petite vibration : « Vous êtes stressée, faites donc une pause » et profite pour se dégourdir les jambes vers les espaces détentes. Une forte vibration, c'est le livreur qui demande l'accès à son domicile pour déposer les vêtements de retour du pressing, Charlotte envoie une commande à la serrure afin que le livreur puis ouvrir la porte à l'aide de son smartphone le temps de déposer les vêtements.

La fin de journée approche, Charlotte se rend compte à 18h qu'un espace dans son agenda a été bloqué par l'application santé et que, si elle le souhaite, 27 % des machines de la salle de gym sont actuellement libres.
Avant de rejoindre son domicile, Charlotte reçoit une notification à proximité de l'hyper marché, pour récupérer la commande faite par le réfrigérateur pour le repas du soir.

Une fois chez elle, Charlotte s'accorde quelques minutes devant son écran mural. Pendant les pubs son capteur vibre très légèrement : « Nous aimerions avoir votre avis sur les publicités en cours de diffusion. Vous obtiendrez des points et des promotions personnalisés ! Restez devant la caméra de votre écran pour accepter » et ignore l'appel.

Cela ressemble à un scénario futuriste, mais toutes les technologies sont déjà disponibles, il est inéluctable que les objets connectés vont faire irruption dans un contexte personnel et implicitement dans l'entreprise et devront être gérés au mieux qu'ils soient fournis par elle-même ou portés par l'employé.

3 Diversité et cas d'utilisation de l'IoT

Les applications de l'internet des objets se traduisent par de nombreux usages concrets – nouveaux ou améliorés – impactant significativement le quotidien des individus et des entreprises. Les bénéfices potentiels attendus facilitent son adoption par cette diversité d'usages :

- Logements et lieux de travail deviennent plus intelligents, plus faciles à gérer et moins coûteux à l'usage, ils offrent notamment des possibilités de contrôle des consommations énergétiques, d'intégration des systèmes de sécurité et de confort accrus.
- Des téléphones intelligents qui permettent l'accès sans fil aux services d'information n'importe où dans le monde.
- La santé connectée, incluant le segment « bien-être » auquel le grand public est le plus sensibilisé, notamment grâce aux « wearables » (vêtement, accessoire portable avec électronique et informatique).
- Des cartes à puce qui servent de pièce d'identité pour permettre l'accès aux informations confidentielles, aux espaces et salles protégés.
- Des petits appareils domestiques qui peuvent être manipulés avec un navigateur web.
- Les décodeurs, les télévisions interactives et les consoles de jeu qui permettent l'accès à des services en ligne.
- Des appareils embarqués dans les véhicules : ordinateur de bord, système de navigation, de péage de diagnostic ou téléphone incorporé
- Le milieu agricole l'utilise dans son processus de production avec des capteurs d'état du végétal, des animaux, embarqués sur les machines agricoles, des outils d'aide à la décision ou de guidage des engins.

Sans la confiance des entreprises le développement de l'IoT en son sein serait limité si les problématiques suivantes ne sont pas adressées :

1. la complétude, l'exactitude, la fiabilité et l'intégrité des données
2. la protection des données personnelles et la propriété des données des entreprises
3. la sécurité et la résilience des infrastructures de connectivité

4 l'IoT en entreprise : risques et menaces

Avant de définir quels sont les risques de l'IoT en entreprise, voici une cartographie des vulnérabilités et des menaces naturelles propre à cet environnement :

- **Systèmes d'exploitation :** nombreux, ils ont pour la plupart été lancés en 2015 : Brillo (Google), LiteOS (Huawei), Windows 10 IoT Core (lancé en partenariat avec Raspberry), Mbed OS (ARM). Ils n'ont pas de référentiels connus de sécurité et utilisent beaucoup de protocoles propriétaires, Linux n'est pas à l'abri de vulnérabilités exploitables.
- **Architectures :** sont très hétérogènes, c'est à la fois dangereux et sécurisant car ce sont les plateformes les plus utilisées qui sont le plus attaquées en général comme nous l'avons vu avec Android.
- **Sécurité physique :** souvent compromise, les objets connectés peuvent être corrompus physiquement et relativement facilement dans pas mal de situations, l'objet étant « à portée de la main » cela permet une analyse fine des composants hardware et des connecteurs pour une prise en main du système d'exploitation en local**.**
- **Intégrité logicielle :** la mise à jour des objets n'est pas garantie, notamment en raison des failles de sécurité des réseaux sans fil utilisés. La sécurité des données stockées ne l'est pas plus au niveau des serveurs. Cela peut compromettre la vie privée des utilisateurs même si celle-ci est déjà menacée par l'usage de leurs données fait par des sociétés commerciales classiques telles que Google ou par des « hackers ».
- **Accès :** les objets connectés peuvent être piratés pour accéder aux réseaux dans lesquels ils sont intégrés comme au sein des entreprises, pour exemple le système de climatisation servant de « passerelle » pour accéder au S.I d'une entreprise.
- **Chiffrement :** Une bonne part des objets connectés contenant des capteurs utilise de faibles ressources, coûte peu chère, et dispose de simples micro-

contrôleurs en guise de CPU, ce qui limite l'usage de la cryptographie pour chiffrer leurs communications**.**

- **Réseaux M2M :** des vulnérabilités sont déjà détectées dans les systèmes du marché, et notamment dans celui de Sigfox, avec la faille « Digital Security » (code CRC récupérable, et le numéro de périphérique diffusé en clair).

Un des premiers risques des objets connectés en entreprise va être le « shadow IoT » qui va se traduire par une auto-organisation réalisé et mise en œuvre au sein de l'organisation sans approbation de la DSI ; aboutissant à des règles non adéquates car ubiquitaire :

- Des contraintes opérationnelles fortes
- Une offre logicielle interne limitée
- Une multitude de services mêlant outils personnels et d'entreprise avec des logiciels gratuits, peu onéreux
- Des risques avérés : conservation de données à l'extérieur de l'entreprise, logiciels non sécurisés,
- risque de contamination, risques de backdoors, risque d'indisponibilité
- Des complications RH
- Des problèmes légaux : contrefaçons, vol – perte de données, conservation des données personnelles

L'univers de l'IoT représente pour l'entreprise une nouvelle « grande » surface d'attaque parce que tous les type de dispositifs sont souvent connectés à des ordinateurs portables, smartphone ou tablettes qui sont ensuite connectés aux réseaux de l'entreprise. Pour ces raisons l'IoT augmentera la surface d'attaque, depuis les dispositifs il sera facile pour un attaquant de bénéficier d'un accès au réseau par la non prise en charge de contrôles de sécurité autour des dispositifs se connectant en entreprise.

La raison pour laquelle de nombreuses entreprises sont relativement «indifférentes» à propos de la sécurité est qu'elles pensent avoir résolu le problème de la sécurité, parce qu'elles croient à tort qu'elles ne disposent pas de données valorisables ou que les données précieuses sont enfermés en lieu sûr, le reste n'a pas d'importance.

Les affirmations précédentes sont incorrectes dans le sens ou les attaquants ciblent délibérément les entreprises avec une sécurité plus faible et une fois à l'intérieur de réseau, ils sont en mesure de trouver et voler les informations les plus importantes à l'activité de l'entreprise.

Les risques imbriqués de sécurité se posent lorsque les entreprises commencent à déployer l'Internet des Objets pour leur propre usage et que les employés apportent ces « objets » sur le réseau d'entreprise pour leur usage personnel. Il faut évaluer l'impact émergent des dispositifs IoT sur l'entreprise afin de réduire les risques d'usages en entreprise.

Parce que les risques de sécurité associés aux smartphones, tablettes et ordinateurs portables sont relativement bien compris, par les études sur le Bring Your Own Device, qui dans le contexte professionnel pose des questions relatives à la sécurité de l'information et à la protection des données, ainsi que sociales et juridiques.
Les consommateurs adoptent appareils portables et intelligents pour le domicile; certains d'entre eux commencent à apparaître sur les réseaux d'entreprise. Comme les appareils IoT s'incrustent dans la vie de tous les jours des consommateurs et implicitement des entreprises, les défis de la sécurité et des risques associés à ces dispositifs deviendront plus complexes.
Quelques exemples de menace propre à l'entreprise :

- Les employés apportent des dispositifs IoT dans le réseau sans contrôle par cette dernière
- Des API de l'entreprise sont exposées sans réellement le savoir
- Les collecteur/ capteur d'information et passerelle interconnexion sont accessibles depuis Internet

Une nouvelle dimension de la sécurité : l'IoT est au carrefour entre Internet et le monde physique. Cela a des implications graves sur la sécurité, la menace d'attaque se déplace par manipulation de l'information pour commander l'actionneur (en d'autres termes, le déplacement du numérique au monde physique). Par conséquent, il augmente considérablement la surface d'attaque avec des menaces et des dispositifs connus, vers des attaques de sécurité supplémentaires des nouveaux appareils, des protocoles et des flux de communications.

L'IoT peut être aussi affectée par diverses catégories de menaces de sécurité, notamment avec les exemples suivants:

- Les vers / virus ordinaires passant de l' IT pour l'IoT: En général limités à des usages en cours d'exécution comme les O/S: Windows, Linux, iOS, Android.
- Des «Script kiddies» ou équivalent, ciblant l'IoT résidentiel: webcams non protégées, vol de contenu ou la perte de contrôle sur le système de la maison.
- Le crime organisé: l'accès à la propriété intellectuelle, le sabotage, et l'espionnage.
- Le Cyber-terrorisme : Les centrales nucléaires (par exemple, le virus Stuxnet), la surveillance du trafic, les chemins de fer ou les infrastructures essentielles.
- Les « Botnets IoT » trouveront des millions de nouvelles recrues sous la forme d'appareils zombies ou autres dans le cas où ils communiquent entre eux, peuvent s'influencés mutuellement sans un contrôle de cohérence dédiée. Cf Attaque DDOS d'OVH par des caméras connectées.
- Les dispositifs pourraient ouvrir des « backdoors » dans votre réseau, faciliter la propagation des logiciels malveillants, stocker de l'information sensible de l'entreprise, et conduire à des conditions de déni de service.
- Les attaques DNS de l'infrastructure d'entreprise peuvent provoquer des dysfonctionnements – « DNS spoofing »en ajoutant des dispositifs IoT sur le réseau d'entreprise, le service DNS est incontournable.

Dans le cas d'employés qui apportent des dispositifs IoT dans le réseau, il est possible de sécuriser l'entreprise. Mais qu'en est-il de tous ces appareils connectés sur des réseaux distants externes de type cloud que les employés utilisent le «BYOIoT» (Bring Your Own IoT) en référence au précèdent BYOD. Les dispositifs IoT qui ont déjà fait leur chemin sur la scène du réseau d'entreprise, et il faut faire en sorte que le réseau ne soit pas dégradé et sans fuite de données confidentielles due à la connexion d'un périphérique compromis, les entreprises doivent aussi penser à ce type scénario.

Les salariés en télétravail de la maison ou en mobilité, disposant d'appareils IoT de type « wearable » ou dans leur domicile contrôlée avec une application à partir d'un smartphone ou une tablette, souvent ciblés par des logiciels malveillants. Dans le cas où celui-ci utilise une passerelle Wi-Fi pour se connecter au réseau d'entreprise, cela représente un point névralgique à contrôler pour la sécurité du S.I. Le risque de contamination croisée des réseaux de la maison peut être très grave à moins que les entreprises appliquent strictement des contrôles de sécurité. Malheureusement, la plupart des personnes supposent que les VPN règlent tous problèmes de connexion à distance mais cela est tout simplement faux.

Beaucoup de dispositifs IoT sont brancher sur les ordinateurs via USB pour la recharge de batterie, USB est un vecteur d'attaque très commun. Il est assez facile d'imaginer un malware IoT conçu pour connaître et exploiter des vulnérabilités dans USB. La politique de sécurité et/ou le système des postes informatiques sont-

ils en mesures de se protéger contre ce type de menace ? anti-virus / malware couvrent-ils ce risque ?

Les réseaux en cours de déploiement – LoRa, Sigfox – vont permettre d'établir un champ de communication entre tous les objets connectés y compris ceux qui vont être déployés en entreprise ou porter en nombre par les salariés eux-mêmes, permettant d'exfiltrer tous types d'informations personnelles ou professionnelles, sans réel contrôle possible.

Alors que la consommation est actuellement axée sur les dispositifs IoT qui présente à minima un risque direct pour l'entreprise, beaucoup d'entre eux se connectent avec un backend d'infrastructure du fournisseur via Internet pour stocker les données des utilisateurs. Des attaques réussies contre ces backend infrastructures pourraient fournir aux attaquants des informations d'identification de l'utilisateur et d'autres informations qui pourraient leur permettre de prendre la main sur le réseau domestique de l'employé. De là, il est tout à fait possible pour un attaquant d'installer keyloggers ou autre malware conçus pour voler les informations d'identifications de l'utilisateur nécessaires pour ouvrir une session sur le réseau d'entreprise.
En général, les personnes sous-estiment sérieusement comment il est facile pour un attaquant, de se déplacer à l'intérieur des réseaux, une fois qu'ils ont accès.

5 Des pistes de protections

La réalité est qu'il y a relativement peu d'indications sur la façon de configurer des périphériques du réseau et il est étonnant de s'apercevoir du manque de normes de configuration, en particulier dans le domaine de l'IoT, mais également pour ICS (partage de connexion internet) et les périphériques SCADA. Alors que nous trouvons des guides de durcissement hardware (ANSSI, NSA,..) fournissant des informations utiles sur la manière de configurer les systèmes d'exploitation, les pare-feu, les commutateurs et les principaux composants du réseau, il y a un manque surprenant de normes de configuration pour tout le reste.
Cette absence de ressources contribue à un manque général de confiance dans la gestion de la configuration sécurisée et limite de façon spectaculaire l'utilisation de l'automatisation pour valider et maintenir des configurations sécurisées.

Pour commencer « Tout ce qui est déployé sur le réseau, vous devez être en mesure de le gérer » L'un des points de discussion récents a été autour de si oui ou non l'entreprise moyenne était à même d'avoir le niveau d'exigence et le temps nécessaire pour gérer l'IoT. Les composants disponibles pour une architecture de sécurité sont :

- **Autorisation :** identification des dispositifs
  - contrôle d'accès au dispositif
  - échange des informations appropriées

- **Authentification :** gestion des dispositifs
  - relation de confiance à un système distant
  - nom d'utilisateur et mot de passe, jeton ou biométrie
- **Politique de sécurité réseau imposées** : protection des communications
  - contrôle de la circulation, de la gestion ou des données réelles
  - mécanismes mis en place pour sécuriser l'infrastructure de réseau
  - Protection des communications par le chiffrement
- **Analyse de sécurité :** Contrôle – Visibilité du système
  - analyse sécurisée qui définit les services par lesquels tous les éléments pouvant participer à fournir la télémétrie utile, dans le but de gagner en visibilité sur les objets

Le principe est de proposer une architecture réduisant les risques face aux menaces, en constituant un cadre de sécurité de base et la mise en œuvre des services de sécurité appropriés pouvant être sélectionnés pour la renforcer.

Les défis de sécurité et les considérations essentielles dans la conception et l'élaboration des dispositifs ou des systèmes IoT sont les suivants :

- Ressources mémoire et de calcul, peuvent ne pas supporter des algorithmes de sécurité complexes
- Sans connectivité, il n'y a pas de sauvegarde
- Nécessite la gestion de la sécurité à distance

Les exigences « complexes » de sécurité pouvant être déployées sur une plate-forme avec des ressources potentiellement limitées:

- Authentification sur plusieurs réseaux en toute sécurité
- S'assurer que les données sont disponibles
- Gestion de l'accès concurrentiel des données
- Gérer la vie privée entre plusieurs utilisateurs
- Fournir l'authentification forte et la protection des données (intégrité et confidentialité)
- Maintenir la disponibilité des données ou du service
- Prévoir l'évolution face aux risques inconnus
- Intérêt primordial et/ou particulier
- Disponibilité des données est d'une importance primordiale -> la température
- Déni de service -> prendre les mesures appropriées en temps réel
- Techniques d'apprentissage automatique (machine learning) - par exemple, en comparant un état de fonctionnement normal à un état d'attaque déjà appris

Les entreprises ont du mal à suivre la sécurité de leur propre système d'information sur les réseaux traditionnels. Il en résulte une représentation peu lisible impliquant une posture de sécurité qui n'est pas en adéquation avec la réalité du réseau. Avec l'IoT, ces questions deviennent un défi encore plus important.

Vous devez être en mesure de quantifier les risques en effectuant une analyse déterminant la probabilité et l'impact lorsque les menaces exploitent des vulnérabilités sur l'IoT. Un bon programme de sécurité type « BYOD » peut non seulement donner une bonne indication sur des évènements à venir, mais aussi servir de bases pour appliquer la politique de sécurité de l'IoT. Est-ce que la gestion des services de sécurité actuelle est en mesure d'accueillir ces systèmes ?

Voici quelques modèles de protections permettant de réduire les risques :

Protection des communications : La protection de la communication nécessite le chiffrement et l'authentification des périphériques pour savoir si oui ou non ils peuvent faire confiance à un système distant. Avec les technologies les plus récentes comme l'utilisation des courbes elliptiques, le travail de cryptographie et dix fois plus performante que ses prédécesseurs aux ressources limitées. Sans construire une PKI au sens strict, l'utilisation d'une autorité de certification (AC), pour intégrer des "certificat de périphérique" permettrait une authentification mutuelle sur un large éventail de dispositifs. De nombreuses normes ont été élaborées pour faciliter le déploiement d'authentification mutuelle, il existe des normes pour les formats de certificats, et les AC qui supportent à la fois les formats de certificats standards et personnalisés. Dans la plupart des cas, les certificats peuvent être facilement gérés sur l'air (OTA) par la norme des protocoles tels que Simple Certificate Enrollment Protocol (SCEP), Enrollment over Secure Transport (EST), et Online Certificate Status Protocol (OCSP).

Avec une AC forte permettant de gérer les certificats, les clés, et les informations d'identification, l'authentification réelle peut facilement être faite par des normes strictes comme la sécurité de la couche transport (TLS) et Datagram TLS (DTLS) à SSL. L'authentification mutuelle, où les deux extrémités authentifient l'autre, est cruciale pour la sécurité des systèmes IoT de bout en bout. En prime, une fois l'authentification TLS ou de DTLS est terminée, les deux extrémités peuvent échanger ou dériver des clés de chiffrement pour commencer communication qui ne peut être déchiffré par écoute clandestine.

Protection des périphériques : les dispositifs de protection contre les attaques exigent à la fois la signature du code exécutable, pour être sûr que l'ensemble du code est autorisé et protégé pendant l'exécution, pour être sûr que les attaques malveillantes ne remplacent pas le code après son chargement. La signature du code à l'aide de la cryptographie assure que le code n'a pas été modifié après avoir été «signé» et sans danger pour l'appareil, il peut être fait au niveau "application" et/ou "firmware", même dans des dispositifs avec juste un firmware monolithique. Tout dispositifs critiques, un capteur, un concentrateur, ou toute autre objet, doivent être configurés pour fonctionner uniquement avec du code signé et ne jamais exécuter du code non signé. Il est dangereux d'accepter des données provenant soit des dispositifs non vérifiées ou services non vérifiées. Ces données peuvent corrompre ou compromettre vos appareils, et donner le contrôle de ces appareils à un parti malveillant qui veut vous nuire ou nuire à autrui à travers vous.

Gestion des périphériques : inévitablement des vulnérabilités seront éventuellement découverts dans des dispositifs qui devront ensuite être « patchés » longtemps après leur déploiement pour une mise à jour nécessaire. Personne ne peut demander à contrôler physiquement la mise à jour chaque dispositif, en particulier si cela implique une flotte de camions. Pour ces raisons, la technologie over-the air (OTA) de gestion doit être intégrée dans les équipements.

Comprendre votre système : bien sûr, peu importe comment vous verrouillez l'ensemble et la façon dont vous gérez votre systèmes, certaines menaces peuvent vaincre toutes les contre-mesures pour établir une connexion dans vos systèmes. Pour ces raisons, il est essentiel d'avoir une capacité d'analyse de sécurité IoT, qui vous aidera à mieux comprendre votre réseau et les notifications sur anomalies qui pourraient être suspectes ou dangereuses, malveillantes ou non.

Contexte et évolution : la plupart des appareils IoT sont "fermés". Les clients ne peuvent pas ajouter de logiciels de sécurité aux dispositifs ayant quitté l'usine, dans le cas contraire cette manipulation annule la garantie. Pour ces raisons, la sécurité doit être intégrée dans des dispositifs afin qu'ils soient «sécurisés par conception » (secure by design). En d'autres termes, pour l'IoT, la sécurité doit évoluer de la sécurité "traditionnelle" des systèmes existants, tels que les serveurs et les ordinateurs personnels (PC) ordinateurs portables et ordinateurs de bureau. La sécurité doit évoluer pour être "construite" dans le système avant qu'il ne quitte l'usine, avec des capacités de mises à jour.

Pour résumer, ne pas lésiné sur les investissements de sécurité, obtenir des capteurs sécurisés des entreprises de bonne réputation, utiliser des systèmes isolés autant que possible, réduire le trafic et le stockage des données, utiliser des certificats de confiance efficaces, utiliser la tokenization, et adopter le chiffrement de bout en bout, et plus facilement les employés devraient être formés et sensibilisé de la façon d'utiliser les dispositifs qui ont accès depuis le réseau d'entreprise.

Une autre piste est de mettre en œuvre un programme de gestion des périphériques mobiles (MDM) programme qui se compose de politiques de sécurité pour superviser qui a accès aux périphériques mobiles de l'entreprise, ainsi que les recommandations qui précisent comment les dispositifs sont sécurisés, suivis et gérés. En mettant en œuvre MDM, les entreprises peuvent restreindre le vecteur potentiel d'attaque associé avec les nouvelles données IoT entrant et veiller à ce que les attaques ne sont pas introduites dans le réseau. Il existe aussi un programme de gestion des applications mobiles (MAM) qui permet aux entreprises de faire respecter la sécurité des données IoT au niveau de l'application et peut à ce que celles-ci gèrent les données comme le précise la politique de sécurité. Par exemple, le logiciel MAM peut arrêter le transfert de données d'un périphérique à un autre. En utilisant ces outils, non seulement les entreprises peuvent mieux gérer leurs informations issues des équipements en mettant en œuvre des restrictions sur ce que l'application peut faire, mais les données IoT sont également chiffrées en sortant du dispositif de manière à protéger les données reçues par cette application.

Les chefs d'entreprise doivent identifier où leur organisation pourrait être vulnérable par le biais d'une analyse des scénarios d'attaque perturbateurs, et l'impact financier et non financier d'une attaque sur l'organisation ainsi que les utilisateurs.

Sensibiliser les consommateurs ainsi que le personnel de première ligne dans les meilleures pratiques de sécurité Les organisations doivent informer et éduquer les consommateurs sur les meilleures pratiques, y compris les mots de passe changent régulièrement, ce qui est encore l'une des causes les plus fréquentes d'une violation de la sécurité, et offrant des conseils sur les correctifs de sécurité.

La complexité est un des plus grands obstacles à l'efficacité la sécurité, et l'IoT va sans aucun doute augmenter de façon exponentielle pour les organisations, grandes et petites. Les personnes en charge vont avoir un rôle important dans la mise en œuvre de la sécurité en faisant « plus et mieux, et encore plus vite ».

En dépit de leurs manques de confiance dans la configuration sécurisée des périphériques IoT, les professionnels de l'informatique ne s'attendent pas à recevoir un budget supplémentaire pour réduire les risques de sécurité associés à l'IoT.

Il est une grande tâche que d'assurer le fonctionnement de l'Internet des objets, cela reste une étape clé consistant à déterminer qui est exactement le responsable. Habituellement, ces dispositifs de petites tailles ne disposent pas de puissance de calcul et de services nécessaires pour une protection particulière dédiée à la sécurité, comme par exemple une ampoule connectée.

Nous allons devoir trouver le moyen de protéger les données dans ce type d'environnement, que ce soit sur l'IoT ou dans un emplacement intermédiaire. Nous allons devoir combiner les capacités de gestion des identités et des actifs de gestion, parce que les utilisateurs vont devenir gestionnaires de leurs réseaux, de leurs clouds personnels au sein de l'entreprise. L'IoT que vous « embarquez » avec vous et celle que vous avez à la maison, se comporte comme un cloud de dispositifs qui vous entourent avec sa propre identité et comme les objets ont également une identité, la question est de savoir comment garder les relations entre vous et le l'identité de ces objets. Il reste beaucoup de travail pour encadrer les politiques sécurités en fonction du contexte et de l'état ou se trouve l'objet.

La nouvelle technologie apporte de nouvelles opportunités, il est temps de profiter de la technologie Big Data pour améliorer la sécurité IoT. La sécurité Algorithmic est le niveau suivant, et implique l'utilisation de techniques d'analyse Big Data sur les millions de points de données qui peuvent être collectées à partir, par exemple, les systèmes d'un aéroport IT: capteurs de porte, badges employés, tableaux de bord, les horaires de nettoyage, systèmes de bagages et plus encore.

En utilisant des algorithmes prédictifs, le système peut apprendre ce qu'est une journée «normale» à l'aéroport par exemple, puis l'alerte lorsque les conditions diffèrent du modèle attendu.

Les entreprises ont besoin de politiques de gouvernance, de risques et de conformité efficaces qui évaluent en permanence et mette à jour votre sécurité.

6 Protection des données personnelles

La légalité de stocker des données IoT : êtes-vous légalement autorisé à stocker les données que vous venez de créer ou de collecter, comme l'information « personnelle » auprès des employés de l'entreprise. Si vous recueillez des données Il est donc important de prendre en compte, où elles sont stockées, et s'il est légal de conserver cette information. Par exemple l'arrivée des objets connectés au sein des entreprises : aux USA, de nombreuses sociétés ont investi, entre autres, dans des traqueurs d'activités, afin d'encourager leurs employés à avoir une meilleure hygiène de vie. On peut y voir un aspect positif, celui de la préservation du bien-être et des économies réalisées mais aussi un aspect négatif, celui de l'intrusion de la vie professionnelle dans la sphère privée. En effet, il est simple, avec ce genre d'objets connectés, de connaître votre hygiène de vie et vos habitudes. Ainsi, si un patron prend connaissance de ces informations, il peut avoir un a priori négatif de son employé et créer de cette manière une sorte de discrimination au sein de l'entreprise. Sans l'autorisation de la personne concernée, une société ne peut l'obliger à se soumettre à ce genre d'expérimentations, [CNIL- Le corp, nouvel objet connecté :
https://www.cnil.fr/sites/default/files/typo/document/CNIL_CAHIERS_IP2_WEB.pdf]

Les personnes qui commencent à réfléchir à la protection de leurs données issues de l'IoT peuvent évaluer leurs nouveaux appareils en répondant à ces quelques questions :

- L'appareil contient-il des données personnelles ?
- L'appareil contient-il des données générées par des machines ? Si oui, comment votre entreprise compte-t-elle utiliser ces données ?
- L'appareil est-il relié à un système de stockage de données interne ? Certains appareils connectés à un réseau stockent-ils les données en local plutôt que sur un serveur ?
- Quel est le contrat de niveau de service (SLA) actuel du département informatique en matière de restauration des données sur l'appareil ?

Sur le plan légal qui entoure les IoT, nous retrouvons un ensemble de textes applicables, mais qui ne vise pas forcément le domaine, ce qui revient à avoir de textes pour réprimer les atteintes et des textes pour imposer leur sécurisation :

- Réglementations spécifiques dans divers pays. Les approches peuvent être assez différentes (Europe - USA)
- Loi Godfrain du 5 janvier 1988 relative à la fraude informatique
- Loi de 1978 sur la protection des données personnelles
- Règlement « données personnelles » - RGPD (General Data Protection Regulation), applicable dès le 25 mai 2018
- Règlement « identification électronique et service de confiance »
- Projet de Directive « NIS » (Network and Information Security)

A l'avenir, il va falloir songer à appliquer aux objets connectés la technique du « Legal by design » pour être certain qu'il soit en conformité avec le droit, mais

quel est celui qui s'applique – fabricant – utilisateur – pays dans lequel il est utilisé ?

Les conséquences annexes d'un défaut de sécurisation des objets connectés, distribués, impliquent qu'une cybersécurité insuffisante diminue le droit à réparation du préjudice de la victime, qui dépend de l'appréciation de la faute de la victime par les magistrats et qui peut aboutir vers une quadruple peine pour l'entreprise en cas de piratage des objets connectés :

- Toutes les conséquences négatives « classiques » liées au piratage : pertes, efforts de remédiation, indisponibilité, effet d'image, procès, action de groupe, etc.
- En cas de données personnelles accédées, risque CNIL de constatation d'un défaut de sécurité
- Si un défaut de sécurité a permis le piratage, diminution des dommages et intérêts en conséquence
- Risque de sanctions diverses – pénales à terme avec 2% du CA mondial plus notation financière

Le BYOD (Bring Your Own Device) ou BYOIoT, consiste à utiliser des équipements personnels dans un contexte professionnel. Nombreux problèmes de sécurité de l'information, de protection des données sans oublier les aspects juridiques et sociaux (surveillance, heures de travail) qui peut se résumer par deux analyses - 1) solution ergonomique et économique, - 2) surcoûts non négligeables (formation, sécurisation, etc.).

D'autres approches existent :

- **CYOD** (*Choose Your Own Device*), choix par les employés dans un catalogue proposé par l'entreprise : l'appareil est propriété de l'entreprise.
- **COPE** (Corporate Owned, Personnally Enabled), matériel professionnel acheté par l'entreprise. L'usage personnel est plus développé que dans le CYOD.

Il est fondamental que les entreprises ayant eu un projet de déploiement BYOD ou une démarche similaire, auront un avantage considérable pour la prise en charge de l'IoT.

Pour finir ce chapitre, quelques normes ou projet de normes :

- ISO/IEC AWI (Approved new Work Item) 30141 Internet of Things Reference Architecture.
- Commission miroir à l'AFNOR
- ISO/IEC JTC 1 Big Data Preliminary Report 2015
- ISO TC 68 12812 Core Banking Mobile Financial Services, Norme en 5 parties
- Création en juin 2015 de la Commission d'études 20 de l'UIT-T sur l'Internet des objets et ses applications

7 Conclusion

L'Internet des objets peut être un concept puissant, cependant, comme tout nouveau déploiement technologique, le risque doit être évalué dans sa globalité d'usage, pour s'assurer que la valeur de l'entreprise est maximisée et que le risque est minimisé. L'IoT a un énorme potentiel qui a déjà modifier le comportement des personnes dans leurs vies quotidiennes, aussi bien professionnellement que personnellement. Les avantages sont nombreux mais l'IoT génère de nouveaux

risques et problèmes complexes. C'est un concept en rupture, dont l'usage nécessite une réflexion responsable et prospective afin de préserver l'entreprise.

IoT est un risque émergent pour l'entreprise, l'analyse de sécurité révèle qu'en réalité les dispositifs IoT sont des « consommables » est ne sont pas conçus pour être sûr; ils sont conçus pour être faciles à utiliser pour les consommateurs. L'Internet des objets a le potentiel pour élargir la surface d'attaque de l'entreprise et ces risques sont susceptibles de se produire en premier avec les télétravailleurs depuis leur domicile, puis avec l'arrivée massive des porteurs d'objets au sein de l'entreprise. Aujourd'hui les entreprises n'ont pas une bonne visibilité des risques de sécurité en associant « indirectement » le réseau de domicile et en ce multipliant, ces dispositifs deviendront un problème plus aigu à traiter.

IoT est juste une autre forme de « BYOD » et ne nécessite pas de dire non aux employés qui pose la question de pouvoir connecter – directement ou - indirectement leurs appareils sur le réseau d'entreprise. Nous ne pourrons pas arrêter la « grande marée » des dispositifs IoT et n'aurons pas d'autre choix que les gérer lorsqu'ils apparaîtront. Pour sortir de cette tendance en forme d'impasse, il faut déterminer dès à présent les moyens de contenir l'accès à ces dispositifs, de sorte que leurs impacts soient minimes quand ils rejoignent le réseau. Notre objectif devrait être de mettre en place des politiques de sécurité qui facilitent les utilisateurs à faire bon usage des objets et qui rendent plus difficile pour les utilisateurs de « détourner » l'usage attendu de ceux-ci.

Nos vies dépendent de l'infrastructure des soins de santé et l'infrastructure civile qui le rend possible pour nous de vivre et de travailler si étroitement ensemble dans les villes. Il est difficile d'imaginer comment la manipulation non autorisée des feux de circulation, les équipements médicaux, ou d'innombrables autres exemples qui peuvent causer des décès.

L'entreprise mettant en œuvre des objets connectés soulèvent de nouvelles problématiques de sécurité et ne rentrent pas forcément dans l'approche classique de sécurité de systèmes d'information, difficile de trouver sur l'étagère une trousse à outil permettant de couvrir l'ensemble des problématiques. Pour relever ces défis, le RSSI doit avoir un rôle actif dans l'élaboration et l'exécution de la stratégie d'entreprise, et la visibilité sur ces menaces est cruciale. De même elles ont besoin d'outils qui va les aider à identifier les menaces pour permettent une réponse préventive à celles-ci.

Le RSSI devra aussi développer une approche ad hoc, qui sera une combinaison d'outils et de méthodes existantes complétées par des apports techniques et méthodologiques trouvés en dehors de son domaine habituel de compétence.

Références et bibliographie

1. Proposition de directive « NIS » : http://www.europarl.europa.eu/registre/docs_autres_institutions/commission_europeenne /com/2013/ 0048/COM_COM%282013%290048_FR.pdf
2. VeraCode study : Internet of Things Devices lack fundamental security : http://www.zdnet.com/article/internet-of-things-devices-lack-fu/
3. Bruce Schneier au sujet de la sécurité de l'IoT : « ça va être pire et s'effondrer » : http://www.infoworld.com/article/2908939/security/schneier-on-really-bad-iot-security-it-s-going-to-come-crashingdown.html
4. Say Goodbye to Privacy : http://www.wired.com/2015/02/say-goodbye-to-privacy/
5. We Asked Executives About The Internet Of Things And Their Answers Reveal That Security Remains A Huge Concern http://www.businessinsider.in/We-Asked-Executives-About-The-Internet-Of-Things-And-Their-Answers-Reveal-That-Security-Remains-A-Huge-Concern/articleshow/45959921.cms

6. These Devices May Be Spying On You (Even In Your Own Home) : http://www.forbes.com/sites/josephsteinberg/2014/01/27/these-devices-may-be-spying-on-you-even-in-your-own-home/
7. Proof-of-Concept CarShark Software Hacks Car Computers, Shutting Down Brakes, Engines, and More :
8. http://www.popsci.com/cars/article/2010-05/researchers-hack-car-computers-shutting-down-brakes-engine-and-more
9. CIA Chief: We'll Spy on You Through Your Dishwasher : http://www.wired.com/2012/03/petraeus-tv-remote/
10. Rapport de la Federal Trade Commission (FTC), janvier 2015, « Privacy & Security in a Connected World » : https://www.ftc.gov/system/files/documents/reports/federal-trade-commission-staff-report-november-2013-workshopentitled-internet-things-privacy/150127iotrpt.pdf
11. NIST official: Internet of Things is indefensible : http://fcw.com/articles/2015/04/16/iot-is-indefensible.aspx

Test d'intrusion dans un système de contrôle de la qualité de l'eau

Adam Reziouk, Aurélien Thierry, Jonathan-Christofer Demay

prenom.nom@airbus.com
Airbus Defence & Space - CyberSecurity

Résumé La sécurisation d'un réseau d'objets connectés est un enjeu majeur à la fois pour les équipements destinés au grand public et ceux destinés à être utilisés au sein d'infrastructures critiques. Nous avons récemment travaillé sur la sécurisation d'un réseau de capteurs mesurant la qualité de l'eau. Ces capteurs forment un réseau sans fil répondant à la norme IEEE 802.15.4 interconnecté à un poste de contrôle à l'aide du protocole 6LoWPAN.
Nous expliquons le fonctionnement de ces normes en détaillant les aspects liés à leur sécurisation et les attaques possibles en pratique. Nous déroulons ensuite l'attaque que nous avons effectuée sur le système de contrôle de la qualité de l'eau : les outils que nous avons développés et mis à disposition au sein du projet Scapy-radio [1] nous ont permis de connaître les détails du réseau et de prendre la place d'un capteur légitime afin de modifier les données remontées au poste de contrôle**.**

1 Introduction

Les infrastructures de surveillance du système de distribution d'eau remontent les **données de nombreux capteurs qui sont alimentés à l'aide de petites turbines** présentes à l'intérieur même des conduits d'eau. Nous avons audité en particulier un système de deux capteurs, en photo sur la figure 1.

Fig. 1. Capteurs intégrés au système de contrôle de l'eau

Les contraintes de fonctionnement de ce type d'objets connectés (taille, puissance de calcul...) ont amené à la création de nouveaux protocoles et modes d'association entre plusieurs objets formant un réseau.

1.1 Modèle OSI

Le protocole 802.15.4 occupe les couches 1 et 2 du modèle OSI et ne gère donc pas l'adressage habituellement laissé au protocole IP (IPv4 ou IPv6) en couche 3. Afin de limiter la taille des trames associées à cette couche, le protocole 6LoWPAN est mis en avant : il permet d'omettre une partie des informations présentes dans IPv6 lorsque celles-ci peuvent être déduites ou négociées à l'avance.

Les couches supérieures sont identiques à celles utilisées dans un réseau classique : TCP ou UDP, puis les protocoles spécifiques à chaque couche applicative. La figure 2 donne les couches dans le modèles OSI liées à un réseau filaire classique (Ethernet) et celles dans le cas d'un réseau d'objets connectés.

Couche	Elément échangé	Réseau classique	Réseau de type IoT
7	Donnée	...	...
6	Donnée	...	...
5	Donnée	...	...
4	Segment	TCP, UDP	TCP, UDP
3	Paquet	IPv4, IPv6	6LoWPAN
2	Trame	Ethernet	802.15.4 MAC
1	Bit	Ethernet	802.15.4 PHY

Fig. 2. Couches dans le modèle OSI pour un réseau filaire classique et un réseau de type IoT

Relier un réseau d'objets connectés avec un réseau plus standard n'est pas évident parce que les couches d'adressage (IPv6 ou 6LoWPAN) sont différentes. Un équipement spécifique, appelé border router, peut alors être ajouté au réseau : il fait partie à la fois d'un réseau IPv6 et du réseau 6LoWPAN et est chargé de transmettre les paquets d'un réseau vers l'autre en les adaptant.

1.1 Organisation du document

A l'instar d'un réseau Wi-Fi sécurisé (WEP, WPA...), un réseau 802.15.4 ne devrait être accessible qu'aux équipements autorisés et disposant des informations de chiffrement adéquats, selon la configuration du réseau. Nous détaillerons ces mécanismes de sécurité ainsi que les méthodes de chiffrement permettant d'assurer la confidentialité et l'intégrité des données transitant dans ce type de réseaux.

Au-delà du réseau sous-jacent, nous souhaitons auditer les applications présentes sur les appareils connectés au réseau 6LoWPAN. Nous verrons comment une fois le réseau rejoint, on peut mettre en un place un border router afin de communiquer de manière transparente avec les équipements du réseau 6LoWPAN. Nous détaillerons alors les actions et attaques effectuées lors de l'audit du système de contrôle de l'eau.

2 Réseau 802.15.4

Dans cette partie nous donnons quelques éléments sur le fonctionnement d'un réseau 802.15.4 et en particulier sur les mécanismes de sécurité implémentés dans le standard. Davantage de détails peuvent être trouvés dans la littérature [2,3,4,6,7].
Dans l'objectif d'auditer ce type de réseaux nous avons réalisé un scanner capable de déterminer les caractéristiques du réseau, de lister les équipements présents et leurs modes de communication.

2.1 Fonctionnement du réseau 802.15.4

La norme 802.15.4 permet de définir de manière fine des réseaux variés dans leur topologie (pair à pair ou centralisé) et dans leur mode de fonctionnement pour le transfert de données. Nous donnons le vocabulaire utilisé dans la norme et présentons les notions principales qui seront utilisées dans la suite de l'étude.

Composants du réseau On sépare les éléments d'un réseau PAN (*Personal Area Network*) en équipements à fonctionnalité réduite (RFD ou *Reduced-Function Device*) et en équipements à pleine fonctionnalité (FFD ou *Full-Function Device*).

Un RFD ne peut communiquer qu'avec un FFD tandis que les FFD peuvent avoir des rôles d'intermédiaires dans le réseau et communiquer entre eux. Les RFD sont particulièrement utiles pour implémenter, avec un impact énergétique faible, la partie réseau d'un capteur basse consommation.

Un réseau 802.15.4 doit avoir exactement un FFD qui prend le rôle de coordinateur de réseau (*PAN Coordinator*) et peut disposer d'autres FFD et RFD.

Tous les équipements du PAN partagent un identifiant de réseau (*PANId*) qui doit être différent des identifiants des réseaux voisins. La méthode de choix de cet identifiant n'est pas spécifiée dans la norme 802.15.4, elle doit donc être déterminée par les couches protocolaires supérieures.

Topologie du réseau Deux topologies de réseau sont possibles. Dans un réseau en étoile (figure 3(a)) le coordinateur de réseau est placé au centre et les autres équipements communiquent uniquement avec lui.

Dans un réseau pair-à-pair (figure 3(b)) les équipements de type FFD peuvent communiquer directement entre eux et le coordinateur de réseau n'a pas un emplacement privilégié. Ce mode de fonctionnement est plus complexe mais il permet d'avoir une architecture dynamique.

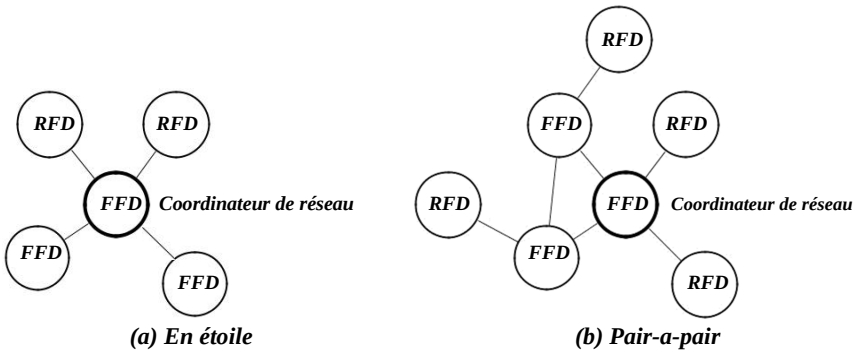


Fig. 3. Topologies d'un réseau 802.15.4

Transfert de données

Format des trames MAC. Les trames échangées au sein de la couche 2 d'un réseau IoT (802.15.4 MAC) contiennent un entête, des données et un code de détection d'erreur (CRC). L'entête contient :

- le type de trame (envoi de balise, échange de données, acquittement, commande),
- un numéro de séquence (compteur de trames),
- des informations sur le niveau de sécurité à appliquer (voir section 2.2),
- les adresses de l'émetteur et du destinataire,
- l'identifiant du réseau PAN (*PANId*) de l'émetteur et du destinataire.

Commandes MAC. Lorsque la trame est de type commande, il peut s'agir d'un des ordres suivants :

- demande d'association,
- réponse à l'association,
- notification de désassociation,
- demande de données,
- notification de conflit sur le *PANId*,
- notification d'équipement orphelin,
- demande de balise,
- resynchronisation du coordinateur de réseau,
- demande GTS (*Garanteed Time Slot*).

La demande GTS permet à un équipement de disposer d'une fenêtre de communication exclusive pendant une période de temps restreinte.

Une commande de resynchronisation est envoyée dès qu'un conflit sur le *PANId* est détecté. Le coordinateur de réseau indique alors un nouveau *PANId* et éventuellement un nouveau canal.

Utilisation de balises. Le coordinateur de réseau peut envoyer régulièrement des balises (beacons) pour donner des informations aux équipements. L'échange de données peut alors se faire sur demande du coordinateur : il envoie une balise pour indiquer la présence de données à récupérer puis l'équipement visé envoie une demande pour ces données et enfin le coordinateur envoie les données.

Un mode sans balises est possible : les équipements doivent alors régulièrement envoyer des demandes de données au coordinateur pour ne pas rater un message (modèle *pull*).

2.2 Sécurité

La norme 802.15.4 a été pensée pour pouvoir assurer la confidentialité et l'intégrité des messages échangés avec un mode de sécurité (par exemple avec un mécanisme de *whitelist*) et une politique cryptographique de sécurité. Ces mécanismes de sécurité ont évolué lors des versions successives de la norme (2003, 2006, 2011).

Modes de sécurité La norme 802.15.4 spécifie comment les couches supérieures peuvent choisir les équipements avec lesquels l'appareil peut communiquer.

Version 2003. La première version de la norme propose un mode non sécurisé (sans restriction), un mode par ACL (Access Control List) et un mode sécurisé.

En mode ACL une liste d'adresses d'équipements autorisés est à disposition de la couche MAC. Un équipement non autorisé verra ses trames réseau rejetées.

Le mode sécurisé associe chaque équipement autorisé au niveau de sécurité à utiliser (pour la communication avec ces équipements), au matériel cryptographique (clé pour le chiffrement, l'intégrité ou les deux) ainsi qu'aux compteurs de trames entrantes, de trames sortantes et de séquence de clé.

Version 2006. La version 2006 ne propose plus que deux modes : sécurisé et non sécurisé. De plus le niveau de sécurité associé à un équipement est maintenant un minimum : un niveau de sécurité supérieur est accepté.

Version 2011. La version 2011 de la norme permet de définir un ensemble de niveaux de sécurité acceptables pour un équipement donné au lieu d'un niveau minimal comme précédemment.

Chiffrement

Version 2003. La version 2003 de la norme définit 8 politiques de sécurité données à la figure 4. Chaque mode spécifie un algorithme et la taille du code d'intégrité (MIC). Nous indiquons quels modes protègent l'intégrité ou la confidentialité des données.

Le chiffrement s'applique uniquement aux données (*payload*) de la trame MAC tandis que la vérification d'intégrité s'applique aux entêtes et aux données. Le mode AES-CCM s'applique à chiffrer les données et le code d'intégrité.

Les modes en chiffrement (AES-CTR, AES-CCM) nécessitent un nonce cryptographique qui est composé de l'adresse de la source, du compteur de trame et du compteur de séquence de clé.

Niveau de sécurité	Confidentialité	Intégrité
0 : Aucune	Non	Non
1 : AES-CTR	Oui	Non
2 : AES-CCM-128	Oui	Oui
3 : AES-CCM-64	Oui	Oui
4 : AES-CCM-32	Oui	Oui
5 : AES-CBC-MAC-128	Non	Oui
6 : AES-CBC-MAC-64	Non	Oui
7 : AES-CBC-MAC-32	Non	Oui

Fig. 4. Politiques de sécurité disponibles (version 2003)

Versions 2006 et 2011. Depuis la version 2006 un entête (*Auxiliary Security Header*) indique le mode cryptographique utilisé par la trame ainsi qu'une méthode d'identification de la clé à utiliser. Les politiques de sécurité sont données en figure 5.

Niveau de sécurité	Confidentialité	Intégrité
0 : Aucun	Non	Non
1 : MIC-32	Non	Oui
2 : MIC-64	Non	Oui
3 : MIC-128	Non	Oui
4 : ENC	Oui	Non
5 : ENC-MIC-32	Oui	Oui
6 : ENC-MIC-64	Oui	Oui
7 : ENC-MIC-128	Oui	Oui

Fig. 5. Politiques de sécurité disponibles (à partir de la version 2006)

Désormais seul le mode AES-CCM est utilisé, dérivé du mode AES-CCM dans le but de pouvoir réaliser uniquement le chiffrement ou la vérification d'intégrité. Il est ainsi possible d'assurer le chiffrement (ENC), la vérification d'intégrité (MIC pour *Message Integrity Code*), ou les deux.

Le nonce requis par le mode CCM est cette fois composé de l'adresse de la source, du compteur de trame et du niveau de sécurité.

Mode AES-CCM. L'algorithme de chiffrement AES (symétrique, par blocs) est un choix naturel pour ce genre d'applications parce qu'il est standard et a été sélectionné pour être implémenté très exactement de manière matérielle. Le mode CCM combine le mode CBC pour la vérification d'intégrité et les caractéristiques de chiffrement du mode CTR. Le mode CTR présente l'avantage de chiffrer octet par octet (comme lors d'un chiffrement par flot) à l'inverse d'un mode par blocs habituel qui doit ajouter du *padding* si les messages sont petits et augmenterait donc inutilement la taille des trames échangées.

Le désavantage du mode CCM est qu'il nécessite l'utilisation d'un nonce différent à chaque message échangé.

Unicité du nonce. Ainsi le couple (clé, nonce) doit être unique pour chaque trame protégée sinon des attaques cryptographiques sont possibles. Toutes les versions de la norme spécifient que le matériel cryptographique (clés) doit être renouvelé lorsque le nonce ne peut pas être unique, c'est à dire lorsque les compteurs de trame sont remis à zéro.

Ce renouvellement n'est pas spécifié dans la norme 802.15.4 et doit être implémenté par les couches protocolaires supérieures.

Sûreté de fonctionnement On se trouve dans un contexte industriel avec des équipements censés fonctionner en permanence. Une solution à un bug logiciel ou matériel est souvent de réinitialiser l'équipement dans un état d'origine que l'on sait fonctionnel. Ainsi plusieurs problèmes (perte de la connexion au coordinateur, difficulté à recevoir des messages pertinents, etc.) peuvent avoir pour conséquence le redémarrage de l'appareil.

Ce *reboot*, très pertinent pour retrouver la disponibilité de l'équipement, pose un risque pour la sécurité des primitives cryptographiques utilisées ainsi que pour les mécanismes anti-rejeu. En effet si les compteurs de trames sont stockés dans la mémoire volatile ou dans une mémoire non volatile mais remis à zéro lors d'un redémarrage, il devient possible de rejouer des trames existantes et d'attaquer la cryptographie sous-jacente (le nonce n'est plus unique).

2.3 Vulnérabilités

Nous présentons ici un aperçu des attaques connues contre un réseau 802.15.4 et nous évaluons leurs impacts potentiels dans un contexte opérationnel. Nous ne prendrons pas en compte les attaques qui nécessiteraient au préalable une compromission physique d'un équipement du réseau ni les attaques uniquement basées sur le brouillage des ondes radio (attaques sur la couche physique) qui sont inhérentes à toute technologie sans fil.

Déni de service De nombreuses attaques par déni de service sont possibles sur tout réseau sans fil (envoi de trames non sollicitées pour épuiser la batterie par exemple) et des attaques de ce type spécifiques à la norme 802.15.4 sont possibles en augmentant les compteurs de trame par exemple.

Ces attaques ne nous intéressent pas dans cette étude car elles ne permettent pas de rejoindre le réseau ni d'accéder à des informations confidentielles.

Faiblesse du chiffrement Une attaque cryptographique est possible lorsqu'il y a réutilisation du nonce (modes AES-CTR, AES-CCM, ENC et ENC-MIC). En effet, pour être chiffrée, la donnée en clair subit une opération simple : un XOR (⊕) avec un flot de chiffrement (*keystream*) dépendant uniquement de la clé et du nonce.

Dans le cas ou deux messages P1 et P2 sont chiffrés avec la même clé et le même nonce, le *keystream* K sera alors identique pour les deux messages. Notons maintenant C1 et C2 les chiffres correspondant à P1 et P2 respectivement. Par définition, on a $C1 = P \oplus K$ et $C2 = P2 \oplus K$. Dans ce cas précis, on obtient donc $C1 \oplus C2 = (P1 \oplus K) \oplus (P2 \oplus K) = P1 \oplus P2$.

L'égalité $P1 \oplus P2 = C1 \oplus C2$ relie fortement les versions chiffrées des versions en clair. Par exemple, si un des deux textes clairs est connu, le second peut être déduit. Dans le cas général, on peut extraire des informations statistiques intéressantes même sans connaître de texte en clair.

Le compteur de trames sortantes est le principal responsable de la variation du nonce. L'attaque est donc possible dès que celui-ci est remis à zéro, ce qui peut arriver au redémarrage.

Rejeu de trames sécurisées Les attaques par rejeu de trames sécurisées ne devraient pas être possibles car les compteurs de trames entrantes sont conservés par l'équipement récepteur de telle sorte qu'il rejette des trames déjà traitées.

Un cas typique de mauvaise gestion des compteurs de trames entrantes est, à nouveau, leur remise à zéro lorsque l'équipement redémarre. Dans ce cas un attaquant peut réaliser une attaque en rejouant des trames chiffrées déjà traitées.

Attaque sur la malléabilité La malléabilité est une propriété cryptographique de l'algorithme utilisé. Si l'on connaît un texte clair P et son équivalent chiffré C, ils sont reliés à un *keystream* K par la relation $C = P \oplus K$ et on peut alors récupérer la valeur du *keystream* K.

Ce *keystream* correspond à un nonce particulier donc a une valeur spécifique du compteur de trames sortantes de l'émetteur. Si le compteur reprend cette valeur le *keystream* sera identique (si la clé n'est pas modifiée). On peut donc forger un message chiffré valide pour cette valeur spécifique du compteur de trames sortantes. Ce message sera accepté par l'équipement récepteur uniquement si son compteur de trames entrantes a été réinitialisé, c'est à dire dans le même cas que pour l'attaque par rejeu.

Cette technique permet uniquement de construire un message chiffré valable mais pas son code d'intégrité (MIC) car celui-ci lui dépend directement de la clé (et non du *keystream*). Les modes de chiffrement avec vérification d'intégrité ne sont donc pas vulnérables.

L'attaque est donc possible dans le cas où les compteurs de trame ne sont pas gérés correctement et s'il n'y a pas de vérification d'intégrité. Elle permet de générer des trames sécurisées valides si l'on connaît un message clair précédent, observé pour la même valeur du compteur de trames sortantes.

3 6LoWPAN et IPv6

Une fois que l'on a récolté suffisamment d'informations sur le réseau cible à l'aide du scanner, on peut passer à l'étape active afin de communiquer avec les équipements du réseau.
Nous expliquerons comment ajouter un *border router* et comment celui-ci peut servir d'interface entre la machine de l'auditeur sur une interface IPv6 d'une part et le réseau 6LoWPAN d'autre part.

3.1 6LoWPAN sur 802.15.4

6LoWPAN est basé sur IPv6 et permet, dans notre cas, l'adressage dans un réseau composé d'équipements respectant la norme 802.15.4. La principale difficulté lors de l'adaptation d'IPv6 à 802.15.4 est la taille limitée des trames 802.15.4 : elle est de 127 bits au maximum alors que la norme IPv6 prévoit au moins 1280 bits. Les entêtes IPv6 vont en plus occuper la majeure partie des 127 bits attribués par la couche inférieure.

La solution apportée par 6LoWPAN consiste à fragmenter les packets IPv6 dans plusieurs trames 802.15.4 puis à les rassembler sur l'équipement de destination. Les entêtes IPv6 sont également réduits par un schéma de compression spécifique.

Entêtes IPv6 et UDP. Les entêtes IPv6 sont compressés sans ménagement comme décrit dans la RFC 6282. De nombreux champs peuvent être réduits. Par exemple :

- il est possible d'utiliser les adresses courtes du 802.15.4,
- la taille de la charge utile peut être déduite des couches inférieures,
- le nombre maximum de sauts intermédiaires dans une connexion (de 0 à 255) peut être réduit à trois valeurs (1, 64, 255).

De même certains champs de l'entête UDP peuvent être omis. Par exemple la somme de contrôle (*checksum*) n'est pas nécessaire si un autre mécanisme de vérification de l'intégrité est présent (par exemple IPSec).

Fragmentation des paquets. Les paquets IPv6 qui ne peuvent pas tenir dans une seule trame 802.15.4 sont découpés selon la RFC 4944. Un paquet IPv6 est alors découpé en un premier fragment et un ou plusieurs fragments successifs.

Ces paquets sont composés, en plus des données, d'un tag (5 bits) qui indique s'ils sont le premier fragment ou un fragment successif, de la taille des données du fragment (11 bits), d'un tag unique à chaque paquet IPv6 (avant fragmentation) et, pour les fragments successifs, de leur emplacement dans le paquet IPv6 d'origine.

3.2 Border router

Pour des raisons de temps de développement nous avons réalisé un *border router* purement logiciel en utilisant les fonctionnalités de réseau virtuel du noyau Linux (TUN pour *tunneling*).

Comme indiqué dans la figure 6 la conversion entre IPv6 et 6LoWPAN se fait au niveau de la couche 3. Les outils utilisés par l'auditeur (*nmap*, *ping*...) pourront ainsi utiliser la sortie Ethernet comme celle connectée au réseau 802.15.4 de manière transparente, selon l'adresse IP cible.

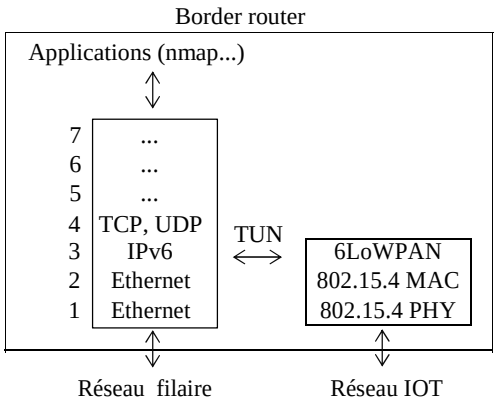


Fig. 6. Border router et liens avec les réseaux Ethernet et IoT

Place dans le réseau. Le *border router* doit pouvoir communiquer avec le maximum d'équipements du réseau. Il est conseillé d'utiliser le scanner décrit précédemment pour déterminer la configuration du réseau afin que l'équipement ajouté la respecte.

Une première possibilité consiste à s'intégrer au réseau en tant qu'équipement supplémentaire. Certains équipements refuseront de communiquer avec un appareil non associé avec le coordinateur de réseau tandis que des réseaux plus sécurisés refuseront l'association parce qu'ils ont une liste d'équipements autorisés.

Il est également possible de se faire passer pour un équipement existant en fournissant l'adresse IPv6 et une adresse 802.15.4 de l'équipement cible.

Selon les modes de sécurité en vigueur, l'association et la communication peuvent être insuffisants si l'on ne dispose pas des clés de chiffrement utilisées pour communiquer avec les autres équipements.

4 Attaque du réseau de capteurs

Les éléments du réseau de capteurs utilisent la norme IEEE 802.15.4 sous la forme d'un réseau centralisé (en étoile) ainsi que les protocoles 6LoWPAN et UDP pour le transport des données. La topologie en étoile est classique pour remonter des données : chaque capteur est un RFD qui n'a besoin de communiquer qu'avec le coordinateur de réseau.

Le but du projet est de permettre aux auditeurs d'utiliser leurs propres ordinateurs avec leurs outils de prédilection sur un réseau 6LoWPAN quelle que soit la configuration du réseau 802.15.4 sous-jacent. Nous allons donc nous concentrer sur les aspects spécifiques liés à l'utilisation de ce type de réseau.

4.1 Découverte du réseau 802.15.4

Scan initial. Nous nous sommes basés sur l'écoute du coordinateur de réseau et des deux équipements qui étaient accessibles à l'aide du scanner, la sortie du scanner est donnée en figure 7. On distingue bien le *PANId* (0xabba) partagé par les trois équipements, que les équipements 1 et 2 communiquent uniquement avec le coordinateur (*Transmitter0*) et les adresses courtes de chacun (0xde00, 0xde01, 0xde02).

```
Transmitter0:
beacon_enabled=0x1
pan_coord=0x1
coord=0x1
gts=0x0
panid=0xabba
```

```
Transmitter1:
short_addr=0xde02
panid=0xabba
Destination0:
   security_enabled=0x1
   frame_version=0x1L
   short_addr=0xde00
   coord=0x1
   command=0x0
   panid=0xabba
   data=0x5
   pan_coord=0x1
```

```
Transmitter2:
short_addr=0xde01
panid=0xabba
Destination0:
   security_enabled=0x1
   frame_version=0x1L
   short_addr=0xde00
   coord=0x1
   command=0x0
   panid=0xabba
   data=0x4
   pan_coord=0x1
```

Fig. 7. Résultat du scan initial du réseau 802.15.4

Nous avons ainsi déterminé que :

- La norme 802.15.4 est bien utilisée, sur le canal 18.
- Chaque équipement communique uniquement avec le coordinateur de réseau, c'est une caractéristique attendue dans un réseau en étoile.
- Le réseau utilise le mode de transfert avec balises.
- Les identifiants présents dans les trames indiquent que c'est la version 2006 de la norme qui est utilisée.
- Le mode sécurisé de la norme est utilisé avec chiffrement (ENC) et vérification d'intégrité (MIC).
- Le coordinateur de réseau n'alloue pas de période d'accès exclusive (GTS). Nous n'avons pas pu observer les adresses longues des équipements, seulement les adresses courtes utilisées après la phase de synchronisation avec le coordinateur.

Resynchronisation. Nous souhaitons connaître les adresses longues des équipements car elles sont utilisées dans les primitives de chiffrement et nous permettront de réaliser les attaques cryptographiques.

Par essais successifs nous avons remarqué que le coordinateur envoie régulièrement des balises et que si les capteurs reçoivent de nombreuses trames parasites ils n'arrivent pas à recevoir les balises légitimes à temps et perdent la synchronisation avec le coordinateur. Nous capturons alors les trames de resynchronisation échangées avec le coordinateur ; celles-ci contiennent bien les adresses longues souhaitées.

Association. Comme on sait maintenant forcer la resynchronisation des capteurs, on s'attarde sur la procédure d'association et les trames échangées avec le coordinateur. On a pu déterminer que :

- Les trames d'association ne sont pas sécurisées (niveau de sécurité a 0) ; les trames suivantes (après association) sont, elles, sécurisées.
- Le coordinateur de réseau n'applique aucune restriction (authentification) pour rejoindre le réseau.

- N'importe quelle adresse étendue (sur 64 bits) est acceptée par le coordinateur.
- Les adresses courtes sont systématiquement attribuées dans le même ordre (0xde01, 0xde02...).

Nous nous sommes alors intéressés aux canaux utilisés après désynchronisation :
- Les équipements cherchent le premier coordinateur de réseau sur les canaux 11 à 26 (dans cet ordre) à l'aide de balises.
- Si, arrivés au canal 26, ils ne trouvent pas de coordinateur, ils redémarrent.

Ainsi si l'on arrive à empêcher la synchronisation sur le canal 18, on peut forcer les équipements à redémarrer.

De plus on peut se faire passer pour le coordinateur de réseau en se plaçant sur un canal inférieur à 18 et en se faisant passer pour le coordinateur légitime (mêmes adresses, même *PANId*). L'association se fera sans problème parce que les trames ne sont pas sécurisées à ce stade.

Nous avons enfin remarqué que le coordinateur de réseau légitime cesse d'envoyer des balises pendant un certain temps s'il ne reçoit plus de trames de données des capteurs. Il semblerait logique que, à l'instar des capteurs, celui-ci redémarre dans l'espoir de relancer une communication normale.

Nous sommes donc à présent capables de redémarrer n'importe lequel des appareils du réseau, capteurs et coordinateur.

Persistance dans le réseau. Nous avons vu qu'on est capables de :
- redémarrer les équipements et le coordinateur,
- créer un faux coordinateur et forcer un (ou des) équipement(s) à s'y connecter après redémarrage,
- créer un faux équipement et le connecter au coordinateur légitime.

Afin d'envoyer des trames à un équipement ou au coordinateur on peut donc soit créer un nouvel appareil soit prendre la place d'un équipement ou coordinateur légitime. Nous avons privilégié cette seconde option qui nous permet de mener l'attaque par rejeu.

4.2 Attaque cryptographique

Nous avons vu comment rejoindre le réseau en prenant la place d'équipements existants ou en créant un nouvel équipement et comment contrôler le redémarrage des équipements légitimes et du coordinateur.

Nous allons à présent étudier la possibilité de communiquer avec le réseau en contournant les mesures de sécurité mises en place par la norme.

Compteurs de trames sortantes. Un compteur de trames sortantes est gardé en mémoire pour chaque autre équipement connu ; ce compteur est incrémenté à chaque envoi de trame et est transmis dans la trame. De plus il est utilisé pour créer le nonce requis par le mode sécurisé.

Quand le compteur de trames sortantes atteint la valeur maximale (2^{32}) et doit être remis à zéro, le matériel de chiffrement doit être changé pour éviter une attaque cryptographique par réutilisation du nonce.

Nous avons observé les compteurs de trames avant et après redémarrage car celui-ci n'est pas chiffré même dans les trames sécurisées et conclu que ces compteurs sont également remis à zéro lors du redémarrage, nous permettant alors de mener l'attaque par réutilisation du nonce.

Compteurs de trames entrantes. Les compteurs de trames entrantes sont un élément essentiel de la sécurité des équipements 802.15.4 afin d'empêcher des attaques par rejeu.

Chaque équipement garde un compteur de trames entrantes pour chaque autre équipement connu et rejette les trames dont le compteur est inférieur à celui gardé en mémoire. Nous rappelons qu'avec le mode de sécurité active (vérification d'intégrité) nous ne pouvons pas modifier les compteurs mais simplement rejouer des trames enregistrées.

Nous avons déterminé que le compteur de trames entrantes est remis à zéro lorsque le coordinateur redémarre, nous autorisant donc à effectuer des attaques par rejeu de trames connues.

Il s'agit d'un enjeu important pour ce type de réseaux car un attaquant pourrait amener les opérateurs à observer des mesures inchangées (rejouées) alors que les capteurs détectent bien des événements anormaux sans pouvoir les transmettre.

Abaissement du niveau de sécurité. Le mode de sécurité appliqué dans ce réseau inclut la vérification de l'intégrité des messages empêchant donc de forger de nouveaux messages. Avec la norme en version 2006 plusieurs niveaux de sécurité sont acceptables : si le niveau minimum requis ne nécessite pas de contrôle d'intégrité il est possible de spécifier l'utilisation de ce niveau de sécurité inférieur dans les trames (*downgrade attack*). Ainsi l'intégrité n'est plus vérifiée et nous pouvons utiliser l'attaque par malléabilité.

Injection de trames sécurisées. Il est maintenant possible d'aller plus loin que le rejeu. Nous avons en effet toutes les cartes en main pour effectuer l'attaque par malléabilité (voir section 2.3) et forger de nouvelles trames sécurisées valides.

La première étape consiste à mener l'attaque par réutilisation du nonce afin d'obtenir suffisamment de *keystreams*. Il s'agit d'une attaque possible si l'on dispose de suffisamment de données chiffrées mais nous ne l'avons pas mise en place par manque de temps.

Nous sommes plutôt partis sur une approche "grey box" dans laquelle le client nous a fourni de nombreux textes clairs. Ces textes clairs nous ont permis de déterminer des *keystreams* correspondant à des valeurs de compteurs que nous pouvons réutiliser en redémarrant les équipements.

Nous utilisons donc ces *keystreams* pour chiffrer correctement les messages et forger des trames sécurisées valides dans lesquelles nous pouvons intégrer des données des couches supérieures et ainsi poursuivre l'audit avec des outils standards.

5 Conclusion

Nous avons présenté les aspects liés à la sécurité des réseaux 802.15.4 et 6LoWPAN et montré comment un auditeur peut scanner ce type de réseaux à la recherche de défauts de conception ou de configuration des équipements. Les contraintes de ces équipements en termes de disponibilité incitent les constructeurs à implémenter des mécanismes de redémarrage en cas de problème afin d'assurer une continuité de service.

Une source importante de vulnérabilités réside dans la mauvaise gestion de compteurs qui ne devraient jamais être remis à zéro (même en cas de redémarrage) et des clés de chiffrement qui devraient être changées dans ce cas. Une configuration erronée des niveaux de sécurité en aggrave les conséquences : si la vérification d'intégrité n'est pas requise, il devient possible de forger des nouvelles trames chiffrées valides.

Nous avons démontré comment effectuer cette attaque sur un réseau de capteurs qui contrôle la qualité de l'eau. Nous avons pu rejouer des trames existantes et en former de nouvelles afin d'envoyer des messages erronés au poste de contrôle.

Références

1. Scapy-radio. https://bitbucket.org/cybertools/scapy-radio.
2. Roberta Daidone, Gianluca Dini, and Marco Tiloca. On experimentally evaluating the impact of security on IEEE 802.15.4 networks. In DCOSS, pages 1{6. IEEE Computer Society, 2011.
3. Vojislav B. Misic, Jun Fung, and Jelena Misic. MAC Layer Attacks in 802.15.4 Sensor Networks. Security in Sensor Networks, 2006.
4. Adam Reziouk, Enzo Laurent, and Jonathan-Christofer Demay. Practical security overview of IEEE 802.15.4. ICEMIS, 2016.
5. Adam Reziouk, Arnaud Lebrun, and Jonathan-Chrostofer Demay. Auditing 6LoW-PAN Networks using Standard Penetration Testing Tools. DEFCON, 2016.
6. Adam Reziouk, Aurelien Thierry, and Jonathan-Christofer Demay. Réseau sans fil 802.15.4 et sécurité. MISC, (86), 2016.
7. Naveen Sastry and David Wagner. Security considerations for IEEE 802.15.4 networks. In Proceedings of the 3rd ACM workshop on Wireless security, WiSe '04, pages 32{42. ACM, 2004.

Véhicules Connectés et Cybersécurité

Fulup Le Foll, fulup@iot.bzh
Stéphane Desneux, sdx@iot.bzh

IOT.BZH

Abstract. The hundreds of millions of cars on our roads every day is a unique source of highly valuable data. While technically, connected car already makes those data available outside the vehicle, this export raises serious questions on how to secure the process. Connected cars raise premium challenges for the integrity of vehicle global security. It also raises issues about who collects and owns those data or how the process may respect users privacy. This talk exposes the ongoing work inside AGL (Automotive Grade Linux) to secure the global architecture from data collection to their export on the cloud. How to run untrusted applications without compromising your security, how to push only selected data from the car to the cloud, how to separate real time operations from less critical ones as entertainment, this without exploding development cost or limiting developers creativity.

1 Contexte Économique

Aujourd'hui, tous les analystes s'accordent à dire que l'Internet des Objets est déjà passé devant le « Big Data[1]» en termes de potentiel de marché. On dénombrait 20 milliards d'objets connectés fin 2015 ; ils devraient être près de 50 milliards dès 2020. Coté automobile, on prévoit 200 à 250 millions de véhicules connectés pour 2020. Les analystes prédisent un quadruplement du marché dans les cinq prochaines années. Ceci représente un ajout en valeur aux marchés automobiles existants de 150 milliards de dollars. En 2018, le marché des services connectés aux véhicules devrait déjà atteindre 40 milliards d'euros, et va continuer à croître jusqu'à la généralisation des véhicules autonomes vers 2035/2040.

Les changements en cours représentent pour le marché automobile une révolution aussi importante que l'arrivée d'internet dans le secteur bancaire, ou la mise en place de la gratuité de la voix chez les opérateurs téléphoniques. Dans cette révolution à venir, il y aura des gagnants et des perdants, comme dans toutes les révolutions : il est donc critique pour les grands constructeurs de ne pas rater la marche. Pour ce faire, ils se doivent d'acquérir des technologies et des compétences qu'ils n'ont pas, et notamment celles issues de la téléphonie mobile pour la partie acquisition et du Cloud et du « Big Data » pour les traitements.

Toutefois, il est important de rappeler qu'une voiture n'est pas une TV ou un téléphone. Il est donc nécessaire de reformater les technologies existantes afin de les rendre compatibles avec les contraintes de l'automobile. Tout le monde comprend qu'une voiture doit être mieux sécurisée qu'une télévision, ou que la durée de vie moyenne d'un véhicule est de 20 ans alors que celle d'un téléphone mobile dépasse rarement 3 ans.

Le challenge de l'automobile, comme celui de tous les autres marchés de l'internet des objets, est triple :

- réussir à simplifier la mise en œuvre d'un modèle de cybersécurité applicatif de bout en bout

[1] https://fr.wikipedia.org/wiki/Big_data

- fournir un ensemble d'outils qui rendent acceptable le coût et la complexité attachés au développement et à la mise en œuvre d'un projet en environnement hautement sécurisé
- assurer que le modèle de sécurité couvre les applications sur l'ensemble de leurs cycles de vie

2 La situation actuelle

Jusqu'à présent, les véhicules comme la grande majorité des autres infrastructures industrielles n'étaient que faiblement connectés à Internet : le risque d'attaques cybercriminelles reste ainsi trop souvent considéré comme faible. À de très rares exceptions près, le cyber-risque n'est pas considéré comme un élément structurant des architectures systèmes. Trop souvent les méthodes utilisées se limitent à un « disaster recovery plan », une prévention des dénis de services, un schéma de lutte contre les virus et autres malwares.

Avec 250 millions de véhicules connectés sur les routes dès 2020, il est évident que les business d'attaque des voitures et autres systèmes industriels vont devenir économiquement viables très rapidement. Combien une société de transports serait-elle prête à payer en voyant ses camions ne démarrant pas un lundi matin ? Quel serait le manque à gagner pour un patron pêcheur dont les traces, routes ou waypoints disparaîtraient suite à une mise à jour de son système de cartographie ?

Les récentes attaques ont montré que les systèmes actuels restent relativement simples à attaquer. À ce jour (octobre 2016), Tesla est le dernier à avoir fait la une de la presse avec l'attaque de ses voitures par un groupe de hackers Chinois[1]. Toutefois même s'ils essaient d'en minimiser la portée médiatique, la liste des constructeurs dont la cybersécurité a été mise en faute est longue. BMW a dû rappeler 2.2 millions de véhicules ; l'attaque des Jeeps durant l'été 2015 a coûté plus de 150 millions de dollars à Fiat-Chrysler. De nouveaux cas plus ou moins importants continueront à apparaître aussi régulièrement que les grandes marées sur les côtes Bretonnes : là où il y a un business, il y a toujours des fournisseurs.

L'attaque des Jeeps a cependant marqué un point de rupture, et ce n'est que depuis début 2016 que le risque cybercriminel est vraiment pris en compte comme un élément structurant des architectures automobiles. Avant 2016, la tendance était de considérer que puisque la cybersécurité était sous-traitée à un intégrateur qui en assurait l'entière responsabilité, les constructeurs n'étaient pas vraiment concernés. Au final, trop de monde pensait que l'attaque d'un véhicule ou d'un système industriel était si complexe, si longue et si chère que personne ne serait intéressé. Aujourd'hui, nous savons que ce n'est pas le cas. Les hackers et autres « chapeaux noirs[2] » sont très intelligents, très bien formés et ont le temps pour eux. Enfin, il y a suffisamment d'argent en jeu pour qu'ils trouvent facilement les financements leur permettant de mener à bien leur besogne.

Aujourd'hui, toute l'industrie admet que les voitures de demain, comme presque tous les systèmes industriels, seront connectés à Internet. Tous ces systèmes seront régulièrement attaqués et certaines attaques seront nécessairement couronnées de succès. Il faut donc mettre en place les mécanismes pour réduire les surfaces d'attaque et installer des défenses pour se protéger des risques connus, mais aussi prévoir des systèmes de mise à jour automatiques pour corriger les erreurs et se prémunir des attaques encore inconnues. L'âge moyen d'une voiture en Europe est de 9.5 ans pour une durée de vie de 20 ans. Personne n'est en mesure de prévoir les

1 http://www.01net.com/actualites/des-hackers-ont-reussi-a-pirater-une-tesla-model-s-a-distance-1039190.html
2 https://fr.wikipedia.org/wiki/Black_hat

technologies de cyberattaques qui seront disponibles dans 10 ou 20 ans : permettre la mise à jour du système n'est donc pas une option.

3 Pourquoi AGL (Automotive Grade Linux) ?

Comme rappelé initialement, les changements en cours représentent une révolution pour le marché automobile. Il est donc critique pour les grands constructeurs de ne pas rater la marche. Le risque pour les constructeurs automobiles de subir avec la voiture connectée le même sort que celui réservé à Nokia [1]avec le Smartphone est loin d'être nul.

Pour éviter ce scénario catastrophe, les constructeurs ont de nombreux challenges à surmonter :

- **Réduire le coût du logiciel :**

  À titre d'exemple, le coût facturé au client pour le système électronique d'une Mercedes Class E n'a augmenté que de 1650€ entre 2010 et 2015, alors que sur la même période, le coût pour le constructeur augmentait de 6500€[2].

  Si à court terme, cet écart reste acceptable sur les véhicules commercialisés entre 60 et 70k€, il est évident que d'une part il n'est pas soutenable sur les véhicules de gammes inférieures et que même sur le haut de gamme sur le long terme, les constructeurs devront réduire leurs coûts afin de continuer à innover sans faire exploser les prix.

  Dans le bas de gamme, le problème est encore plus critique : il faut compter 650€ pour un système de navigation standard intégré dans la voiture, alors que l'équivalent n'est facturé que 150€ par TomTom pour un système externe équivalent.

  Si les constructeurs ne veulent pas perdre le marché des véhicules connectés, ils doivent drastiquement réduire le coût de leur électronique embarquée.

- **Ne pas perdre la guerre du « Big Data » :**

  Dans les 50 milliards de dollars ajoutés au marché du «Big Data» par les véhicules connectés, une part non négligeable provient de la fourniture de services : le streaming de musique, la mise à jour de la cartographie, la gestion du trafic, etc. Les constructeurs sont dans l'obligation de trouver comment récupérer une partie des sommes attachées à la fourniture de ces services, car seule une petite part peut être facturée au client lors de l'achat du véhicule.

  Dans cette lutte, les constructeurs ont un point fort : eux seuls contrôlent les données fournies par leurs voitures.

  Toutefois ils ont aussi beaucoup de faiblesses : peu ou pas de connaissance au business du « Big Data » ; pour certains d'entre eux, une santé financière qui n'a rien d'exceptionnel et enfin le temps qui joue contre eux.

  À l'inverse, leur principal concurrent « Google », qui espère récupérer 30% de ce marché en devenir, a des budgets quasi-illimités. Il contrôle environ 80 % des téléphones mobiles, possède une cartographie mondiale ainsi que de nombreux autres services très utiles et qu'une grande majorité d'utilisateurs aimerait voir intégrés dans les voitures.

[1] https://www.linkedin.com/pulse/nokia-failure-story-ibrahim-abd-elaziz?forceNoSplash=true
[2] http://www.strategyand.pwc.com/media/file/Connected-Car-Study-2015.pdf

- **Aller vers la voiture autonome :**

  Même si la voiture autonome n'est pas pour demain, tout le monde s'y prépare activement. La quantité d'innovations nécessaires pour atteindre ce Graal est sans précédent dans l'histoire de l'automobile.

  Malheureusement, les plateformes actuelles ne sont absolument pas compatibles avec le rythme d'innovation effréné nécessaire à la progression vers des véhicules 100 % autonomes. Aujourd'hui, aucun constructeur ne possède des systèmes évolutifs et réutilisables qui lui permettraient de véritablement capitaliser l'expérience d'un véhicule à l'autre, d'une génération à la suivante.

- **Passer les freins sociologiques :**

  Les études montrent que même si les utilisateurs sont demandeurs de nouvelles fonctionnalités, ils ont aussi des craintes non seulement sur la cybercriminalité, mais aussi sur la protection de leur vie privée.

  Comme le montre l'étude de McKinsey[1], la relation entre voitures connectées et consommateurs est complexe : la majorité des consommateurs ne maîtrisent que très partiellement les enjeux attachés aux avantages/inconvénients des véhicules connectés et pourtant demandent de plus en plus de fonctionnalités qui imposent de connecter leurs véhicules à Internet.

  Ils veulent plus de confort de conduite, avec des cartes mises à jour automatiquement ou des listes de musiques synchronisées sur leurs préférences personnelles depuis leur domicile ou leur téléphone. Ils demandent plus de sûreté avec une détection automatique des écarts de route, des piétons, des distances de sécurité, etc.

  Toutefois, dans le même temps, ils craignent pour le respect de leur vie privée et restent sceptiques sur la capacité des constructeurs à les protéger des attaques cybercriminelles.

  Tant que les voitures connectées resteront destinées à des passionnés de technologie, il n'y aura que très peu de risques. En revanche, avant d'équiper monsieur Tout-le-monde, il faudra être certain que la technologie ne décevra pas, sous peine de voir un rejet en masse de plusieurs années par le marché.

Afin de dépasser un usage limité au véhicule de luxe et atteindre le marché de masse, les constructeurs doivent réduire les coûts des systèmes informatiques embarqués de manière drastique. Aucun constructeur n'étant assez puissant pour résoudre seul ce problème, ils ont dû se résoudre à développer des solutions communes afin de distribuer les coûts de recherche et développement entre différents acteurs de la filière. L'objectif des constructeurs est, tout comme dans la téléphonie, d'avoir une plate-forme partagée unique qui supporte tous les composants non visibles du client. Une fois les services de base assurés par cette plate-forme commune à bas coût, ils pourront focaliser leur effort financier sur la partie « expérience utilisateur » et les autres éléments qui fournissent de véritables différenciateurs sur leur marché.

AGL est l'un des principaux consortiums mondiaux travaillant sur le sujet; il regroupe tous les constructeurs Japonais mais aussi les grands intégrateurs mondiaux comme Continental ou Panasonic. S'il existe d'autres consortiums comme Genivi[2] en Europe, AGL est aujourd'hui celui qui est le plus avancé dans la fourniture d'une plateforme logicielle à destination des développeurs automobiles. C'est aussi la seule plateforme automobile qui intègre de manière native un modèle de cybersécurité de bout en bout.

[1] http://www.mckinsey.com/industries/automotive-and-assembly/our-insights/whats-driving-the-connected-car
[2] https://www.genivi.org/

4 Cybersécurité et Options Techniques

- **Garantir la source des logiciels**

Sur un site sensible, la première des mesures de sécurité consiste à vérifier l'identité des personnes. De la même manière sur un système informatique, la première des sécurités consiste à vérifier la provenance du logiciel. Authentifier la source d'un programme est la seule solution pour garantir que vous allez bien exécuter le code que vous pensez vouloir exécuter.

- **Signatures et PKI**

Bien que les techniques de PKI [1]soient connues depuis de nombreuses années, elles restent complexes à mettre en œuvre. Il faut une autorité pour créer les clefs, les distribuer et les révoquer. Il faut également trouver un système qui puisse protéger les véhicules à titre individuel, sans toutefois tomber dans une complexité trop grande qui serait impossible à gérer. Bien que le marché de l'automobile soit minuscule comparé à celui des téléphones, il faut tout de même compter une centaine de millions de nouveaux véhicules chaque année. De plus, la durée de vie (20 ans) ainsi que le nombre d'intervenants (fabriquant, sous-traitants, mécaniciens, propriétaires, locataires ou simples utilisateurs) rendent la gestion de la PKI bien plus complexe que dans une entreprise traditionnelle.

- **Avant le démarrage**

A l'inverse des systèmes d'entreprise qui sont statiques et dont l'accès peut au moins en théorie être régulé, il est impossible d'interdire l'accès physique aux systèmes embarqués d'un véhicule. Il faut donc configurer l'électronique d'une manière telle qu'elle ne puisse pas être compromise, même par quelqu'un ayant un accès matériel au dispositif :

  - Supprimer tous les mécanismes de bas niveau comme le JTAG qui pourrait être utilisé pour injecter du code malicieux directement en mémoire.
  - Garantir que le « bootloader » ne peut pas être compromis, par exemple en l'inscrivant dans une mémoire en lecture seule, directement dans le composant en usine.
  - Mettre en place des mécanismes de fusibles physiques, qui une fois « grillés » interdisent l'accès à toutes les fonctions de développement, de debug, etc.

A noter que toutes les protections doivent rester compatibles avec les procédures de maintenance, dont certaines peuvent imposer la réinitialisation totale du système dans un garage à partir d'un périphérique physique comme une clef USB ou une « valise » d'intervention.

- **Pendant le démarrage**

La première des choses est de garantir qu'on exécute le bon noyau système avec les bonnes options de lancement. Cette technique appelée « boot sécurisé », bien qu'un peu complexe à mettre en œuvre, est disponible sur toutes les électroniques embarquées destinées à la production.

Tous les composants critiques du système doivent être vérifiés par des clefs de signatures cryptographiques. De plus, Linux intègre un système appelé IMA [2]pour la mesure de l'intégrité de l'architecture et un autre appelé EVM pour module de vérification étendue. Ce dernier vérifie que les attributs étendus des fichiers qui

[1] https://fr.wikipedia.org/wiki/Infrastructure_%C3%A0_cl%C3%A9s_publiques

[2] https://wiki.gentoo.org/wiki/Integrity_Measurement_Architecture

contiennent entre autres les droits et privilèges n'ont pas étés modifiés de manière accidentelle ou malicieuse.

Outre la complexité de mise en œuvre qui doit rester acceptable pour ne pas faire exploser les coûts, les voitures ont un problème supplémentaire attaché à la vitesse de démarrage initial. A titre d'exemple, la vidéo de la caméra de recul doit pouvoir s'afficher seulement 2s après la mise du contact : il faut donc un système de démarrage non seulement sécurisé mais aussi très rapide.

- **Après le démarrage**

Une fois que le système a effectivement démarré, il faut pouvoir vérifier que les applications et services sont bien ceux que l'on pense qu'ils sont. Si les services restent assez souvent groupés avec le reste du système d'exploitation et peuvent donc être ajoutés aux vérifications IMA/EVM, c'est plus rarement le cas pour les applications qui ont en général un cycle de vie indépendant du système de base.

Pour les applications le système doit :

  - **Vérifier le code :** L'application doit être téléchargée dans un paquet scellé qui contient non seulement l'application, mais aussi ses règles et privilèges de sécurité. L'ensemble doit être protégé par des signatures cryptographiques qui garantissent l'intégrité et l'origine.

  - **Gérer les dépendances :** Installer une application impose presque toujours d'en vérifier les dépendances. Celles-ci doivent être disponibles et accessibles avec le niveau de privilèges attaché à l'application, sous peine d'annuler son installation ou sa mise à jour.

  - **Démarrer une application/service :** avant le lancement de tout processus, son intégrité doit être certifiée. Afin de limiter l'impact de ce contrôle sur les performances, il est indispensable d'utiliser les coprocesseurs présents sur la plateforme matérielle mais aussi d'implémenter des mécanismes de tampons pour limiter le nombre de calculs de signatures d'intégrité.

Garantir la source d'un composant et en assurer l'intégrité tout en restant capable de le mettre à jour pendant plus de 10 ans est un véritable challenge. Les expériences passées ont montré que perdre une clé pouvait compromettre globalement le système sur le très long terme. Il faut donc mettre en place un système qui soit à la fois souple, performant mais, une fois encore, dont la complexité reste suffisamment masquée afin de ne pas faire exploser les coûts.

- **Isoler et Compartimenter**

Comme dans le monde réel, la sécurité des logiciels est implémentée en isolant les contextes d'exécution. Dans le monde physique, on construit des grands murs avec des portes bien gardées ; dans le monde virtuel, on peut aussi utiliser des mécanismes d'isolations pour garantir qu'un contexte d'exécution ne puisse pas « déborder » chez son voisin.

  - **Virtualisation :** Elle peut soit être « hardware » et utiliser les caractéristiques avancées du processeur associé à un hyperviseur du type XEN/KVM ; ou alors comme dans le cas de Docker être purement logicielle et s'appuyer sur les mécanismes standards du noyau Linux comme les NameSpaces ou les Cgroups**.**

  - **Les contrôles d'accès** : Historiquement, tous les dérivés d'Unix supportent les contrôles dits « discretionary » où le propriétaire d'une ressource décide de qui peut accéder, modifier ou utiliser celle-ci. Les versions plus

récentes de Linux ont ajouté la notion de contrôles « obligatoires » où les règles d'accès ne sont plus fixées par le propriétaire de la ressource, mais par un administrateur de sécurité qui fixe de manière globale les règles d'accès.

- **Les restrictions de permissions :** elles peuvent être fixées pour un utilisateur donné, ou par application. Exemples : « un jeune conducteur n'a pas le droit de conduire à plus de 90km/h » ou « l'application Autoradio n'a pas le droit d'accéder à la caméra de recul ».

L'ensemble de ces techniques sont complémentaires : elles doivent toutes être combinées pour arriver à un système à la fois flexible, performant et dont le modèle de sécurité reste maintenable sur le long terme.

5 Le modèle de Cybersécurité d'AGL

Le projet AGL a adopté le modèle de sécurité de bout en bout proposé par IoT.bzh. L'annonce en a été faite lors de l'Automotive Linux Summit de Tokyo en Juillet 2016, et l'intégration dans les référentiels d'AGL est disponible depuis la version 2.0[1].

Bien que le modèle de sécurité d'AGL soit encore en phase de développement, il propose dès à présent un certain nombre de composants techniques qui permettent de commencer les développements.

- **Garantir l'origine du code**
  - **Boot sécurisé :** valorise les capacités des coprocesseurs de cryptographie pour accélérer le démarrage. Un mécanisme souple de gestion des clés pour supporter les petites séries et les modes de développement.
  - **Vérification des applications** : Les applications sont gérées via des paquetages au format « widget » (.wgt) du W3C. Lors de l'installation, le système vérifie non seulement l'origine, mais aussi que les permissions demandées sont en corrélation avec la source du logiciel.
  - **Gestion du consentement :** la plupart des applications nécessitent des permissions et il est important que l'utilisateur puisse conserver le contrôle de la plateforme, ce tout particulièrement pour les données qui concernent sa vie privée**.**

- **Une Architecture en Couches**

Le modèle d'AGL est basé sur une architecture en couches avec un modèle qui autorise l'exécution d'applications « non-trustées ». Le modèle de sécurité applicative s'appuie d'une part sur les mécanismes du noyau de Linux, comme les CGroups [2]ou les NameSpaces [3]ainsi que sur le mécanisme SMACK [4]pour le contrôle obligatoire des accès. D'autre part, la gestion de la communication inter-applications passe par un mécanisme de « binder [5]» qui effectue la vérification des permissions attachées aux APIs [6]à l'aide de la base de privilèges Cynara[1], sur le même principe que Tizen[2] chez Samsung.

1 https://gerrit.automotivelinux.org/gerrit
2 https://fr.wikipedia.org/wiki/Cgroups
3 https://en.wikipedia.org/wiki/Linux_namespaces
4 https://fr.wikipedia.org/wiki/Simplified_Mandatory_Access_Control_Kernel
5 http://docs.iot.bzh/docs/architecture/en/dev/reference/ap/binder/afb-overview.html
6 https://fr.wikipedia.org/wiki/API

- **Une architecture distribuée**

Le modèle de développement d'applications AGL s'appuie sur une transparence d'API qui permet au développeur d'écrire ses composants de manière réutilisable sans avoir à se soucier de l'architecture finale de déploiement.

Par exemple un agent d'acquisition d'un bus CAN [3]sera coupé en deux parties : d'une part un module traitant le niveau bas qui va récupérer les messages binaires et les interpréter afin de les rendre compréhensibles aux applications ; d'autre part, un module « logique business » qui est en charge d'implémenter la logique applicative (ex : signal « le véhicule bouge »).

Une fois développé, chaque module est comparable à une brique de Lego. Il peut être intégré soit dans un processus unique, soit sur des processus indépendants ou même avoir une partie exécutée dans la voiture et l'autre dans un service de type « cloud » quelque part sur Internet.

Évidemment, chaque modèle d'implémentation a un impact sur la sécurité comme sur les performances ; toutefois il est important de comprendre que ce n'est plus au développeur de décider du modèle de déploiement. Chaque module développé devient réutilisable dans des architectures différentes qui peuvent par exemple correspondre à différents niveaux de fonctionnalités et de prix des véhicules concernés.

- **Une vérification statique de la sécurité**

Une voiture connectée compte plus de 100 millions de lignes de code[4], avec une majorité de composants développés par des équipes externes sur lesquelles les constructeurs n'ont absolument aucun contrôle. Prétendre qu'il est possible d'utiliser les systèmes traditionnels de certification pour un tel assemblage tient au mieux de l'ignorance, au pire de la malhonnêteté intellectuelle. La seule solution est donc de restreindre par l'extérieur les capacités de nuisance des applications comme des services.

AGL 2.0 intègre un mécanisme de manifeste où chaque application déclare les ressources qu'elle compte utiliser ainsi que les APIs qu'elle expose. Ces demandes sont ensuite comparées aux permissions attachées au niveau de signature et de contrôle d'origine du composant concerné. Dans le cas où les demandes outrepassent ces autorisations, l'application n'est pas installée. En revanche, si elles sont acceptables, l'application est installée et la base Cynara de protection des APIs ainsi que les labels SMACK et CGroups sont provisionnés afin de garantir que l'application ne pourra jamais outrepasser les droits qui lui ont été attribués.

6 Mécanismes de cybersécurité en cours d'intégration

Bien qu'il soit déjà utilisable, le système de cybersécurité d'AGL est loin d'être finalisé. Beaucoup de travaux sont en cours, et parmi eux certains sont plus avancés et ont de bonne chance d'être intégrés pour la fin 2016 :

1 https://wiki.tizen.org/wiki/Security:Cynara
2 https://www.tizen.org/
3 https://fr.wikipedia.org/wiki/Controller_Area_Network
4 https://www.technologyreview.com/s/508231/many-cars-have-a-hundred-milion-lines-of-code/

- **Gestion de la Trusted Zone :** Ce système spécifique à la famille de processeurs ARM implémente un sous-système sécurisé qui assure une isolation « électronique » entre la partie« trustée » [1]de la partie système d'exploitation.

- **Hypervision :** Les processeurs sont de plus en plus puissants et afin de réduire les coûts, il faut regrouper les fonctions. Toutefois, pour faire cohabiter des systèmes avec des niveaux de sécurité ou de certifications différents sans craindre que l'un puisse déteindre sur l'autre, il faut impérativement implémenter une isolation proche du niveau « électronique » pour garantir une isolation parfaite entre chacun des sous-systèmes. Pour ce faire, on utilise des hyperviseurs et notamment XEN [2]ou KVM [3]qui sont des produits opensource disponibles en standard sur toutes les plateformes Linux. Historiquement utilisés dans le monde des serveurs, les hyperviseurs sont à présent disponibles également dans des formes adaptées aux systèmes embarqués.

- **Identité de l'utilisateur :** Beaucoup d'usages attachés aux services de « Big Data » nécessitent de vérifier l'identité de l'utilisateur. A titre d'exemple : sans savoir qui conduit le véhicule, le système ne peut pas fournir une liste des musiques préférées de l'utilisateur ou encore valider le paiement automatique d'un parcmètre. La gestion d'identité est aussi indispensable aux nouveaux modes de consommation des véhicules, notamment pour assurer la gestion des véhicules partagés.

- **Mise à jour « on the air » :** Aussi appelé SOTA[4], ce mécanisme est partiellement disponible dans AGL 2.0. Toutefois un travail d'intégration avec le système de développement comme avec le modèle de sécurité reste à faire avant qu'il puisse être classé comme composant natif d'AGL. La mise à jour automatique est un élément critique de la politique de cybersécurité de tout système connecté à Internet.

7 Conclusion

Comme pour l'ensemble de l'industrie connectée, le travail de sécurisation des véhicules de demain ne fait que commencer. À ce jour, les attaques connues sont toutes les faits de « white hats [5]» et les véritables modèles d'attaques malicieuses restent à découvrir.

Les attaques cybercriminelles peuvent être comparées à un incendie : si on intervient suffisamment vite, elles sont rarement graves. Avec le déploiement de centaines de millions de véhicules connectés, les premières véritables attaques ne devraient pas tarder. Il sera alors essentiel d'apprendre rapidement et de réagir dans les délais les plus brefs avant que l'ensemble du système ne s'embrase.

Comme toujours en sécurité, le diable se cache dans le détail. Avoir les bonnes briques de cybersécurité n'est pas suffisant : il faut s'assurer qu'elles soient parfaitement intégrées avec un niveau de complexité qui reste acceptable. Faire un système sécurisé simple est sans aucun doute le plus grand challenge des architectes de cybersécurité. Un système trop complexe n'a aucune chance d'être accepté : il ne serait pas maintenable, les utilisateurs trouveraient toujours un moyen de le contourner et le marché refuserait de le financer.

1 https://en.wikipedia.org/wiki/Trusted_system
2 https://www.linux.com/news/xen-virtualization-takes-automotive
3 http://www.linux-kvm.org
4 http://events.linuxfoundation.org/sites/events/files/slides/OTA%20Updates%20in%20AGL%20Using%20OSTree.pdf
5 https://fr.wikipedia.org/wiki/White_hat

Les véhicules de demain seront tous connectés et tous les véhicules connectés seront attaqués. Le risque est connu mais la bonne nouvelle est que les technologies pour le minimiser sont disponibles et matures. Construire une architecture sécurisée pour les automobiles connectées n'est pas forcement simple, mais reste néanmoins réalisable. Si l'industrie accepte de joindre ses efforts afin de trouver les financements compatibles avec la hauteur des enjeux, alors les véhicules connectés seront fiables et les constructeurs garderont le contrôle des données de leurs utilisateurs. Dans le cas contraire, il est probable qu'un « Google » prendra ce marché et réduira la valeur ajoutée des constructeurs de la même manière qu'il a réduit celle des fabricants de téléphones.

Références

- **Internet des Objets.**

1. https://fr.wikipedia.org/wiki/Internet_des_objets
2. https://en.wikipedia.org/wiki/Internet_of_Things

- **Big Data / Mégadonnées**

3. https://fr.wikipedia.org/wiki/Big_data

- **Analyses**

4. http://iot-analytics.com/iot-market-forecasts-overview/
5. http://www.gartner.com/newsroom/id/2819918
6. http://www.forbes.com/sites/gilpress/2014/08/18/its-official-the-internet-of-things-takes-over-big-data-as-the-most-hyped-technology/

- **Tizen**

7. https://www.tizen.org/
8. https://en.wikipedia.org/wiki/Tizen
9. https://www.tizen.org/blogs/bdub/2015/tizen-mwc-iot-growing-and-so-tizen

- **Automotive Grade Linux (AGL) / Genivi**

10. https://www.automotivelinux.org/
11. http://genivi.org/

- **Renesas**

12. http://www.renesas.com/applications/automotive/index.jsp
13. http://elinux.org/R-Car/Boards/Porter

- **Intel**

14. http://www.intel.com/content/www/us/en/internet-of-things/overview.html
15. http://www.theverge.com/2015/1/16/7555647/the-internet-of-things-is-already-a-2-billion-business-for-intel

- **Samsung**

16. http://pro.clubic.com/actualite-e-business/investissement/actualite-770482-sigfox-samsung.html
17. https://www.artik.io/

- **Consortiums IoT**

18. OCF (Open Connectivity Foundation) : https://openconnectivity.org/
19. IIC (Industrial Internet Consortium) : http://www.iiconsortium.org/
20. Allseen Alliance :https://allseenalliance.org/

Securing the IoT Jungle

Category: general

Dr. Jacques Fournier : Cybersecurity research program manager,
CEA LETI, jacques.fournier@cea.fr

Dr. Assia Tria : Technical referent,
CEA-TECH Paca, assia.tria@cea.fr

Abstract . The Internet of Things ("IoT") refers to an "infrastructure in which billions of sensors embedded in common, everyday devices […] are designed to record, process, store and transfer data." Thanks to the recent advances in miniaturization and the falling costs of electronic devices, the Internet of Things suddenly became relevant for the many different industries and end-users. In the IoT, objects are potentially interconnected with other objects and with broader networks like the Internet. This creates new risks, in particular to the confidentiality, authenticity and integrity of data exchanged between objects. The ubiquity of smart devices without physical protection and surveillance makes them easy preys to hardware and software attacks. These objects can be stolen, counterfeited and corrupted. The data stored on these devices could be accessible, including cryptographic data that would provide access to other sensitive data. Securing the IoT ecosystem is a multiple level problem which shall be key to the deployment of the associated technologies and their acceptance by the end-users.

Keywords: Internet of things, security, privacy, embedded systems.

The Internet of Things ("IoT") refers to an "infrastructure in which billions of sensors embedded in common, everyday devices […] are designed to record, process, store and transfer data". The IoT should not be considered a utopian concept. In the current state of things, it is based on several existing technologies such as RFID, Near Field Communication (NFC), sensors and actuators, wireless, communications machine-to-machine, the ultrawideband, Routing Protocol for Low power etc. According to Ericsson, the IoT will radically change carriers businesses, providing everything-as-a-service. Cisco predicts that by 2015, 25 billions of objects will be connected and this amount will double by 2020.

Thanks to the recent advances in miniaturization and the falling costs of electronic devices, the Internet of Things suddenly became relevant for several industries (smart manufacturing industries, connected vehicles, smart energy grids, smart city applications, connected medical devices…) and different categories of end-users. In the IoT, objects are potentially inter-connected with other objects and with broader networks like the Internet. This creates new risks, in particular to the confidentiality, authenticity and integrity of data exchanged between objects. For instance, an unauthorized access to any device of your personal sphere can reveal a lot of information about you: health, relations, behavior, location. A compromised node in a city management infrastructure can lead to traffic jams, or worse electricity and water blackout… The strong impact the IoT can have on cyber-physical systems pushes security to the main stage and makes it a key point of the potential IoT success and adoption by end-users.

ICT development has shown in the past that security is sometimes overlooked during the design phase, and its integration consequently causes technical difficulties and increasing costs, and can greatly reduce the quality of the associated systems. Given the particular nature of IoT systems with respect to "traditional" ICT systems, it is essential that the original design elements of the IoT comply with privacy and security as much as with all user requirements.

1 Particularities of the IoT

No one can precisely define what the IoT ecosystem will look like in the near future : current definitions depict a system of (inter)connected objects from applications as diverse and complex as automotive, domestic appliances, mobile phones, health care to critical infrastructure management. Trends either follow the concept of the "Internet of existing Things" where we must interconnect existing devices using current infrastructures to that of the "Internet of every Thing" where new devices and new technologies need to be deployed. In this paper, concerning the second concept, we will rather talk about the "Internet of future Things" because behind the fact of talking about connecting "every Thing", there are not only technological challenges about whether anything can be connected but above all societal issues about whether we want everything to be (inter)connected.

The complexity of defining the IoT comes from its **heterogeneity**: IoT refers to an ecosystem of interconnected objects, going from sensors embedded into everyday-life devices to complex SCADA-like infrastructures, through the use of smart phones and tablets… with different constraints in terms of protocols used, power consumption and communication interfaces.

Another particularity of the IoT is the **scale of deployment**: we are here concerned with billions of connected objects and any approach that shall be deployed in such a context shall be able to scale efficiently both in terms of technical feasibility and cost. The security services and resistance to attacks inherent to the IoT devices must also be able to scale with the large number of appliances and services that shall be deployed, given that they will become more and more accessible to hackers or malicious applications.

A third aspect that is emerging as a clear trademark of the IoT is the bargaining value of the data being collected and manipulated by the IoT devices. The latter can allow to collect, either directly (like monitoring one's location, health etc.) or indirectly (like learning about one's sleeping or working hours by simply having access to one's electricity consumption), information about individual end-users. This can not only have a serious impact on how privacy issues are managed at societal and judicial levels but also at economic and business levels. As shown so far by social media champions, end-user data and profiles are valuable information that can be monetized whether to advertisers or insurance companies.

The above particularities contribute in making of IoT a complex security challenge, linked to the fact that it is extremely difficult to have one threat model addressing all security concerns with expected societal and economic impacts.

Even through the concept of the IoT can be vague, the core components or constituent elements of a system composing an IoT infrastructure can be identified as

- The node which is a low power device basically composed of sensors, a small computing machine and a communication (often considered to be wireless) interface. Depending on the application, such a node can be embedded into a tamper resistant packaging. The node may also be fitted with an on-chip mechanism for energy harvesting in order to power itself.

- The gateway manages the interface between several nodes belonging to one same “Area Network” and the external connected (internet) world for aggregation and transmission of data.
- The server which may be “located” within a cloud infrastructure and which is responsible for the supervision/management of the associated network of gateways and nodes and the control and access to the network.

Currently, there is no single standard or infrastructure for implementing the IoT. It is built around several applications which have one common goal of improving users' and citizens' lives and driving down the costs of such services. Some of those applications are (non-exhaustively) illustrated in Figure 1.

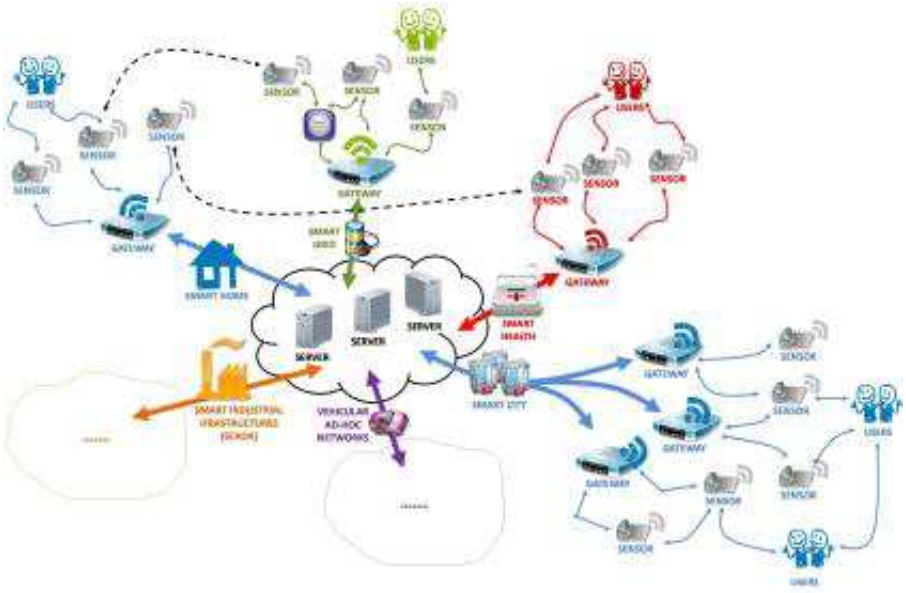


Fig.1. IoT ecosystem

2 Security challenges

One way of dealing with the issue of security in the IoT is to consider the latter as a conglomerate of different “vertical” applications, namely, as illustrated in Figure 1, smart health applications, smart energy grids, smart vehicles, smart city and smart industrial infrastructures. Each of those applications has its own assets and the risks associated to those assets give rise to “security priorities” which can be depicted as follows:

- In the case of smart health applications, the principal security concerns would be patient-focused. The devices used should not only be safe for the patient but they must also guarantee the confidentiality and integrity of the patient’s data in transit or at rest and at best protect the user’s identity.

- For smart energy grids, the main concern would be the resilience of the infrastructure to intentional attacks, whether they come from outside the network or from the inside. In this field we also have the issue of critical infrastructure protection. One central feature of those smart energy grids in terms of security is the smart meter because it can be an entry point for attacking an entire network and also because it can be easily accessible to attackers [1].

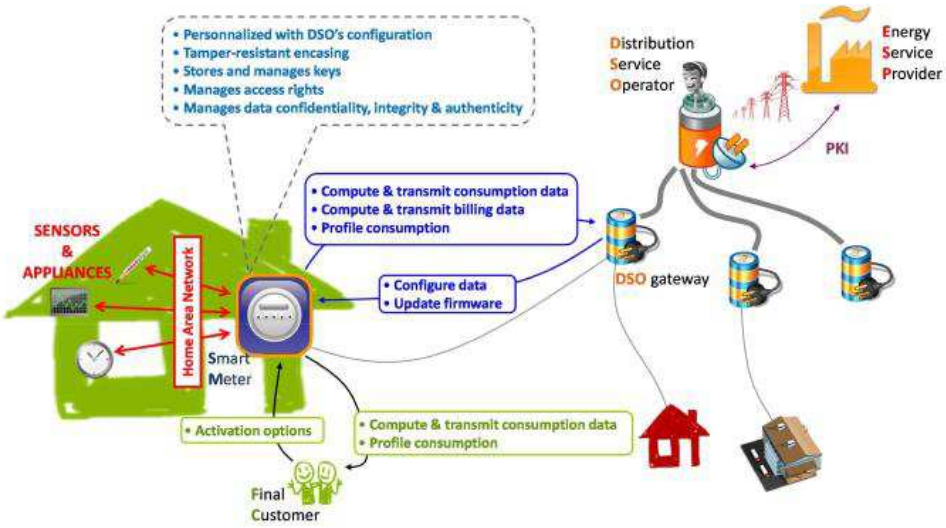


Fig. 2. Security issues pertaining to smart meter

- For smart vehicles or the heavily publicized field of connected autonomous vehicles, the integrity of the vehicle itself (i.e. the security of the embedded system) would be at the heart of the security associated to this field, especially when it comes to the physical safety of the vehicle's driver and passengers. The particularity of this market segment is that large investments have to be made on the device/vehicle side with the risk of user dropping at the smallest security/safety issue.

- The protection of specific Intellectual Properties (IP) and the resilience of the infrastructure are the main security issues raised by the deployment of IoT infrastructures in smart industries. Such risks have to be leveraged with the economic gains (in terms of production throughput, efficiency, streamlining maintenance) yielded by the "IoTization" of manufacturing sites.

Due to the specificities of these different use cases, specific dedicated security architectures have to be specified and implemented for each of those architectures. One striking observation that can be made for all those applications is that any disruption in the security chain of any of those applications can lead to large scale security problems (in terms of persons/victims affected, in terms of geographical spreading of the problems and their consequences etc.) leading national security issues even for infrastructures that are not primarily identified as critical infrastructures (like energy or distribution networks, lighting etc.).

However, doing such a security engineering work relies on the fact that the core security technologies are available "off the shelf", which is far from being the case today. The ubiquity of smart devices without physical protection and surveillance makes them easy preys to hardware and software attacks. These objects can be stolen, counterfeited and corrupted. Without specific countermeasures, the data stored on these devices would then be accessible, including cryptographic data that would provide access to other sensitive data. Moreover, wireless transmissions, could be easily eavesdropped. The security challenges of the IoT must be first addressed by looking at its constituent building nodes. The underlying paradigm is that by deploying secure, low power and functionally efficient nodes, secure, safe and resilient infrastructures can be built.

- Trust in the connected nodes

A first issue to be addressed is that of the trust in the IoT nodes, i.e. to what extent one can trust a given node to be whatever it claims to be or whatever it claims to be doing. To achieve this intrinsically, research work has been carried on the hardware authenticity and integrity of integrated circuits [2]. One of the objectives here is also to provide a scheme that covers for the overwhelmingly complex and growingly uncontrollable supply chains of ICs that shall be used in the IoT.

- Confidentiality, Integrity and Authenticity

IoT nodes are expected to provide the 'classical' security services of data Confidentiality, Integrity and Authenticity (CIA). There are cryptographic solutions to deliver services of CIA, but these are often computation-intensive and memory-greedy. They are not necessarily adapted to the stringent power and die size constraints of IoT nodes. Elliptic curves cryptography could provide a level of security similar robust cryptography asymmetric classic with the advantage of being inexpensive in terms of resources (memory, calculation and bandwidth). On this aspect, low power cryptographic hardware accelerators for asymmetric cryptographic algorithms like Elliptic Curves Cryptography must be designed [3].

- Privacy

Privacy shall be "by design". In general, integrating security features into an existing system can become very complex, sometimes impossible, and often increases the cost of the final product significantly. A more efficient approach is to take into account those security requirements at the early beginning of a project and integrate them in the design and development phase. It requires developing tools and mechanisms allowing privacy-by-design in the network. IoT technologies are emerging and promised to a bright future, this is the opportunity to think and design the security from the foundations. Investigation must be done around secure implementations of Pairing Based Cryptography which is up to now the only cryptographic tool able to guarantee user privacy in a connected world [4, 5].

- Low power and deployment

IoT devices are often simple low power sensors or actuators. They should then run protocols – including security protocols – that are lightweight, though providing an appropriate level of security. For example, entropy harvesting can be generated by using sensors, channel and battery information as entropy sources, in order to locally generate random numbers used in security protocols [6]. Moreover, Elliptic Curve Cryptography can provide lightweight security for such low power sensors. Usually, such sensors are autonomous and deployed at a large scale. A few thousands sensors can be deployed and therefore the security must be managed in a hierarchical way [7]. In the latter cited example, one of the aims was to easily isolate and potentially revocate nodes that may have been tampered with, without any impact on the functionality and the security of the entire.

- Management of heterogeneous systems

More than the different hardware capabilities of each node, the diversity in terms of constraints added by each type of network link involved in a communication session is a problem. The approach until now has been to establish “hop-to-hop” security. For instance, every packet received from a node is decrypted in the gateway then re-enciphered before to be sent to the server; with this approach gateways handle clear-text information and this may not be acceptable. In opposition to the “hop-to-hop” security, the third challenge for IoT security is to design new “end-to-end” security schemes. This means that the information will be secured from the sensor to the server without being deciphered anywhere else than at the destination. The challenge is now to design security features in protocols, new protocols must be designed in order to be compatible with Internet one and with the low computation capabilities of IoT nodes [8].

The main difficulties with deploying security for the IoT come mainly from the heterogeneity of this “system of systems”, lack of standard for security and inter-device communications, lack of off-the-shelf trusted IoT devices and tools (respecting all power, size and security constraints) and lack of appropriate business models (`low end' devices will potentially be communicating with `high end' ones…). Despite all those difficulties, securing IoT systems is not an option but a must. To overcome some of those hurdles, systems have to be rethought and redesigned and reaching such objectives shall take time. For that matter, one approach is to split the IoT into two categories representative of the technical security solutions that can be deployed.

3 Short term: Internet of existing things

In the short term, we should look at what we call the Internet of existing Things (IoeT) which is an IoT deployed using already existing technologies that have to be adapted and upgraded to address the different security & privacy issues of confidentiality, authenticity and integrity:

- Efficient, extremely low power implementations of cryptographic algorithms, either based on existing schemes like ECC or the AES, or by searching for more efficient algorithms based on Light Weight Cryptography. No matter which algorithms are chosen, their implementations have to be inherently resistant to physical attacks (side channel information leakages or sensitivity to fault injections [9].

- Low cost (or inherent) tamper resistant designs and designs resistant to physical (side channel and fault) attacks [10].

- Efficient key generation and storage realisations like technologies based on Physical Unclonable Functions (PUFs).

- Efficient key management procedures for exponentially growing infrastructures with the corresponding personalization model.

- Adapting the low level IP protocols to suit IoT requirements like it is already the case with the IPv6 Low power Wireless Personal Area Network (6LoWPA) [11].

- Efficient data fusion and management on the server side to predict system disruption and implement adapted and fast system healing and recovery from attacks.

4 Long term: Internet of future things

In the long term, we will look at the Internet of future Things (IofT) where systems can be completely rethought in order to use new technologies like

- Pairing Based Cryptography (PBC) that can solve the key management issue for a scalable IofT through IBE for example. One of the major issues for existing secure systems is the distribution of cryptographic keys. The way of dealing with this issue is the use and deployment of a Public Key Infrastructure (PKI) where certificates have to be managed and checked for every connected object by a Trusted Third Party. PKI infrastructures are relevant for today's networks made up of hundreds of millions of computers but shall not scale efficiently for tomorrow's billions of connected objects. Due to that, new cryptographic techniques, and the associated infrastructures, shall have to be deployed and one such technique could be based on Pairing algorithms where the identity of the device can be directly used as the encryption key for an asymmetric cryptosystem.

- Homomorphic encryption for secure and private data management and storage on the server side. Homomorphic encryption refers to a cryptographic scheme whereby arithmetic and Boolean operations can be performed directly on the encrypted data without needing to decrypt data. The main advantage provided by homomorphic encryption is the saving of computing resources, storage and energy which is fundamental in the context of the internet of things.

5 Taking things into perspective

Securing an object is very complex technically because the security target and the leading surface are naturally larger than for a 'standard' web application which may contain the servers for example. In the case of a communicating object, all imaginable attacks are nearly possible. The passive protections like "defense in depth" must be developed. At the present time, very few or no device to our knowledge implements such methods.

Among the difficulties met when implementing such solutions, we have the specific constraints associated to embedded systems like the available memory, the computing power or the power consumption. In that, unfortunately, 'classical' cryptography can be quite greedy on all three aspects! In order to address those various constraints, new alternatives must be researched and standardized: for example "Ligthweight Cryptography" could be an interesting approach for addressing all three constraints of memory, computing power and power consumption. The development of quantum computers about which we know that it makes very vulnerable the RSA-based systems and divides by 2 the 'level of security' currently associated to the different flavours (in terms of key size) of AES is an element to be taken into account for smart objects designed to be present in our houses, our offices and our cities for many years. It is therefore necessary to already work on post-quantum cryptographic architectures as Europe and the ETSI have begun to address. Anticipating attacks and progress in managing Qbits quantum computers is a high constraint in the technical development of future objects and applications.

Références

1. Trusted Computing for Embedded Systems", Candaele Bernard, Soudris Dimitrios, Anagnostopoulos Iraklis (Eds.), pp 135-142, Springer, ISBN 978-3-319-09419-9, November 2014.
2. http://www.hint-project.eu/.
3. G.Reymond, V.Murillo : “A Hardware Pipelined Architecture of a Scalable Montgomery Modular Multiplier over GF(2^m)”, in International Conference on Reconfigurable Computing and FPGAs (ReConFig), 12/2013.
4. R.Lashermes, J.Fournier, L.Goubin : “Inverting the Final Exponentiation of Tate Pairings on Ordinary Elliptic Curves Using Faults”. In Cryptographic Hardware and Embedded Systems (CHES 2013), Santa Barbara, CA, USA, 8/2013.pp 365-382.
5. R. Lashermes, M. Paindavoine, N. El Mrabet, J.A. Fournier & L. Goubin “Practical validation of several fault attacks against the Miller algorithm” in Workshop on Fault Diagnosis and Tolerance in Cryptography, Busan, September 2014.
6. C. Hennebert, H. Hossayni, C. Lauradoux: “Entropy harvesting from physical sensors”. In the proceedings of WISEC 2013: 149-154.
7. Christine Hennebert, Vincent Berg: “A Framework of Deployment Strategy for Hierarchical WSN Security Management”. DPM/SETOP 2011: 310-318.
8. http://www.iot-butler.eu/
9. “On the importance of considering physical attacks when implementing lightweight cryptography” by Alexandre Adomnicai, Benjamin Lac, Anne Canteaut, Laurent Masson, Renaud Sirdey, Assia Tria and Jacques J.A. Fournier, NIST Workshop on Light Weight Cryptography, October 2016.
10. “Secure Architectures: Requirements and Assessment” by Jacques Fournier, invited talk given at the 1st International Workshop on Embedded Reconfigurable Architectures for Applied Cryptography (CryptArchi 2003), January 2003.
11. C. Hennebert and J. Dos Santos, “Security protocols and privacy issues into 6lowpan stack: A synthesis”, Internet of Things Journal, IEEE, vol. 1, no. 5, pp. 384-398, Oct 2014.

Internet of Things:
Security Issues, Challenges and Directions

Ahmed Amokrane

CoESSI

amokrane.ahmed@gmail.com

Abstract. The Internet of Things (IoT) is experiencing an exponential growth and it is giving rise to various applications ranging from connected homes and cars, health monitoring and smart utilities to military and critical infrastructures. This holds the potential to empower and advance nearly every individual and business. However, this comes with a number of security risks businesses and consumers will inevitably face. In this paper, we investigate the security implications an impact of integrating and using connected devices. Specifically, we investigate the security challenges both technical and corporate, along with the attack surface which gets wider as connected objects are ubiquitous and interoperable. Throughout the paper, we provide real world examples of recent security problems in IoT devices as well as security tests on IoT solutions we carried out. Finally, we provide hints and research directions towards a secure IoT world.

Keywords: IoT, cybersecurity, risk, attack surface, penetration testing

1 Introduction

The Internet of things (IoT) is nowadays present in various domains ranging from healthcare, home automation and smart grids to critical applications such as factory automation and the military. Various sources forecast the number of connected devices to reach 20 to 34 billion in 2020 [1, 2], with a potential economic impact of $3.9 trillion to $11.1 trillion a year by 2025 [3]. To name a few, the US government alone spent on IoT solutions $8.8 in 2015 and almost $35 billion from 2011 through 2015 [4]. This exponential growth will cover further domains of application. In fact, IoT has the potential to empower individuals' and businesses', and it will bring an uncountable of benefits in the different domains of general public and industry [5].

What we know as IoT today is, in fact, an ecosystem of several components as illustrated in Fig. 1. At the edge, sensors and actuators are deployed to collect data on the controlled environment. These sensors communicate through wireless media in general with data concentrators and gateways. This data exchange usually uses IoT specific protocols such as ZigBee, Z-Wave, BT4LE, LoRa, SigFox… The data is then sent through carriers' networks to core cloud platforms for data storage and analytics. IoT solutions include also mobile Apps designed for users to interact with the deployed sensors and actuators. This interaction is achieved through the cloud platforms or directly with the gateways using high level Application Programming Interfaces (APIs) such as Restful APIs.

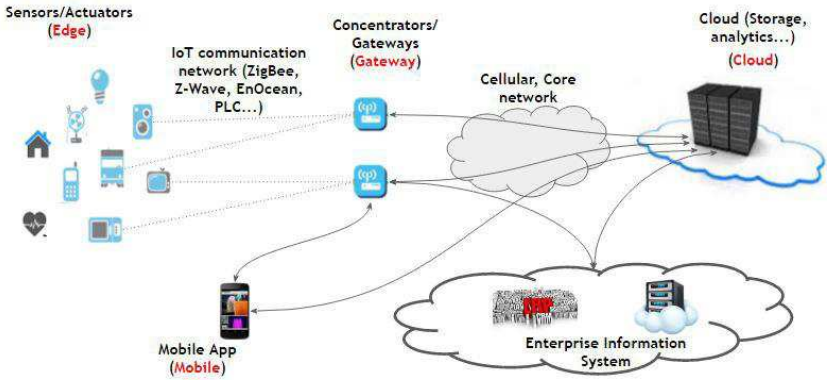


Fig. 1. Typical IoT Solution Architecture

Despite the benefits and the spectacular growth of IoT, security and privacy concerns are among the top factors that might slow down the rate of adoption according to various studies [5]. In fact, on the one hand, IoT manufacturers are not having the security concerns in their top priority list when developing hardware and software components for IoT [6, 7]. For instance, some industries such as the car industry did not, in the past, measure the risk brought by building IoT into cars [8, 9]. Also, manufacturing companies entering the IoT market may not have experience in dealing with security issues or lack economic incentives to provide strong security features, ongoing support or software security and updates [5, 10]. On the other hand, as we become increasingly reliant on intelligent devices in every aspect of our lives, they sometimes have access to critical and sensitive personal data such as social security numbers and banking information [7]. Moreover, some devices control the physical world, in what we call also Cyber-Physical Systems (CPS) [11]. As a result, malicious use by attackers can result in physical damage and public safety way beyond the logical attack. Consequently, having secure devices and communications is of paramount importance.

It is worth noting that adversaries could be simple curious people, more advanced hackers and hacktivists or state sponsored hackers. In light of this, numerous studies predict an exponential growth for the IoT security market. For instance, Markets and Markets predicts a IoT security market to grow from $6.89 Billion in 2015 to $28.90 Billion by 2020 [12], while Markets Research [13] predicts the IoT security market to grow 55% annually over the duration of 2015-2019. This reflects not only the security concerns and awareness, but also the opportunities and the role regulatory bodies and governments should play.

In this paper, we investigate the IoT security starting from a general perspective down to technical examples of possible attacks. More specifically, we first investigate the security challenges facing the IoT. These challenges are to be faced both at the risk management level (for instance in cases where IoT solutions are integrated into enterprise information systems) and at the technical level for building secure IoT solutions. To illustrate cases where security challenges are not properly tackled, we provide real examples of severe security issues in IoT products. Second, we investigate the attack surface in the IoT. As this attack surface gets wider, it spans more perimeters compared to traditional computer networks and information systems. Then, to demonstrate practical attacks, we present some of our own experiments on IoT devices through examples of web, wireless network and hardware attacks. Finally, we present our hints and thoughts towards securing IoT as a whole ecosystem. These hints address various aspects of security ranging from policy and regulation to technical ones.

The remainder of this paper is organized as follows. Section 2 present the related work. Section 3 presents the security challenges facing the IoT both at the risk management and the technical levels. Section 4 describes in more details the attack surface in the IoT. Section 5 presents examples of our own experiments on IoT devices. Section 6 stresses some points of improvements towards building a secure IoT. Finally, section 7 concludes this paper.

2 Related Work

IoT security is an issue that is being looked at by industry and the research community recently. In this section, we summarize some of the relevant works.

The first set of works looked at the security challenges and issues posed by IoT. For instance, Cardenas et al. [14] presented the challenges for securing cyber physical systems, which are by the same IoT solutions. They show the new attacks and attack vectors related to IoT domain, the difference with classical IT environment, and the security measures that apply. Authors in [15] looked at the specific challenges raised by the use of IPv6 in IoT infrastructures. In [10], authors presented some of the challenges facing IoT, which were clustered into four major characteristics of IoT devices: diverse threats, lack of resources, disparity in IoT quality and multiple points of vulnerability. Authors in [16] looked at the security challenges in IoT by looking at the legal framework that must be established at an international level. Few other works addressed the implications of building secure IoT devices. In [17], authors studied the tradeoff between security, safety, and availability in IoT applications such as healthcare. Authors show the possibility and some hints on finding the middle ground between security measures used to block access to devices and the safety of people in emergency cases.

The second set of works evaluated specific IoT devices and solutions. For instance, Wurm et al. [18] evaluated an IoT bridge (Haier SmartCare) and a smart meter (Itron Centron CL200) to show hardware vulnerabilities such as active UART and JTAG ports. In fact, they managed to exploit the ports and have unrestricted access to the device. Zillner et al. [19] addressed the practical security problem in ZigBee Light Link (ZLL) used by some IoT smart light solutions. They highlighted problems related to the use of a known master key for securing critical information exchange. Authors in [20] presented vulnerabilities they found across the attack surface from analyzing 50 smart home devices. Among their findings, none of the devices presented a mutual authentication for communication between clients and servers, 19% of the used mobile Apps in these devices did not use SSL and 15 web portals were vulnerable and allowed unauthorized access to the backend systems. Authors in [7] analyzed 10 popular IoT devices. Among their findings, 80% of them manipulated personal data, which presents a privacy concerns, 80% failed to require passwords of sufficient complexity, 70% did not use encryption for communication over the Internet, 60% had security holes in their web interfaces and 60% did not use encryption for software updates. Authors in [21] evaluated 14 IoT devices by looking at the hardware, PC Apps, mobile Apps and cloud communications. They showed that 12 devices out of the 14 provide active serial consoles. Other studies showed the use of hardcoded credentials in IoT devices. For instance, researchers from Sec-consult [22] analyzed the firmware images of more than 4000 embedded devices (Internet gateways, IP cameras, VoIP phones,...) of over 70 vendors.The researchers identified only 580 unique private keys for all of the devices. Furthermore, the researchers have shown that among these key, 230 are used by 4 Million IoT devices.

The last set of works focused on the penetration testing process and security evaluation of IoT devices.For instance, the OWASP IoT project [23] aims to build a panorama of the attack surface and top 10 vulnerabilities in IoT. More detailed methods such as NESCOR [24] are good references for building tools and show guidance for penetration testing of IoT. Other works presented specific tools for penetration testing. For instance, authors in [25] presented killerbee, a penetration testing tool for IEEE 802.15.4-based networks. The tool, which uses a specific

hardware dongle, can launch a set of attacks for traffic sniffing, clear text exchanged key extraction and association attack. In [26], authors presented a tool based on SDR for penetration testing. The presented tool includes implementation of ZigBee, Z-Wave and BT4LE in scapy, named scapy-radio. Authors in [19] introduced SecBee, a tool based on Killerbee and scapy-radio that exploits some vulnerabilities in ZigBee such as lack of proper cryptography for key transport.

In this paper, we present the investigate the IoT security from a general perspective down to technical examples. We focus on the security challenges and the attack surface by showcasing through real hacks published in the literature and through our own experiments.

3 Security challenges in IoT

The security challenges in IoT can be seen from two different perspectives: a risk management and a technological perspective. In this section, we examine these challenges and present examples of cases where they were not correctly tackled.

3.1 Risk management challenges

IoT presents risks that might already exist in traditional information systems. However, these risks are increased significantly in the IoT because of its pervasiveness [5]. In fact, IoT solutions are being integrated into information systems in enterprise and general public, sometimes without realizing it [27]. For instance, assets tracking systems and building management systems integrated to ERP are examples of this integration. Consequently, risk management stakeholders are not always informed and the risks are not well understood. These risks can have severe impacts on the safety of people, regulatory compliance, business image, users privacy and unauthorized access, for both manufacturers, resellers and users of IoT. Consequently, the risk management process is split among the stakeholders. In fact, a survey by SANS institute [28] demonstrated a shared responsibility for risk management between IT security group, device manufacturers, IT operations group and department managers. Consequently, manufacturer should be willing to take their part of responsibility in risk management related to their products.

3.2 Technical and technological challenges

Interoperability In the IoT world, the billions of connected devices are manufactured by different vendors. The point is to be able to integrate different devices from different vendors and offer a turnkey solution for endusers. In this case, standardized protocols as well as trust bootstrapping are needed. For instance, ZigBee Light Link protocol (ZLL), which is a ZigBee application level profile, uses a master key to secure the first packet exchange between ZLL compatible devices. To allow interoperability between manufacturers, this master key is shared by all compatible ZLL devices and is given by the ZigBee Alliance to certified manufacturers. This first exchange is used to share the network key, which is then used to secure the rest of the communications. As a result, the security of such a solution is tied to the master key being kept secret. Unfortunately, this master key was disclosed [19], which made this solution equivalent to having the network key shared in clear text. Note that we demonstrate this type of attack in section 5.

Physical security of deployed devices IoT devices might be deployed in harsh and hostile environments. For instance, parking meters, smart meters, public light sensors/actuators are deployed in the street. In this regard, physical security of these devices is not always guaranteed. This is a considerable attack vector as attackers might tamper with the devices to extract/inject data and software and even have control over to the network. For instance, the Hospira LifeCare Drug Pumps used for drug administration in hospitals use Wifi for communication and have an Ethernet port. The lack of

authentication for Telnet sessions can allow a user connected through Ethernet to recover the Wifi WPA keys [29]. Moreover, the device uses hardcoded local account credentials. Consequently, the attacker can access the "life critical network" that interconnects these devices and have the possibility to run commands and even update any device using malicious firmware. Consequences of such an attack could be people's physical integrity. Note that this can go as far as looking at the device hardware integrity all the way through its manufacturing and supply chain. In fact, one should expect that attackers will seek to compromise the supply chain of IoT devices to implement rogue hardware or software components.

Device, users and software authentication and authorization IoT devices are meant to operate independently without human intervention. In this regard, devices should identify themselves prior to being integrated into the network. Legacy Identity and Access Management (IAM) used in enterprise networks might not work due to scale and ubiquity of IoT devices. This authentication and authorization issue is also raised for users of these IoT devices. In fact, users should be authenticated and granted access to their required data. Moreover, devices should be able to authenticate the software issuer and check software integrity. Finally, having an ecosystem of billions decentralized devices and users makes this problem even more challenging.

Secure wireless medium for data transmission IoT devices use wireless communication protocols to exchange data. The wireless medium is accessible to anyone in the vicinity of the network, which makes the boundary of the network not restricted to a building but to its entire surroundings. This makes exchanged traffic accessible to attackers if no proper security controls are implemented. Moreover, the use of standard protocols with default parameters such as encryption keys or credentials can leave the network at risk of an attack without a need to physically be on site. To name a few, the Mitsubishi Outlander car uses Wifi for communication with a user's mobile App [30]. It was shown by researchers that the Wifi key is crackable within few hours [30]. In this case, anyone within the vicinity of the vehicle can connect to the car's Wifi network and issue commands towards the car. Another example is the TrackingPoint self-aiming rifles which are also vulnerable as they use a default WPA2 Wifi key that cannot be changed in addition to an unauthenticated API [31]. This makes the rifle accessible to any attacker in the vicinity, which can go as far as modifying the target of the shot.

The illusion of security through obscurity For decades now, security through obscurity was used by manufacturers and software developers to keep attackers at bay, and IoT solutions are not an exception in this regard. For instance, proprietary cryptographic algorithms and wireless protocols have been used. However, this is an illusion of security as reverse engineering efforts can result in hacks. To name a few, the SimpliSafe smart home alarm solution sold for over 300000 units in the US uses a proprietary wireless protocol for data exchange, which relies on a PIN code for security sent in clear text. A security researcher managed to reverse engineer the wireless protocol [32]. Consequently, he managed to send messages for alarm disabling by just being at the vicinity.

Internet access to devices and cloud platforms IoT is meant to have Internet connection brought all the way down to the devices. This makes them discoverable using specialized search engines such as Shodan [1] or Cencys (censys.io) services they offer, open ports geographical location, etc. Moreover, IoT solutions use data aggregation and cloud platforms for data storage and processing. These platforms are accessible by the mobile devices through the Internet. Consequently, a vulnerability in these cloud platforms can make the IoT solution vulnerable to any attacker around the globe. To name a few, NISSAN's leaf connected car is vulnerable to remote access as it uses a cloud platform for data and command exchange between a mobile App and the car [33]. As a result, commands for turning on/off the AC of the car can be

[1] Shodan.io

issued to virtually any LEAF car given that we know the car's Vehicle Identification Number (VIN) and country.

Secure APIs for communication IoT devices use APIs (usually Restful APIs) for communication with cloud platforms and mobile Apps. These APIs should be lightweight due to network and hardware constraints on the devices. As a result, they are sometimes hard to secure to guarantee full security of exchanged data. For instance, the NISSAN Leaf non-secure Restful API [33], the Mitsubishi Outlander [30], the TrackingPoint rifles [31] are all relevant examples as no authentication is required for using the API.

Secure updates and patch management IoT devices should allow for updates and patch management to keep them updated about the security issues. However, this is challenging for different reasons. For instance, some IoT devices do not have enough network bandwidth for frequent updates, such as the ones that use SigFox networks. Other constraint are also related to resources available on the devices to perform these regular updates such as storage capacity and battery. In addition to that, a fleet of millions of connected devices is a great deal to manage and update at a regular basis.

Publicly available mobile Apps The mobile Apps used by most of IoT solution are publicly available. In some cases, this could be exploited by attackers to reverse engineer and gain access to the IoT devices if hardcoded credentials or any security related parameters are present on the mobile Apps' source code. For instance, in the NISSAN Leaf [33] and the Mitsubishi Outlander [30] examples, researchers analyzed the mobile Apps to understand the Restful API that manages the car. From there, they could forge requests directly towards the cars without a need for the mobile App.

Resources and cost constraints IoT devices are usually cheap solutions made for the general public. The security features are, thus, not counted for in the product budget. Consequently, putting specialized technology such as Secure Element (SE) could be cost prohibitive. Moreover, adding complex cryptographic operations is not always possible due to resource and performance constraints on IoT devices. To add to the matter, IoT products seek to strike a tradeoff between security and efficiency [10]. In fact, implementing confidentiality, integrity, and availability measures impacts negatively the optimization of energy or/and computational resources.

4 IoT attack surface

The attack surface in IoT gets wider as more attack vectors are present. In this section, we investigate this attack surface.

4.1 Web and administrative interfaces

Web and administrative interfaces are used for data access and administration/management purposes. They are either embedded in the sensor/actuators, gateways, cloud applications and the mobile Apps. A lot of these interfaces are accessible through the Internet. Consequently, these interfaces can lead to full control over the target devices if exploited by an attacker. Vulnerabilities related to this attack surface include the use of weak password for web interfaces, lack of password/role management, use of hardcoded credentials, weak access control, lack of input validation, etc.

4.2 Cloud infrastructures

The use of third party cloud infrastructures can present an entry point to the IoT solution. In fact, a poor isolation between tenants in a cloud platform or an attack on the cloud provider can have consequences on the hosted IoT backend. This can include web applications but not limited to. Moreover, even secure applications can present privacy and regulatory concerns depending on the cloud provider and its location. For instance, data ownership and data access rights are common issues that can result in privacy-related consequences. Vulnerabilities related to this attack surface include the use of non-verified third party cloud platforms, lack of isolation between cloud tenants, lack of agreements on data ownership, etc.

4.3 Mobile Apps

The mobile Apps in some solutions give the user full access to to manage and control the IoT solution. Tampering with the mobile App through sensitive data extraction or data/command forgery can let an attacker have access to the IoT solution. Moreover, these mobile Apps are usually public, which make easily accessible for attackers. Vulnerabilities related to this attack surface include using hardcoded credentials in the Mobile App's code, improper session handling, broken cryptography, etc.

4.4 Wireless network communications

The use of the wireless medium for data transmission makes it possible for an attacker to record signals, carry different types of attacks such as jamming, traffic injection, traffic capture and analysis, communication protocol reverse engineering, over the air information extraction etc. Moreover, traditional network communications are also an attack vector to consider. In fact, most of IoT solutions use traditional networks to communicate, share data and reach out to cloud platforms. Vulnerabilities related to this attack surface include the use of clear text data exchange, broken cryptography, lack of device/data authentication, use of obscurity instead of encryption, etc.

4.5 Communication APIs and web services

The high level communication APIs are an important attack vector as they can allow an attacker to access exchanged data and control devices. They can also be used by attackers to inject traffic and commands. These APIs are independent from the underlying network media and technologies, so common attacks such as eavesdropping and DoS are also possible. Vulnerabilities related to this attack surface include the use of clear text APIs, broken cryptography, lack of authentication and authorization, unnecessary accessibility over the Internet, etc.

4.6 Hardware

The IoT devices' hardware parts and components present also a new attack vector. In fact, accessible USB, JTAG, UART ports can be exploited to gain unauthorized access and control over devices. Also, direct access to storage components (e.g., EEPROM, Flash memory) can allow an attacker to dump content (industrial secret) or inject malicious software (e.g., rogue firmware with rootkits and backdoors). Other attacks on the environment surrounding the devices is also possible such as tampering with sensors to force them to send false data, which might impact the IoT application. Vulnerabilities related to this attack surface include the use of a non-secure boot process, the use of non authenticated firmware updates, active serial ports, cleartext storage of critical information (e.g., passwords, crypto keys), lack of physical protection, etc.

4.7 Device software updates

IoT should have the ability to be updated to patch vulnerabilities and add functionalists. Moreover, these updates should be done remotely to allow IoT solutions to continue working while being updated. However, these updates can be a considerable entry point for an attack. In fact, an attacker can intercept software updates and extract sensitive information such as hardcoded credentials. Furthermore, an attacker can exploit these updates to inject malicious code into the devices. Moreover, device updates can be disrupted by an attacker to continue to exploit existing vulnerabilities. Vulnerabilities related to this attack surface include cleartext software updates, lack of authentication of software, storage of software updates in mobile Apps, etc.

5 IoT attack and exploitation showcase

In this section, we present some example on exploitation and attacking IoT devices. This is meant to showcase the accessibility of IoT attack surface.

5.1 Finding Internet connected devices using IoT search engines

In this example, we show how we can find in few seconds online vulnerable webcams using the Shodan search engine. Shodan is a search engine that crawls the Internet for connected devices and provides open ports as well as other information provided by these devices. For instance, we search for webcamxp. This search can be refined using filters for countries webcamxp country:FR. The search result is shown in Fig. 2. We can see that the search results gave 44 IP cameras. Some of them are accessible without any authentication as the

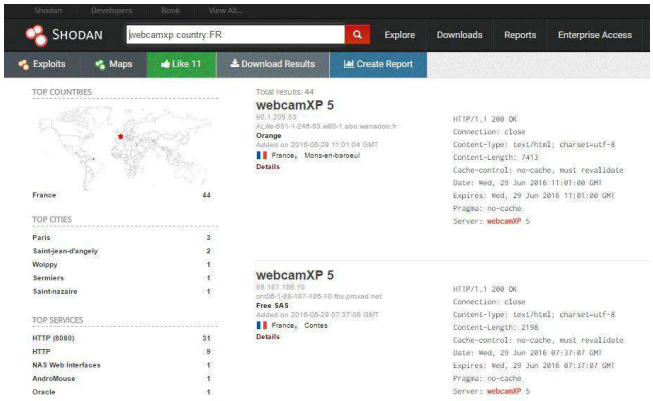


Fig. 2. Example of vulnerable IP camera localization using Shodan

live video stream can be seen through the Internet without any authentication. Fig. 3 shows such an example.

Fig. 3. Example of vulnerable IP camera live streaming

5.2 Capturing over the air (OTA) exchanged information

In this section, we look at the security of certain wireless protocols. More specifically, we show cases where exchanged cryptographic keys over the air in a non secure way is possible. In fact, to achieve interoperability, some protocols allow for key exchange in clear text or using default master keys. For instance, ZigBee is one of these protocols. In fact, a ZigBee network is composed of coordinators (usually one) and routers that communicate with the coordinator. The coordinator is responsible for key distribution to the new routers joining the network by transmitting a network key for each new joining router. To illustrate this example, we captured the association of a new router to a coordinator. To keep the demonstration generic, we used two XBee ZB modules. The first XBee is configured as a router and the second as a coordinator. In the rst example, we used the coordinator to transmit the network key to new joining router in cleartext, which is the default option when no Link Key (Master key) is configured. In the second example, we used a master key (Link key) in both the coordinator and the router to encrypt the network key exchange. To capture the traffic, we used our developed tool Wiotex (Wireless IoT EXploitation tool). This tool is inspired by Killerbee [25]. However, it is meant to be generic and be used for different protocols and not limited to ZigBee. In fact, it uses SDR and the scapy-radio [26] for traffic capture and injection. It extends the related ZigBee protocol implementation given in scapy-radio. Wiotex implements our traffic analysis and packet crafting layers. Among Wiotex features, the ability to intercept keys transported in packets either in clear text or encrypted using a known/guessed master/link key such as the ZLL standard. Wiotex has the ability to work on PCAP files or by live capturing packets using an SDR device.

In the first example, traffic is captured and stored in a PCAP file and then analyzed by Wiotex for key extraction. As we can see in Fig. 4, the key is extracted by Wiotex.

```
Extracting OTA keys from the pcap file :
/tmp/capture.pcap
We are using the following default Master/Link keys :
9F:55:95:F1:02:57:C8:A4:69:CB:F4:2B:C9:3F:EE:31
5A:69:67:42:65:65:41:6C:6C:69:61:6E:63:65:30:39
C0:C1:C2:C3:C4:C5:C6:C7:C8:C9:CA:CB:CC:CD:CE:CF
00:00:00:00:00:00:00:00:FF:FF:FF:FF:FF:FF:FF:FF

..........

We could extract the following key exchanged in clear text from the file :
BA:BB:74:5B:4B:23:CC:AC:7E:21:75:AD:68:68:CF:D2
```

Fig. 4. Key extraction from PCAP file using Wiotex

In the second example, we explicitly provide Wiotex with one or multiple Link keys (dictionary) to use to try to decrypt the key exchange. The results are shown in Fig. 5. From the figure, we can see that Wiotex intercepts the exchanged key after decrypting the payload using one of the provided Link/Master keys. Note that this attack can be combined with hardware attacks, where the hardware attack is used to extract the Link/Master key stored in the devices as described in [34]. Even though traffic analysis is possible using tools such as Wireshark or Ubiqua, information extraction in this case is done manually. Wiotex automates the extraction and allows for Link/Master key dictionnary attack.

It is worth noting that Wiotex has other features such as channel jamming, traffic entropy analysis, fuzzing, network mapping, association related attacks, etc. Wiotex is under development and testing for implementation of different network attacks and different IoT wireless protocols. In fact, other protocols can also vulnerable to this type of critical information leakage.

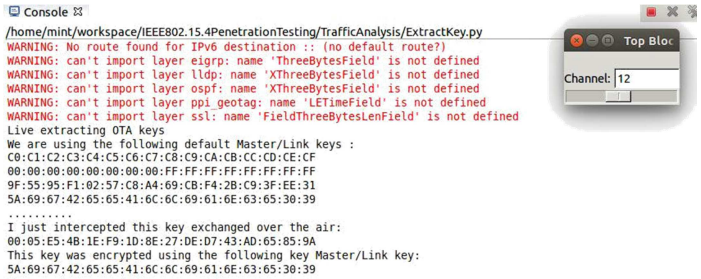


Fig. 5. Live OTA key interception using Wiotex

5.3 Hardware access to take over IoT devices

In this section, we present some among the numerous possibilities for accessing IoT devices by means of hardware. Among others, we show the example of extracting the content of a flash memory on a motherboard. To do so, we open the device and plug directly to the flash memory chip using simple wires or a specialized. Then we use a hardware dongle such as the Bus Pirate for communication between the our laptop and the target flash memory. On the laptop, we use tools such as Flashrom to dump memory contents. The testing set up is shown in Fig. 6.

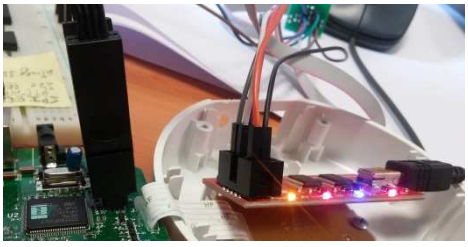


Fig. 6. Hardware exploitation testing setup

Note that other hardware testing tools exist such as Hardsploit[1]. Once the content is dumped, we can use additional tools such as binwalk to parse the binary. For instance, the result of running binwalk on the binary is shown in Fig. 7. From the figure, we can see that the flash memory contains a firrmware for a Broadcom chip with a file system packed in squash fs. Extracting the file system is straightforward using binwalk. Consequently, the confidentiality of the firmware is not protected in this case. Further exploitation could be to modify the firmware and pack it back using tools such as firmware mod kit and to load it back into the flash memory. The target device will use this modified firmware if no signature is used to check software authenticity, which is the case in our testing setup.

Additional exploitation methods using UART, JTAG/SWD ports is also possible and are not illustrated in this paper. Please refer to [34] for further details and explanation.

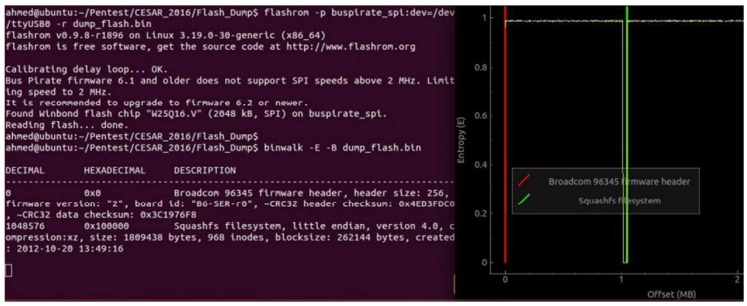


Fig. 7. Flash memory content analysis using hardware exploitation

6 Towards a secure IoT

The IoT stakeholders ecosystem is quite complex. Consequently, efforts should be put at different levels and by the different stakeholders. In the following, we present some directions towards achieving a secure IoT.

6.1 Built-in security into IoT

Security by design should be considered when it comes to devices that manipulate sensitive data or control sensitive processes. Also, secure development lifecycles and low latency patch management capabilities should also be introduced. Additional controls such as secure boot, software authenticity and integrity validation, role-based access control in operating systems should also be integrated at design and manufacturing stages. Moreover, security by default should be built into devices such as forcing users to change default passwords, impose password policies, change default encryption keys and certificates, use of secure APIs only, disabling serial ports (e.g., JTAG, UART...), etc.

6.2 Security as an added value in IoT devices

Although currently security does not yield financial motivations for manufacturers, it will change soon. In fact, compliance with privacy, data protection and ethical practices will win costumer trust, and eventually result in financial gains [10]. Moreover, security awareness is improving among IoT users, which can drive customer choices.

[1] hardsploit.io

6.3 Regulation and security norms for IoT

As of now, few if any certifications exist for IoT devices. As a result, there is no obvious possibility for a consumer to know what level of security an IoT solution offers. In this regard, governments and regulatory agencies should drive the efforts towards defining certifications for IoT devices with regard to their built-in security. Moreover, regulation on both ownership and access to data generated by these devices and used by different stakeholders should be clarified, easily understandable and enforceable. For instance, customers should know who collects and uses which part of their IoT data.

6.4 Continuous risk assessment

As IoT solutions are taking over physical processes and are being integrated into enterprise, the risk they introduce should be assessed an managed. In fact manufacturers should conduct a privacy and security risk assessment of their IoT solutions as part of the design process [6]. Moreover, enterprises should clearly understand the new risks introduced into their information systems, and conduct a holistic risk assessment/management. The risk assessment will identify where security controls are needed. Furthermore, the risk assessment should be maintained and updated as the infrastructures evolve and new vulnerabilities are discovered. It is also worth noting that a balance between risk and benefits of introducing IoT solutions should always be studied.

6.5 Policy and procedures for using and managing IoT

As IoT devices are being integrated into enterprise, existing policy and procedures should be extended to cover these devices. For instance, patch management and password policies should apply for IoT devices. Note that this is not common practice today. In fact, a survey by SANS institute [28] showed that 46% of respondents did not have a policy that drives the necessary level of visibility and management of IoT devices and 22% responded by "unknown", which is roughly equivalent to that fact that there is no policy in place.

6.6 Continuous monitoring

IoT solutions are ubiquitous and can be heavy traffic generators. In this regard, continuous monitoring through Intrusion Detection Systems (IDS) is necessary in a lot of cases to detect attacks and unusual traffic patterns. Attack detection can range from signal jamming and traffic injection to hardware tampering. Some existing solutions for traditional networks can be used. However, more aspects related to IoT such as hardware tampering detection should be accounted for. Note that monitoring is not common practice as only 41% confirmed collecting monitoring data in a SANS study [28].

6.7 Technical penetration testings

The implementation of security controls and the permeability of the IoT devices, networks and infrastructure should be evaluated and up to date. As such, vulnerability intelligence, patch management policies and regular penetration testing are needed to maintain a security level. Existing references and methodologies such as the OWASP [23] works and NESCOR [24] can be of help in this regard.

6.8 Hiding IoT devices from the Internet whenever possible

IoT devices' exposure to the Internet makes possible their discovery (using shodan for instance) and remote attack from all over the world. To prevent such attacks, air-gaping, i.e., isolating from the Internet, is a first step whenever possible. The air-gaping technique have been in use in Industrial Control Systems from decades now, where devices

such PLCs are only connected into a local network without any connexion to the Internet whatsoever. In cases where remote access is mandatory, other techniques for hiding the devices are also possible. For instance, recent works suggest hiding IoT devices from discovery using privacy-centered networks such as TOR [35]. This makes the devices not discoverable when searching the Internet.

7 Conclusion

Internet of Things is a powerful concept that is going to change the way people live, commute, work and entertain. Moreover, IoT applications can bring a lot of added value into different business sectors. However, this comes at significant security risks due to a larger attack surface and ubiquitous threats. Consequently, cybersecurity considerations should be carefully taken into account. In this regard, in addition to existing practices in networks and information systems, cybersecurity experts should face the challenge of managing a much broader set of interconnected items incorporating wearable devices, sensors and technology that is not currently deployed. More specifically, a lot of effort should be put for risk evaluation and management, along with technical auditing and penetration testing for deployed solutions. Finally, a security by design and pushing for security as an added value in IoT solution is of paramount importance to gain the trust of customers.

Acknowledgment

We thank Audrey Zambrano for her work at CoESSI on the development of Wiotex, Herv Daussin for his help and the reviewers for their comments.

References

1. J. Camhi, "BI Intelligence projects 34 billion devices will be connected by 2020," BII, Nov 2015. [Online]. Available: http://read.bi/1M5Vk3p
2. "Gartner Says 6.4 Billion Connected "Things" Will Be in Use in 2016," Gartner Press Release, Nov 2015. [Online]. Available: http://gtnr.it/1kJqJPO
3. McKinsey, "Unlocking the potential of the Internet of Thing," McKinsey & Company Report, June 2015. [Online]. Available: http://bit.ly/213q3VE
4. A. Meola, "The US government is pouring money into the Internet of Things," BII, May 2016. [Online]. Available: http://read.bi/2bv8IkL
5. F. T. Commission, "Internet of things: Privacy & security in a connected world," FTC STA Report, January 2015.
6. E. Ramirez, "Privacy and the IoT: Navigating Policy Issues," Opening Remarks of FTC Chairwoman for CES, Las Vegas, January 2015.
7. H. P. Enterprise, "Internet of things research study," Research Report, 2015.
8. FBI, "Motor Vehicle Increasingly Vulnerable to remote Exploits," FBI Public Service Annoucement, March 2016. [Online]. Available: http://bit.ly/2bAJQIR
9. P. Szoldra, "The truth about car hacking is scarier than we realized," techinsider.io article, June 2016. [Online]. Available: http://til.ink/290mk5s
10. D. Kostadinov, "Security Challenges in the Internet of Things (IoT)," Infosec Institute post, Nov. 2015. [Online]. Available: http://bit.ly/1TiRd6b
11. R. Baheti and H. Gill, "Cyber-physical systems," The impact of control technology, vol. 12, pp. 161{166, 2011.
12. "IoT Security Market by Technologies, Industry Verticals and Applications - Global Forecast to 2020," Markets and Markets Report, Aug 2015.
13. "Global IoT Security Market 2015-2019," Market Research Reports Study, March 2016. [Online]. Available: http://bit.ly/1UzQ4sN

14. A. Cardenas, S. Amin, B. Sinopoli, A. Giani, A. Perrig, and S. Sastry, "Challenges for securing cyber physical systems," in Workshop on future directions in cyber-physical systems security, 2009, p. 5.
15. T. Heer, O. Garcia-Morchon, R. Hummen, S. L. Keoh, S. S. Kumar, and K. Wehrle, "Security challenges in the ip-based internet of things," Wireless Personal Communications, vol. 61, no. 3, pp. 527{542, 2011.
16. R. H. Weber, "Internet of things{new security and privacy challenges," Computer Law & Security Review, vol. 26, no. 1, pp. 23{30, 2010.
17. R. Altawy and A. M. Youssef, "Security tradeoff s in cyber physical systems: A case study survey on implantable medical devices," IEEE Access, vol. 4, pp. 959{979, March 2016.
18. J. Wurm, K. Hoang, O. Arias, A.-R. Sadeghi, and Y. Jin, "Security analysis on consumer and industrial iot devices," in 21st Asia and South Pacific Design Au-tomation Conference (ASP-DAC). IEEE, 2016, pp. 519{524.
19. T. Zillner and S. Strobl, "Zigbee exploited-the good, the bad and the ugly," Black Hat USA, vol. 2015, 2015.
20. M. B. Barcena, "Insecurity in the internet of things," Symantec Report, 2015.
21. W. Wineberg, "Internet of things: Hacking 14 devices," in DEF CON 23, 2015.
22. "House of Keys: Industry-Wide HTTPS Certificate and SSH Key Reuse," sec-consult.com blog post, Nov 2015. [Online]. Available: http://bit.ly/1Nefie3
23. "OWASP Internet of Things Project," OWASP IoT website, 2016. [Online]. Available: http://bit.ly/1k0dSrD
24. J. Searle, "NESCOR Guide to Penetration Testing for Electric Utilities," National Electric Sector Cybersecurity Organization Resource (NESCOR) publication, 2013. [Online]. Available: http://bit.ly/1NXzSQS
25. J. Wright, "Killerbee: practical zigbee exploitation framework," in 11th ToorCon conference, San Diego, 2009.
26. J. Picod, A. Lebrun, and J. Demay, "Bringing software defined radio to the pene-tration testing community," in Black Hat USA Conference, 2014.
27. ISACA, "Internet of Things: Risk and Value Considerations," ISACA Internet of Things Series White Paper, 2015.
28. J. Pescatore and G. Shpantzer, "Securing the Internet of Things Survey," A SANS Analyst Survey, Nov. 2015. [Online]. Available: http://bit.ly/1Nefie3
29. E. Kovacs, "Serious Security Flaws Found in Hospira LifeCare Drug Pumps," securityweek.com article, May 2015. [Online]. Available: http://bit.ly/2aQlTAB
30. PenTestPartners, "Hacking the Mitsubishi Outlander PHEV hybrid," Pentest Partners blog post, 2016. [Online]. Available: http://bit.ly/1TW4pCK
31. R. A. Sandvik and M. Auger, "When iot attacks: Hacking a linux-powered rifle," Black Hat USA, 2015.
32. A. Zonenberg, "Remotely Disabling a Wireless Burglar Alarm," IOActive blog post, February 2016. [Online]. Available: http://bit.ly/1KrWAAA
33. T. Hunt, "Controlling vehicle features of Nissan LEAFs across the globe via vulner-able APIs," Blog post, February 2016. [Online]. Available: http://bit.ly/1UDGIfo
34. A. Amokrane, "Les pentests materiels dans les environnements iot," Multi-System and Internet Security Cookbook (MISC), vol. 84, March 2016.
35. M. B. Barcena, "Now you can hide your smart home on the darknet," Symantec Report, July 2016. [Online]. Available: http://bit.ly/29Wne3O

State of the art of IETF security related protocols for IoT

(catégorie : générale)

Renzo Navas[1], Laurent Toutain, and Kumaran Vijayasankar[2]

[1] Telecom Bretagne, Department of Network, Security and Multimedia,
Cesson-Sevigne, France
firstname.lastname@telecom-bretagne.eu

[2] Texas Instruments, WCS R&D,
Dallas TX, USA
kumaran@ti.com

Abstract. The Internet of Things (IoT) is becoming a reality and the Internet Engineering Task Force (IETF) is the main open standardization body responsible for it. The IoT implies billions of new devices connected to the Internet and, while several problems like interoperability and routing have been solved, security solutions suited for IoT are still an active field of research. This document is a survey of the state of the art at IETF of security related protocols for IoT. The needed IETF background and a highlight of current efforts on security for IoT is offered. An insight of unsolved problems and future perspectives on IETF concludes this survey. This is an informational document and detailed description of the protocols is not on scope.

Keywords: ietf, iot, security, protocols, standarization, survey

1 Introduction

This document gives an overview of the current state of the art of the Internet Engineering Task Force (IETF) working groups dealing with several aspects of security on constrained nodes and networks. The security aspects addressed by the presented protocols include: encryption (confidentiality), message authentication (integrity), entity authentication, and negrained authorization.

Security-related protocols and architectures were historically aimed for traditional networks -like the Internet- and devices. They were not conceived to work on constrained nodes or networks; these constraints include for example: nodes with limited RAM, ROM, CPU power and energy, high latency and unreliable network transmissions.

The aforementioned constraints add additional challenges when implementing or designing security protocols and architectures. Most of them simply cannot run on constrained environments. Hence new protocols and architectures need to be defined for achieving security goals on constrained nodes and networks. The aim of the IETF is to reuse to the greatest extent possible the already defined security protocols and architectures adapting them to the constrained world of the Internet Of Things (IoT).

The rest of this paper is structured as follows: Chapter 2 presents a chronological summary of the IETF IoT-oriented working groups and most important protocols. Chapter 3 presents the fundamental standards that are the base for the security-related protocols. Chapter 4 deals specifically with the IETF security-related protocols categorizing them according to the layer they apply: network, transport or application. Chapter 5 presents the Authentication and Authorization Framework for IoT: ACE, currently being defined at IETF. Finally, Chapter 6 offers a summary of the unsolved security problems that yet need to be addressed and future axes of research at IETF.

2 A brief history of the IETF IoT-oriented working groups

The first efforts of IETF protocols adapted for the Internet of Things (IoT) date back to 2005 with the 6LoWPAN (IPv6 over Low power WPAN) working group (WG) which provided the first adaptation of IPv6 for constrained-node networks on RFC4944: *Transmission of IPv6 Packets over IEEE 802.15.4 Networks*[1] published on 2007-09. The ROLL (Routing Over Low power and Lossy networks) WG addressed the routing problem on 6LoWPANs with the design and publication of RPL: *IPv6 Routing Protocol for Low-Power and Lossy Networks* (LLN)[2] published on 2012-03. These documents provided the foundations IoT world allowing IPv6 connectivity on Low-Power and Lossy Networks.

Since around the year 2010 with the start of the CoRE (Constrained RESTful Environments) WG we can observe a growing interest on the IETF about constrained environment related working groups and protocols. The CoRE WG deals with application-level goals and one of his most valuable outputs has been the *Constrained Application Protocol* (CoAP)[3] (RFC 7252). The 6lo (IPv6 over Networks of Resource-constrained Nodes) WG further continues the work of the 6LoWPAN WG and aims to facilitate IPv6 connectivity over constrained node networks with heterogeneous link layer technologies; similarly, 6TiSCH (IPv6 over the TSCH - Timeslotted Channel Hopping- mode of IEEE 802.15.4e) WG focus on enabling IPv6 over the TSCH mode of the IEEE 802.15.4e standard. LWIG (Light-Weight Implementation Guidance) WG publishes guidelines for implementation of all these protocols in constrained nodes and has standardized a common terminology to be used on future standards.

The IPv6 adaptation 6LoWPAN, the RPL routing protocol, and the CoAP application protocol have become the pillar protocols for connectivity and interoperability on the IoT. Other IoT-related protocols are The User Datagram Protocol (UDP) published on 1980 and the Datagram Transport Layer Security (DTLS) protocol published on 2006, even if they pre-date the IoT paradigm they are well suited to solve the IoT constraints. All this protocols constitute what has become the basic IoT stack. This IoT stack[1] side by side with the standard Internet protocols stack can be seen on Figure 1.

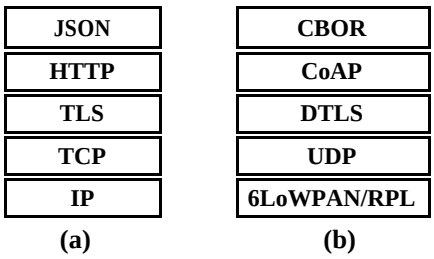


Fig. 1. IETF Protocols stack for: (a) Standard Internet; (b) Internet of Things;

[1] With the addition of the CBOR binary data format, that will be introduced further on this paper

Security-oriented WGs started to arise starting from the year 2013 with the **DICE (DTLS In Constrained Environments) WG** profiling DTLS for IoT and later with the **ACE WG** dealing with Authentication and Authorization on Constrained Environments. The 6lo (IPv6 over Networks of Resource-constrained Nodes) and 6TiSCH WGs also started to focus on security-related milestones. In general all the IoT groups started to focus on security once the deployment of the base IoT protocols started to become a reality but the basic security needs where not yet addressed.

3 Fundamentals

This chapter presents the fundamental standards that are the base for the security-related protocols that will be presented later.

3.1 Nomenclature

Terminology for Constrained-Node Networks[4] (RFC7228) defines a common terminology that has been used thoroughly on all the other IETF constrained environments new protocols. One important output is the distinction of three different Classes of Constrained Devices shown on Table 1.

Table 1. Classes of devices according to RFC7228 [4]

Device Class	RAM (KB)	Flash (KB)
Class 0	<< 10	<< 100
Class 1	10	100
Class 2	100	250

Class 0 devices are assumed to not have the resources required to communicate directly with the Internet in a secure manner. Class 2 devices have not difficulties implementing standard protocols. Class 1 devices are assumed on most new security solutions and protocol adaptations; in other words, current IETF for IoT security proposals are targeted to devices with 10 KB of RAM 10KB and 100 KB of Flash.

Having a standardized terminology helps to properly describe the different problematic on IoT in a consistent way and propose solutions accordingly.

3.2 CoAP Protocol and Security

The Constrained Application Protocol (CoAP)[3] (RFC 7252) is a specialized web transfer protocol for use with constrained nodes and constrained networks. One of its main goals is to provide a RESTful transfer service similar to HTTP, but simplified for the use on constrained devices on constrained networks. CoAP provides a request/response interaction model between application endpoints, supports built-in discovery of services and resources, and includes key concepts of the Web such as URIs and Internet media types. Even if CoAP has been designed to run over UDP, it can be adapted to run on top of any other datagram oriented protocol (e.g.: CoAP-over-SMS, CoAP-over-LoRaWAN, CoAP-over-Bluetooth Low Energy).

Security in CoAP. The standardized approach for securing CoAP (securing meaning confidentiality and integrity) is using the *Datagram Transport Layer Security (DTLS) Version 1.2* (RFC 6347). CoAP assumes that the cryptographic credentials are already previsioned, and defines four security modes:

1. **NoSec:** There is no protocol-level security (DTLS is disabled).
2. **PreSharedKey**: DTLS is enabled and pre-shared keys ciphersuites are used.

3. **RawPublicKey**: DTLS is enabled and the device has an asymmetric key pair without a certificate that is validated using an out-of-band mechanism.
4. **Certificate**: DTLS is enabled and the device has an asymmetric key pair with an X.509 certificate.

CoAP does not have primitives to assure neither authentication nor authorization; if these security services are required they will have to be provided either by communication security (i.e., IPsec or DTLS) or by object security (within the payload). The DICE (DTLS In Constrained Environments) and ACE (Authentication and Authorization for Constrained Environments) working groups were created to assess the remaining challenges and complete an appropriate framework needed for a secure IoT environment.

3.3 CBOR: Concise Binary Object Representation

The Concise Binary Object Representation (CBOR)[5] (RFC 7049) is a binary data format inspired by JSON and aimed at compact representation of most common data types used at Internet standards; it also has the explicit goals of a lightweight implementation in terms of code and ram needed, and no needed schema description to decode. It is normally used on top of CoAP to represent information.

4 IETF security at different layers

IETF has developed IoT security solutions at all the layers it is competent, this includes network-layer (Layer 3) and above. Security at link-layer (Layer 2) and below are not on scope of IETF. Different types of applications will need security at different layers. There is a rough consensus currently at IETF to go for application layer security (also called object security), on spite of the others: network-layer solution is seen as cumbersome, difficult to implement and not so lightweight; transport-layer provides some end-to-end (e2e) security problems at the presence of proxies; on the contrary application layer is seen as the most flexible solution and suitable for caching and proxying while maintaining e2e security properties. We will present security solutions at each layer.

4.1 Security at network-layer

Security at network-layer is the less active field at IETF. Security is based on adaptations of the IPsec protocol suite: IKEv2 and ESP. The LWIG working group is the home for these efforts. RFC 7815: Minimal Internet Key Exchange Version 2 (IKEv2) Initiator Implementation [6] is the most important document and presents a lightweight version of the IKEv2 protocol; this protocol is used for performing mutual authentication and establishing and maintaining security associations (fresh keys) on the IPsec suite. To encrypt the data a minimal version of the IP Encapsulation Security Payload (ESP) is being currently worked on the same working group; however the future of IPsec encryption for IoT is not clear, and seems to be relegated to higher layer security solutions.

4.2 Security at transport-layer

Transport-layer solution for IoT is provided by Datagram Transport Layer Security (DTLS). A special working group DTLS In Constrained Environments (dice) was created to define a DTLS profile suited for IoT. The output document is *Transport Layer Security (TLS) / Datagram Transport Layer Security (DTLS) Profiles for the Internet of Things* (RFC 7925 [7]). This document dictates how to set available configurable options and protocol extensions: e.g. setting timers values, choosing appropriate DTLS ciphersuites. The three types of credentials to use correspond which the mandate of CoAP: (1) for PSK the mandatory ciphersuite is

TLSPSK_WITH_AES_128_CCM_8; (2) for raw public key the ciphersuite is TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 which uses elliptic curve cryptography; (3) for certificates the mandatory ciphersuite is the same as raw public key, but the key will be wrapped on a X.509 v3 certificate. After the DTLS authentication credentials have been used on the DTLS handshake protocol all methods encrypt with the same symmetric algorithm AES-128-CCM, with an 8 byte tag/integrity value. Once we encrypt the DTLS per datagram overhead is of 13 bytes without counting the tag.

One of the advantages of transport-layer security is that can be used to transport any application data, including CoAP. Hence most IETF IoT protocols assumed at least a PSK between nodes, the establishment of a DTLS channel following the guidelines of DICE and then security was guaranteed. However, one of the drawbacks of this solution is that end-to-end security cannot be achieved in the presence of CoAP proxies (forwarding, caching), the DTLS secure channel will have to be terminated at the proxy, this led to the definition of higher layer security solutions.

4.3 Security at application-layer

Security at this layer, also referred as object security, has the advantage that can be maintained end-to-end and the security properties can be set on a per-message basis. The pillar of object security is CBOR Object Signing and Encryption (COSE) [8]. COSE describes how to create and process encryption, signatures and message authentication codes using CBOR for serialization.COSE will has more flexible security properties than DTLS but at the cost of more overhead. COSE is used to build upon other solutions like Object Security of CoAP (OSCOAP) [9]. OSCOAP provides end-to-end encryption, integrity, and replay protection of CoAP messages. It has two modes one only protecting payload, and other also protecting CoAP certain options and header fields. It also provides a secure binding between CoAP request and response messages, and freshness of requests and responses.

4.4 Security at layers summary

A summary of the mentioned security solutions contrasted with the IoT non-secured protocols and standard Internet protocols can be seen on Table 2.

Table 2. IETF security at different layers: Protocols

Layer	Std. Internet	Secure Std. Internet	IoT	Secure IoT
Application	HTTP	HTTPS	CoAP,CBOR	OSCOAP,COSE
Transport	TCP	TLS	UDP	DTLS
Network	IP	IKEv2+IPsec(ESP)	6LoWPAN	Min.IKEv2+ESP

5 An Authentication and Authorization Framework: OAuth 2.0 for IoT

The Authentication and Authorization for Constrained Environments (ACE) working group is the most active on developing a comprehensive security solution for the IoT. As his title depicts the main objectives are *Authentication and Authorization, encryption* will be leveraged on solutions previously mentioned like iot-profiled-DTLS or object security. The working group has two explicit goals: (1) Produce use cases and requirements and (2) Identify authentication and authorization mechanisms suitable for resource access in constrained environments. The first

goal has been addressed and the second goal is being addressed by the work-in-progress: *Authentication and Authorization for Constrained Environments (ACE)* [10]. This section will offer an overview of the current ACE solution. The building blocks of the solution are: CoAP, CBOR/COSE, the OAuth 2.0 framework and the proof-of-possession tokens.

5.1 Standard OAuth 2.0

OAuth 2.0 is an open standard that enables a resource owner to grant a third-party limited access to a protected resource; is designed to be used over HTTP and TLS. Access Tokens are authorization credentials that grant access to protected resources. The *Proof-of-Possession (PoP) Tokens* are access tokens bound to a specific cryptographic key. To access a protected resource not only the token must be possessed but also the possession of the associated key must be proven. One important security property of PoP tokens is that they can be **transported over unsecured channels**.

5.2 ACE's IoT OAuth 2.0

ACE's OAuth v2.0 for the IoT uses PoP access tokens; HTTP is replaced by CoAP and JSON by CBOR, CoAP runs over UDP instead of TCP, and hence DTLS should be used instead of TLS. Application-layer security solutions are also envisioned and are based on COSE (CBOR Encoded Message Syntax)[8] and OSCOAP (Object Security of CoAP).

OAuth Entities: The main entities involved in the OAuth framework are:

- *Resource Server (RS)*: An entity which hosts a protected resource.
- *Client (C):* An entity which attempts to access a protected resource on a RS.
- *Authorization Server (AS):* An entity that enforces the Resource Owner's policies, prepares and endorses authorization and authentication data.

OAuth Message Flows: The procedure that allows C to get access to a protected resource is the following:

1. C sends a Token Request message to the AS.
2. If C is authorized AS generates and sends to C the Access Token (opaque to C) and Client Information (e.g. contains the key bound to the PoP access token).
3. C sends the Access Token to RS and the protected resource Request
4. RS validates the request with the associated access token, if successful, responds with an (encrypted) representation of the protected resource.

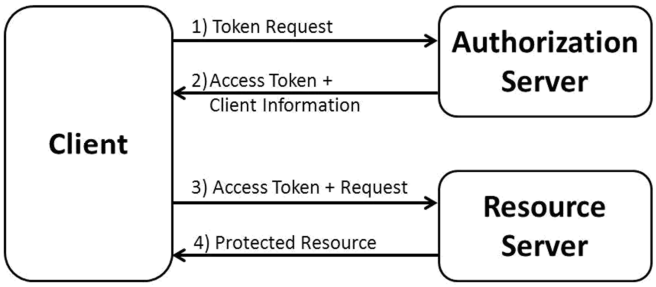


Fig.2 . OAuth 2.0 Basic Messages Flow and Entities

Figure 2 represents the basic OAuth 2.0 message flow and its main entities. The PoP Token offers the secure establishment of an authenticated key -with associated authorization permissions- between previously unknown parties and over an unsecured channel. This fresh key can be further used to establish authenticated and secured communications between these parties.

6 IETF perspectives, industrial applications and conclusion

IETF Perspectives. The IETF is providing tools that permit today to offer an IoT-suited security solution for most of the devices and use cases. There is still unsolved problems. Provisioning-bootstrapping of cryptographic material is also being studied at the Thing-to-Thing research group, all current solutions assume some pre-previsioning or a trusted third party. Cryptographic generated Identities, ID-based cryptography and new elliptic curves are also current topics studied at the crypto forum research group. Solutions for Low-Power Wide Area Networks with ultra-low speeds/bandwidth are needed and will be addressed on a future WG: *LPWAN* (*IPv6 over Low Power Wide-Area Networks*).
The ACE WG is defining a comprehensive security framework that aims at solving most of IoT use cases, but time awareness is required at the constrained node. New types of time-constrained terminology are needed and solutions that do not rely on time synchronization will need to be proposed.

Industrial. Industrial applications are using IETF's already well-established IoT protocols such as CoAP adapting it to their needs (e.g.: running on TCP). Most recent protocols (e.g: COSE) or in-progress ones (ACE Framework) will take time to be adopted on the industrial world due to the conservative nature of it. However, we believe that the time required for the IETF protocols to be adopted by the industry will likely decrease. We base our last assertion on the good interaction between IEEE (a body more associated with the industry) and IETF for example defining cross-layer solutions at the 6TiSCH working group; also the now forming LPWAN WG has some key technology players (SIGFOX, LoRA Alliance, Wi-SUN, NarrowBand-IOT) being involved, and ACE WG has some industrial referents such as Ericsson and ARM; meaning they will give valuable input to future protocols, making them more aligned with the industrial needs, and more likely to be quickly adopted.

Conclusion. Secure Identity representation, Bootstrapping, Privacy, and Multicast-security are some of the yet unsolved problems. IETF might never arrive to a one- fits-all solution, but the open nature of the protocols and layered design are enough tools to make security for IoT a reality.

Références

1. G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks," RFC 4944 (Proposed Standard), Internet Engineering Task Force, Sep. 2007, updated by RFCs 6282, 6775. [Online]. Available: http://www.ietf.org/rfc/rfc4944.txt

2. T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," RFC 6550 (Proposed Standard), Internet Engineering Task Force, Mar. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6550.txt

3. Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," RFC 7252 (Proposed Standard), Internet Engineering Task Force, Jun. 2014. [Online]. Available: http://www.ietf.org/rfc/rfc7252.txt

4. C. Bormann, M. Ersue, and A. Keranen, "Terminology for Constrained-Node Networks," RFC 7228 (Informational), Internet Engineering Task Force, May 2014. [Online]. Available: http://www.ietf.org/rfc/rfc7228.txt

5. C. Bormann and P. Hoffman, "Concise Binary Object Representation (CBOR)," RFC 7049 (Proposed Standard), Internet Engineering Task Force, Oct. 2013. [Online]. Available: http://www.ietf.org/rfc/rfc7049.txt

6. T. Kivinen, "Minimal Internet Key Exchange Version 2 (IKEv2) Initiator Implementation," RFC 7815 (Informational), Internet Engineering Task Force, Mar. 2016. [Online]. Available: http://www.ietf.org/rfc/rfc7815.txt

7. T. Fossati and H. Tschofenig, "Transport Layer Security (TLS) / Datagram Transport Layer Security (DTLS) Pro les for the Internet of Things," RFC 7925, Jul. 2016. [Online]. Available: https://rfc-editor.org/rfc/rfc7925.txt

8. J. Schaad, "CBOR Object Signing and Encryption (COSE)," Internet Engineering Task Force, Internet-Draft draft-ietf-cose-msg-18, Sep. 2016, work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-ietf-cose-msg-18

9. G. Selander, J. Mattsson, L. Seitz, and F. Palombini, "Object Security of CoAP (OSCOAP)," Internet Engineering Task Force, Internet-Draft draft-selander-ace-object-security-05, Jul. 2016, work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-selander-ace-object-security-05

10. E. Wahlstroem et al., "Authentication and Authorization for Constrained Environments (ACE)," IETF, Internet-Draft draft-ietf-ace-oauth-authz-02, Jun. 2016, work in Progress.

Sécurité LoRaWAN
(Catégorie : Spécialisé)

:

Franck L'HEREEC
Ingénieur sécurité Orange Labs
franck.lhereec@orange.com

Nicolas JOULAIN
Consultant sécurité Orange Cyberdefense
nicolas.joulain@orange.com

Abstract. LoRa (Long Range) est un réseau longue portée, bas débit et basse consommation dédié aux objets connectés. Notamment déployé par Orange, ce réseau s'appuie sur les spécifications LoRaWAN de la LoRa Alliance dont il est membre. Ce réseau privilégie les communications objet vers serveur, toutefois des échanges bidirectionnels sont possibles. Des considérations de sécurité ont été intégrées dès la première version des spécifications telles que la cryptographie AES 128. La présente soumission propose une introduction à la technologie LoRaWAN, une présentation des mécanismes de sécurité sous-jacents ainsi qu'un retour d'expérience des outils et méthodes utilisés lors de l'audit d'objets et d'équipements d'infrastructure impliquant des communications LoRa.

Keywords: Attaques matérielles, audit radio, AES, SDR, LPWAN

1 Introduction

Le phénomène de l'Internet des Objets a récemment vu émerger une multitude de technologies de connectivité sans fil qui lui sont dédiés. Ces technologies se différencient principalement par les critères de consommation énergétique, de couverture, de coût et de sécurité qu'elles proposent. Celles-ci peuvent être classées selon trois catégories majeures :

- Les technologies M2M basées sur des technologies cellulaires historiques telles que l'UMTS, L'EC-GSM, le LTE et ses dérivés;
- Les technologies locales basse consommation regroupées sous la norme IEEE 802.15 des « Wireless Personal Area Network » (WPAN). Les plus connues de ces technologies sont : Zigbee et 6LoWPAN (basées sur 802.15.4), Z-Wave, BLE (Bluetooth Smart), EnOcean, ULE (DECT) ainsi que différents protocoles propriétaires tels que X2D-X3D de Delta Dore ou Chacon DIO.
- Les technologies LPWA (Low-Power Wide-Area) qui proposent des communications sur de très longues portées avec de très faibles consommations énergétiques (Sigfox, LoRaWAN, Qowisio, Wheightless…).

Ce document se focalise sur les réseaux LoRaWAN dont les spécifications sont publiées par la LoRa Alliance. Basé sur un retour d'expérience terrain, le présent article introduit dans un premier temps les aspects technologiques des réseaux LoRaWAN. En seconde partie, le modèle de sécurité basé sur de la cryptographie AES128 est présenté. La troisième section aborde les principales techniques d'audit permettant

d'analyser la sécurité des composants impliqués dans une chaîne de communication LoRaWAN (l'objet, les passerelles et le cœur de réseau).

2 LoRaWAN : écosystème et aspects technologiques

2.1 La technologie LoRa comme couche physique

Avant d'introduire les concepts d'architecture et de sécurité des réseaux LoRaWAN, précisons tout d'abord que ceux-ci sont basés sur la technologie LoRa (Long Range). Il s'agit d'une couche physique radio à large bande permettant de créer des communications bidirectionnelles sans fil longue distance et à bas prix. A l'origine la technologie LoRa a été développée par Cycléo, une startup française créée en 2009 à Grenoble. La technologie LoRa est aujourd'hui propriété de Semtech, une société californienne de semi-conducteur qui a racheté Cycléo pour 5M$ en 2012. LoRa est actuellement distribué sous la forme de puces électroniques de la série SX127(X).

2.2 Les spécifications de la LoRa Alliance

- Basée sur la technologie LoRa, la première spécification LoRaWAN a été publiée en janvier 2015 par la LoRa Alliance. Cette spécification définit le protocole et l'architecture nécessaire à la mise en œuvre d'un LPWAN avec comme objectifs :
- l'optimisation de la consommation énergétique des objets ;
- la mise en œuvre de réseaux à très grande capacité (en termes de nombre d'objets connectés) ;
- la sécurité des communications ;
- la mise en œuvre d'infrastructures et d'objets à faibles coûts.

Cette spécification s'attache par ailleurs à garantir la conformité de la technologie aux contraintes régionales de l'usage des bandes radios. En effet, pour des raisons de coûts, les réseaux LoRaWAN exploitent les bandes de fréquences dites ISM. Il s'agit de bandes de fréquences non licenciées mais soumises à une réglementation en termes:

- de puissance d'émission ;
- de temps d'utilisation de la bande de fréquence, autrement appelé le « *duty cycle* ».

Note : La bande de fréquence ISM varie selon les continents et les législations locales (ex : 902 à 928 MHz pour l'Amérique du nord, 863 à 870 MHz pour l'Europe, 920 à 925 MHz pour le Japon …).

Actuellement en version 1.0.2, la version LoRaWAN 1.1 est prévue fin 2016. Notons que les spécifications sont accessibles gratuitement à tous. Chaque nouvelle révision n'est toutefois accessible pendant les 6 premiers mois qu'aux seuls membres. Début 2016, la LoRa Alliance dénombrait plus de 220 membres parmi lesquels nous retrouvons notamment:

- Des opérateurs, ex : KPN, Proximus, Bouygues Telecom et Orange.
- Des équipementiers, ex : Cisco, Kerlink, Sagemcom.
- Des intégrateurs, ex : IBM……

2.3 Caractéristiques fonctionnelles d'un réseau LoRaWAN

Portée, débit et consommation énergétique.
La spécification LoRaWAN permet de définir un réseau dédié à l'internet des objets qui regroupe l'ensemble des caractéristiques d'une technologie LPWAN :

- Des communications longue portée. La portée annoncée est de 15 Km en zone rurale et de 3 à 5 Km en zone urbaine dense. Les signaux radio LoRaWAN présentent par ailleurs un bon taux de pénétration en intérieur des bâtiments. Cette propriété permet notamment d'atteindre des objets en profondeur, dans des caves ou des sous-sols, inaccessibles aux technologies GSM historiques.
- Un faible débit : afin de répondre aux besoins spécifiques des IoT, les débits proposés sont faibles et varient de 300 bits/s à 250 Kbits/s en fonction des besoins et des interférences.
- Une faible consommation électrique : les cas d'usage IoT « *powerless* » sont les premiers visés par la technologie LoRaWAN. A titre d'exemple, la loRa Alliance annonce une durée de vie plus de huit ans pour une batterie de 2000 mAh équipant un IoT connecté LoRaWAN (un objet connecté en LTE-M affiche une durée de vie de batterie de 18 mois). Ces performances sont notamment atteintes grâce à une puissance d'émission radio très faible : entre 10 et 25 mW. En comparaison, une puce GPRS émet avec une puissance de 250 mW (x10)).

Le schéma suivant positionne différentes technologies en fonction de leur débit et de leur portée :

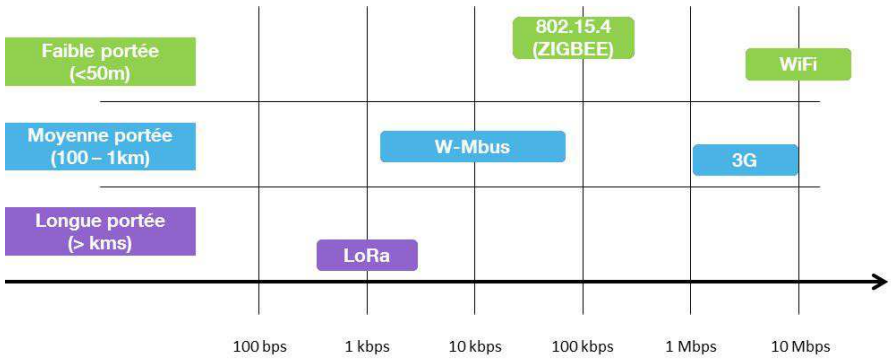


Fig. 1. Débits et portées de différentes technologies de communication

Communications bidirectionnelles : trois classes d'objets.

Au sein d'un réseau LoRaWAN, il est possible de mettre en œuvre des communications bidirectionnelles. Toutefois, les bandes d'émission et de réception étant proches, il n'est pas possible de générer des communications simultanées. Notons par ailleurs que dans un souci de gestion de l'énergie, les communications sont généralement initiées par les objets. A ce titre trois classes d'objets sont prévues pour répondre à des cas d'usages différents :

- Classe A, communications bidirectionnelles : chaque transmission montante (i.e. : de l'objet au serveur) est suivie par deux courtes fenêtres descendantes autorisant la réception de messages par l'objet. Les plages de liaison descendante interviennent juste après que l'objet ait transmis en liaison montante. Les besoins de communication depuis le serveur à d'autres moments doivent impérativement attendre la prochaine liaison montante planifiée par l'objet.
- Classe B, communications bidirectionnelles avec plages de réception: en complément des fonctionnalités de la classe A, ces objets ont des plages de réception supplémentaires planifiées par le serveur. Le serveur impose ainsi à l'objet des temps d'écoute.

- Classe C, communications bidirectionnelles avec plages de réception maximum: ces objets sont toujours en écoute, sauf lorsqu'ils transmettent. Ces objets consomment plus que les objets classe A et B, mais offrent une latence minimale pour la communication serveur vers objet.

2.4 L'architecture réseau

Comme le montre la documentation proposée par la LoRa Alliance, l'architecture réseau est la suivante :

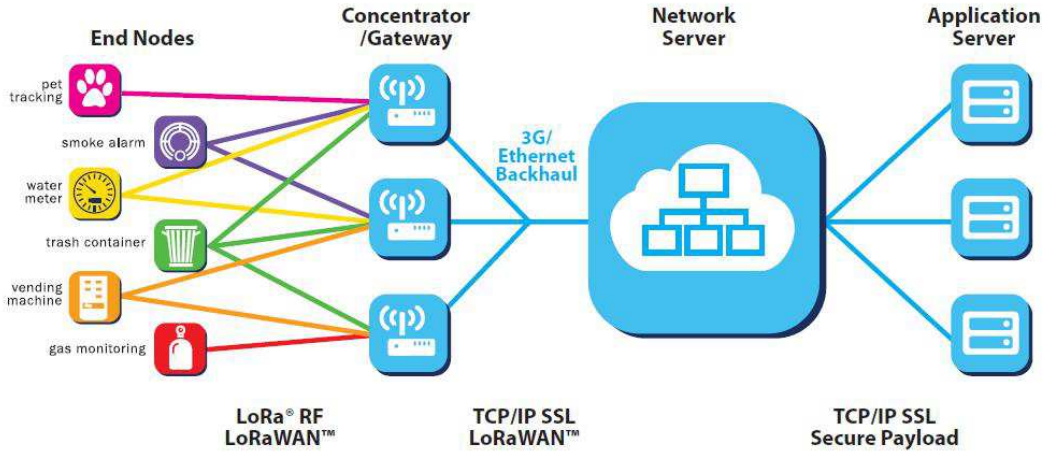


Fig. 2. Architecture LoRaWAN (source : www.lora-alliance.org)

L'architecture LoRa est constituée de quatre composants majeurs :

- **Les objets.** Ils sont généralement composés d'un ou plusieurs microcontrôleurs et d'une puce Semtech de la série SX127(X). Les microcontrôleurs gèrent la partie applicative, la liaison avec les capteurs/actionneurs et implémentent la partie LoRaWAN. La puce Semtech gère les aspects radio. Le schéma suivant introduit les différentes fonctions implémentées au sein d'un objet :

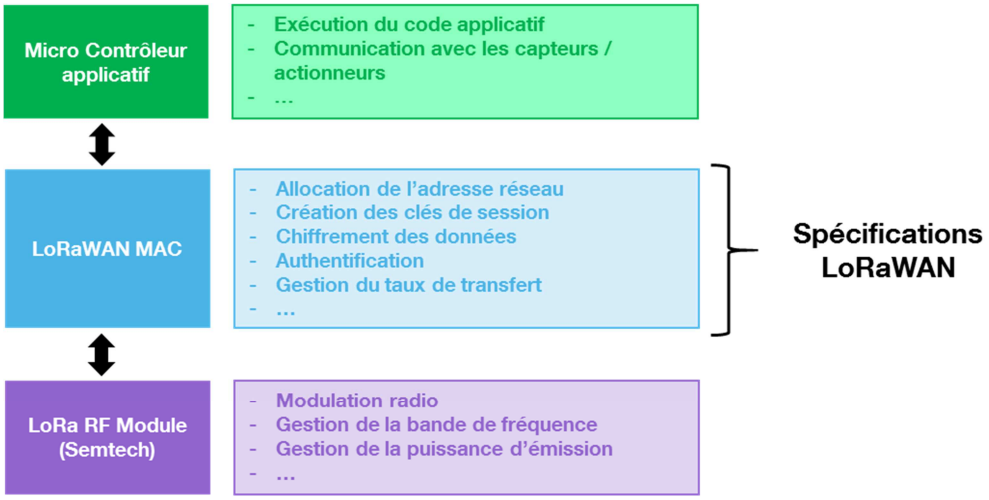


Fig. 3. Fonctions implémentées au sein de l'objet

- **Les passerelles.** Ces équipements transfèrent les messages radios issus des objets vers le serveur réseau au moyen d'un lien IP. La nature du lien IP permet de connecter les passerelles au serveur réseau par de multiples types de liens physiques (connexion 3G, lien (A)DSL, Fibre, Satellite …). Ce type de réseau se caractérise par une topologie en étoile où de multiples objets sont susceptibles de communiquer avec de multiples passerelles. A contrario d'un réseau GSM, il n'existe pas de phase d'enregistrement sur une cellule. Par ailleurs, précisons que la LoRa Alliance ne spécifie pas le protocole mis en œuvre entre les passerelles et le serveur réseau ;
- **Le serveur réseau.** Son rôle principal consiste à gérer la connexion radio de l'objet avec le cœur de réseau. Pour ce faire, des commandes MAC (Medium Access Control) sont échangées entre le serveur réseau et l'objet. Celles-ci permettent par exemple de définir le canal radio à utiliser ou encore, d'allouer des slots de réception. Le serveur réseau assure également une fonction de déduplication des messages potentiellement reçus par plusieurs passerelles. Le serveur réseau transfère ensuite le message au serveur applicatif ;
- **Le serveur applicatif.** Cette brique consomme le contenu du message envoyé par l'objet afin d'assurer la fonction métier recherchée par le service mis en œuvre.

3 Modèle de sécurité

3.1 Une sécurité basée sur AES128

Trois clés de sécurité.

Le modèle de sécurité de LoRa est basé sur un ensemble de trois clés. Le schéma suivant illustre l'usage de la NwkSkey et de l'AppSkey (note : l'AppKey est utilisée lors d'une phase d'enregistrement amont) :

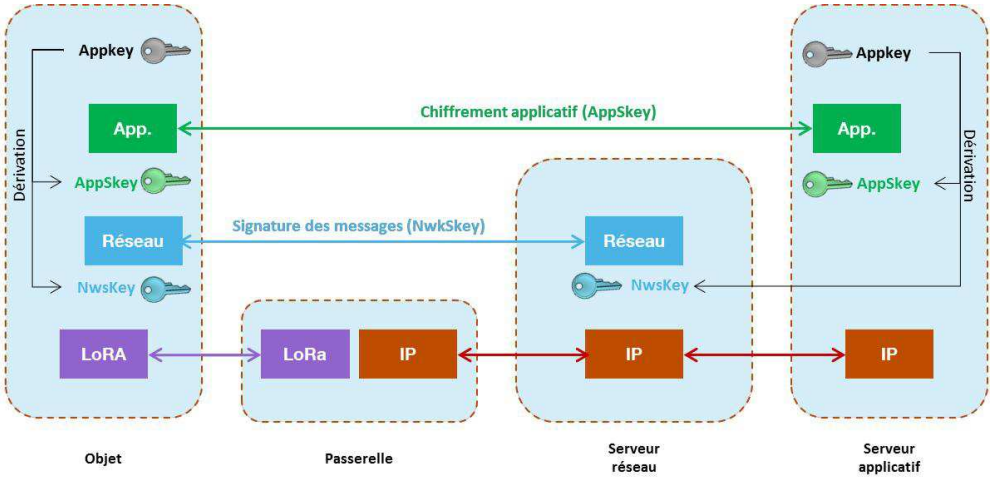


Fig. 4. : Sécurité des messages LoRaWAN

Le rôle de chacune des clés est le suivant :

- **La NwkSkey** est une clé de session réseau spécifique à chaque objet. Celle-ci est utilisée par l'objet et le serveur réseau pour signer et vérifier les messages. La signature permet ainsi de vérifier l'intégrité des messages et d'authentifier l'objet émetteur. Pour cela un champ MIC (*message integrity code*) est utilisé au sein des en-têtes de message. En complément un champ compteur est également utilisé pour éviter les attaques de type rejeu (le compteur est intégré à la partie du message signée par le champ MIC). La clé NwkSkey permet également de chiffrer et déchiffrer le contenu des messages d'administration du réseau, également appelés messages « MAC ». Note : la clé NwkSkey ne permet pas de déchiffrer le contenu des messages applicatifs.
- **L'AppSKey** est une clé de session applicative spécifique à chaque objet. Celle-ci est utilisée par l'objet et le serveur applicatif pour chiffrer et déchiffrer le contenu applicatif des messages. Cette clé n'est pas connue du serveur réseau qui ainsi ne peut accéder au contenu du message.
- **L'AppKey** est une clé statique AES de 128 bits assignée de façon unique à chaque objet par le propriétaire de l'application. Cette clé est nécessaire seulement lorsque le mode de d'activation retenu est le mode « *over the air* » (cf. 3.1.2). Dans ce cas, l'AppKey est alors dérivée pour générer deux clés de session : la NwkSKey (clé de session réseau) et l'AppSKey (clé de session applicative).

Remarques sur le modèle de sécurité LoRaWAN :

- Par défaut, les données applicatives ne sont pas protégées en intégrité de bout en bout du terminal au serveur applicatif (par les mécanismes LoRaWAN). Des mécanismes d'intégrité complémentaires sont susceptibles d'être ajoutés au niveau applicatif;
- La notion de session est peu précise dans la spécification. Dans la version LoRaWAN 1.0 : le renouvellement des sessions est exclusivement à l'initiative de l'objet, le réseau ne peut pas forcer le renouvellement d'une session. En pratique nous observons que les objets renouvellent peu les clés de session, notamment pour limiter la consommation d'énergie.

Activation des objets et initialisation des clés cryptographiques de session.

Pour être actif au sein d'un réseau LoRaWAN, les objets doivent être activés. Pour ce faire, deux méthodes ont été définies : le mode « *Over The Air* » et le mode « *Personnalisation* ».

Le mode Over The Air.

Dans ce mode, l'objet doit suivre une procédure d'accès avant de pouvoir échanger des données utiles avec le cœur de réseau. Cette procédure repose sur trois principaux paramètres configurés par le fabricant au niveau de l'objet :

- Le DevEUI, l'identifiant unique de l'objet.
- L'AppEUI, l'identifiant applicatif de l'objet.
- Et l'AppKey, la clé statique AES de 128 bits assignée de façon unique à chaque objet.

Cette procédure repose sur deux messages MAC successifs:

- **Le message Join-Request**. Ce premier message est émis par l'objet vers le cœur de réseau. Il contient notamment l'AppEUI et le DevEUI suivi de deux octets (DevNonce) pseudo-aléatoires. Ce message comporte un champ MIC (tag d'authentification calculé à partir de l'AppKey) qui permettra au serveur réseau d'authentifier l'objet émetteur du message Join-Request. Notons que ce message est transmis en clair et n'est pas chiffré.
- **Le message Join-Accept**. Ce second message est émis par le serveur réseau pour confirmer à l'objet qu'il est autorisé à accéder au réseau. Ce message contient un « AppNonce » qui est une valeur pseudo-aléatoire permettant à l'objet de dériver les clés de session NwkSKey et AppSKey. Afin de garantir la sécurité de l'aléa, un MIC est également utilisé. Ce message est chiffré avec l'AppKey de l'objet (également connu par le cœur de réseau au niveau du serveur applicatif).

Note : ce mode est le choix retenu par l'implémentation LoRaWAN d'Orange.

Le mode personnalisation.

Dans ce mode, l'objet peut communiquer avec le réseau dès sa sortie de l'usine. Pour ce faire, le DevAddr et les deux clés NwkSKey et AppSKey sont directement stockés dans l'objet lors de sa fabrication.

D'un point de vue sécurité, ce mode est plus limité car les clés de session ne peuvent pas être renouvelées.

Notons que le mécanisme OTA permet à l'opérateur LoRaWAN d'être partie prenante dans la génération des clés de session puisqu'il va générer l'AppNonce nécessaire à la dérivation des clés. Cet AppNonce doit être différent à chaque nouvelle procédure de Join ce qui permet de renouveler les clés.

3.2 Points d'attention

Comme tout système basé sur de la cryptographie, quelques précautions sont nécessaires pour garantir la sécurité des communications LoRaWAN :

- **Gestion des clés :** la sécurité LoRaWAN repose avant tout sur la confidentialité des clés de sécurité NwkSKey, AppSKey et AppKey. En fonction des implémentations, de multiples parties prenantes ont, à un moment, accès à ces clés : le fabricant de l'objet, le responsable de l'application, l'opérateur réseau… De fait et pour assurer la confidentialité des clés, il convient de réduire au maximum le nombre de personnes ayant accès aux clés et de s'assurer que chaque partie prenante manipule les clés de sécurité avec le plus grand soin.

- **Génération des clés** : la spécification LoRaWAN 1.0 alerte sur le processus de génération des clés. En effet, il est explicitement décrit que les clés doivent être uniques et que la compromission des clés d'un objet ne doit pas remettre en cause la sécurité d'autres objets. Toutefois, pour simplifier le processus de génération des clés, certains acteurs pourraient être tentés de dériver une clé racine avec une information non confidentielle (ex : l'identifiant de l'objet). Ce genre de pratique introduit des faiblesses et est susceptible d'impacter la sécurité de l'ensemble d'une flotte d'objets dont les clés de sessions sont dérivées depuis la même clé racine. En conséquence, il convient de prendre toutes les précautions nécessaires lors de la conception du processus de génération des clés (non défini par la spécification LoRaWAN).
- **Protection des clés sur l'objet :** le modèle de sécurité de LoRaWAN implique le stockage des clés de sécurité au sein de l'objet. Par nature, ces objets sont géographiquement dispersés et il n'est pas possible d'assurer une sécurité physique forte comme c'est le cas pour un actif IT hébergé au sein d'un centre d'hébergement de données. Les clés stockées sur les objets sont donc exposées à des attaques physiques (cf. 1.1). En conséquence, l'extraction de clés pourrait permettre à un attaquant de déchiffrer les communications de l'objet et d'usurper son identité. Rappelons toutefois que si le caractère unique et non prédictible des clés est respecté, l'impact d'une attaque physique sur un objet reste circonscrit à l'objet en question et ne permet pas de compromettre l'ensemble d'une flotte. De fait, l'impact est limité mais pour les cas d'usages les plus sensibles, il est recommandé de se prémunir de ces attaques physiques par l'usage de mesures complémentaires (ex : Secure Element).

En complément des points d'attention portant sur la cryptographie, il est nécessaire de prendre en considération la relative jeunesse de la technologie. En effet, de nombreux acteurs se sont lancés dans le développement et la mise en œuvre d'infrastructures LoRaWAN. Qu'il s'agisse d'implémentations open source ou de produits commerciaux, nombreuses sont les infrastructures actuelles qui ont été mises en œuvre pour des besoins de tests, sans que la sécurité n'ait été correctement adressée. En conséquence, et pour des services en production, une attention particulière est nécessaire pour s'assurer de la sécurité des composants ou services proposés.

4 Retour d'expériences sur les méthodes et outils d'audit spécifiques

En tant qu'opérateur de réseau LoRaWAN, Orange a comme responsabilité d'assurer et de maîtriser la sécurité des équipements LoRa déployés sur son réseau. Pour ce faire une de nos activités porte sur les audits de sécurité que nous réalisons à la fois au niveau de la phase de qualification après « *sourcing* » de l'équipement ainsi que sur les sites de production. Dans ce chapitre nous évoquerons les points suivants :

- présentation de ce que sont les attaques matérielles et du retour d'expérience que nous avons sur cette thématique concernant les objets et passerelles audités ;
- les attaques radio mises en œuvre.

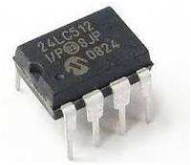
4.1 Attaques matérielles

Introduction.
Les objets IoT LoRaWAN sont des systèmes embarqués assez classiques dans lesquels nous retrouvons des microcontrôleurs, et parfois des mémoires telles que NAND FLASH, EEPROM, SDRAM. Dans cet environnement embarqué notre objectif est de vérifier la bonne implémentation de la sécurité. Dans le cadre de LoRaWAN nous sommes particulièrement vigilants à ce que les clés (AppKey, NwkSKey et AppSkey) soient correctement protégées et non accessibles à tout attaquant ayant un accès physique à l'objet.

La démarche d'audit matériel reste assez commune à une démarche d'audit classique, à savoir :

- Réaliser une reconnaissance de l'environnement embarqué. Souvent une inspection visuelle permet d'identifier les mémoires, le microcontrôleur, les bus de données, le composant radio, les connecteurs…
- Analyser le matériel par « scan ». Pour ce faire, nous nous appuyons sur la documentation des composants et nous utilisons un multimètre pour identifier des points d'entrée qui peuvent être intéressants pour tenter de réaliser une intrusion sur l'objet (brochage d'une UART ou d'un port JTAG,…).
- Rechercher les vulnérabilités connues et exploitables sur les composants identifiés au préalable.
- Exploiter les vulnérabilités dans le but d'accéder aux éléments logiciels de l'objet (exporter le contenu d'une mémoire, se connecter à un port série, extraire un *firmware*…).
- Réaliser des attaques d'usurpation (si le vol de clés est possible), ou des attaques par déni de service par exemple.

Voici le type de composants identifiés sur des objets LoRaWAN :

Type de composant	Rôle	Vulnérabilités recherchées par un attaquant
Microcontrôleur 	-Permet d'exécuter un firmware. - Contient ses propres mémoires embarquées (FLASH, EEPROM, SRAM). - Manipule des données en temps réel. - Communique avec d'autres mémoires ou d'autres composants de la carte électronique.	- Voir si le JTAG est disponible. - Voir si un port UART est actif. - Extraire le firmware. - Regarder les registres. - Extraire des clés LoRaWAN contenues dans la mémoire interne. - Faire du reverse engineering.
Mémoires (SPI ou I2C 	- Stocke des informations potentiellement sensibles : firmware, données sensibles (clés LoRaWAN), informations utiles lors de l'étape de collecte d'informations (logs,…).	- Extraire des clés LoRa contenues dans la mémoire externe. - Extraire le firmware
Composant radio	- Contient tout ce qui est nécessaire à la modulation/démodulation radio LoRaWAN.	- Prise de contrôle du Composant radio pour récupérer les trames envoyées et reçues par l'objet.

Quelques notions sont à connaître pour bien comprendre les attaques matérielles (notamment pour les microcontrôleurs). L'attaquant va chercher à identifier deux types d'éléments :

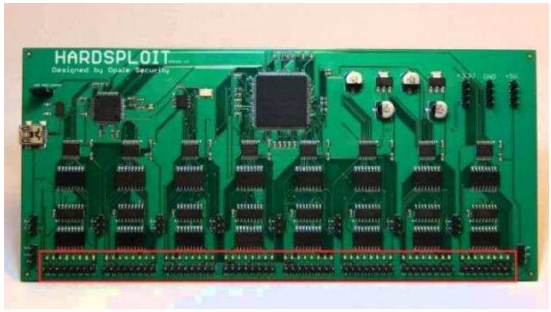
- Le bus JTAG (*Joint Test Action Group*) permettant de faire des tests, inspecter les registres, poser des points d'arrêt, réaliser la mise à jour d'un *firmware*. Il est composé de cinq signaux (ou deux pour certaines architectures).
- L'UART (*Universal Asynchronous Receiver Transmitter*) qui est un émetteur-récepteur asynchrone universel. Il est notamment utilisé pour mettre en place

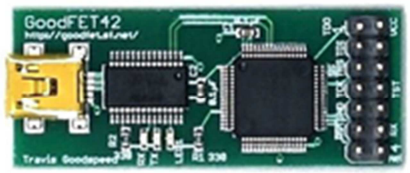
une liaison série avec un ordinateur. Sur cette liaison transitent parfois des informations précieuses pouvant conduire à la compromission de l'équipement (logs, menus, shells...).

Présentation de notre banc de test.
Afin de pouvoir identifier certaines failles de sécurité liées à une mauvaise conception de sécurité matérielle, nous avons besoin d'outils parce que :

- certains boîtiers de composants électroniques sont de type BGA (Ball Grid Array), les pins du composant ne sont pas facilement ou pas du tout accessibles ;
- la phase d'identification visuelle des pins du JTAG/UART n'est pas toujours évidente ;
- nous avons besoin de trouver la vitesse ainsi que la configuration adéquate pour se connecter à un bus de données

Pour cela nous utilisons les outils suivants

L'outil	Rôle
Hardsploit https://hardsploit.io/ 	Permet de lire/écrire dans certaines mémoires. Extraire un *firmware*. Interagir avec l'UART.
STLink : http://www.st.com/en/development-tools/st-link-v2.html?icmp=tt3469_gl_qron_apr2016 	Permet de prendre le contrôle d'un *firmware* de microcontrôleurs ST tels que le STM32 via le JTAG.

JTAGulator http://www.grandideastudio.com/jtagulator/ 	Permet de détecter le brochage d'un JTAG ou d'une UART
GoodFET http://goodfet.sourceforge.net/ 	Permet de faire du JTAG sur certains microcontrôleurs (MSP 430), extraire des données de mémoires.

Retour sur des failles identifiées d'un objet ou d'une passerelle LoRaWAN**.**

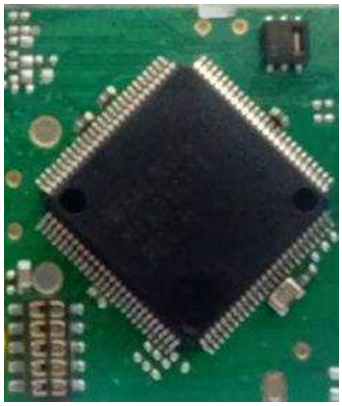
Nous utilisons la même méthodologie pour identifier des failles sur des passerelles et des objets LoRaWAN. Voici une synthèse de ce que l'on regarde lors des audits menés sur ces équipements :

- port série disponible et non désactivé ;
- port USB de debug accessible ;
- JTAG présent sur certaines cartes électroniques ;
- bootloader non protégé donc possibilité de flasher une passerelle avec un nouveau *firmware* ;
- extraction de firmware avec récupération de certaines données sensibles telles que des clés SSH

Ci-dessous un cas d'étude sur un objet LoRaWAN.

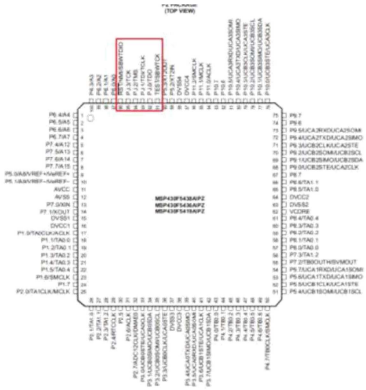
Après ouverture de l'objet, voici les étapes permettant d'identifier les failles sur celui-ci :

Etape 1 : identification des composants électroniques + connecteurs qui nous semblent importants



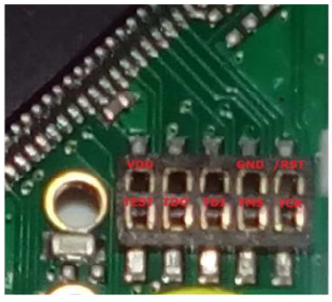
- Présence d'un microcontrôleur de type MSP430
- Présence d'un connecteur près du microcontrôleur

Etape 2 : une fois le microcontrôleur identifié, analyse de sa documentation technique pour déterminer la localisation des pins du JTAG.



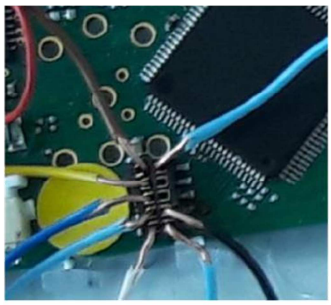
- JTAG présent sur les pins 91 à 96 du microcontrôleur

Etape 3 : identifier si les pins du JTAG sont reliés au connecteur.



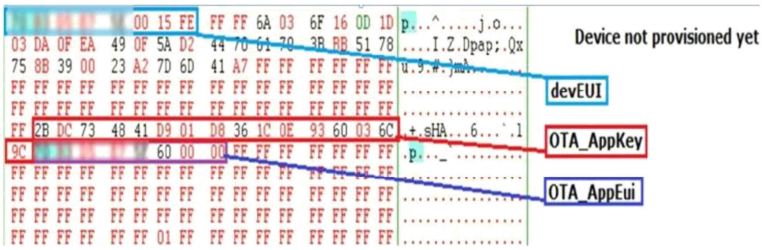
- Utilisation d'un multimètre pour déterminer le brochage du connecteur

Etape 4 : exploitation de la prise JTAG

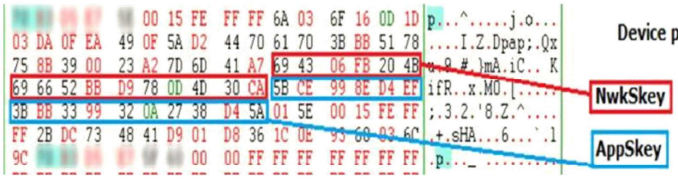


- Utilisation de GoodFET qui supporte le JTAG des microcontrôleurs MSP430
- Extraction du *firmware*

Etape 5 : recherche de la présence de clés dans la mémoire du microcontrôleur.



- Avant son attachement au réseau LoRaWAN on peut retrouver la clé racine (AppKey) utilisée pour la dérivation des clés de sessions.



- Après son attachement au réseau LoRaWAN, on peut retrouver les clés de sessions AppSkey et NwkSkey stockées dans l'objet.

Ayant connaissance de ces informations, l'attaquant a tout le matériel cryptographique nécessaire pour fabriquer et envoyer des trames LoRaWAN en se faisant passer pour un objet légitime. L'attaquant a également la possibilité de déchiffrer les trames LoRaWAN entre cet objet et le réseau LoRaWAN.

4.2 Attaques radio

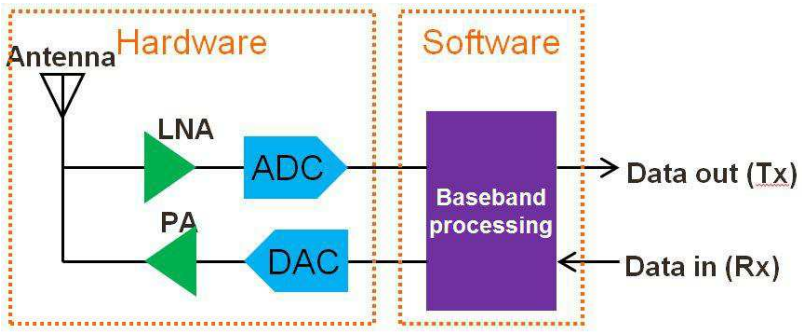
Introduction.

Nous développons des outils sur la radio, dans le but de qualifier nos passerelles, nos objets et notre infrastructure cœur face à des attaques radios potentielles.

LoRaWAN et le SDR

Dans le domaine public, il existe plusieurs publications intéressantes à lire pour ceux qui s'intéressent à la rétro-conception de la technologie radio LoRaWAN. A noter que la technologie radio Semtech est brevetée (n° 13154071.8).

Le SDR (*Software Defined Radio*) est un émetteur/récepteur radio dont les traitements sont en grande partie réalisés par du logiciel. On peut schématiser le SDR de la façon suivante :



La partie hardware est composée des éléments suivants :

- LNA : amplificateur à faible bruit.
- AP : amplificateur de puissance.
- CAN : convertisseur analogique numérique.
- CNA : convertisseur numérique analogique.

La partie traitement du signal est ensuite réalisée au niveau logiciel. Un des outils les plus utilisés est gnu-radio.
A noter : quelques liens intéressants sur la rétro-conception de LoRaWAN :

- Une présentation de Matt Knight, ingénieur sécurité de la société Bastille Networks dans laquelle il présente ses travaux sur l'analyse de LoRaWAN. (http://static1.squarespace.com/static/54cecce7e4b054df1848b5f9/t/57489e6e07eaa0105215dc6c/1464376943218/Reversing-Lora-Knight.pdf)
- Josh Blum a développé un outil Pothos permettant de décoder et encoder du LoRaWAN couplé à une SDR (LimeSDR). (https://myriadrf.org/blog/lora-modem-limesdr/)
- Le logiciel open source GNU radio propose maintenant des blocs pour analyser des communications LoRaWAN. Le projet en charge du développement de ces blocs est gr-lora (https://github.com/rpp0/gr-lora).

Les SDRs utilisent des composants matériels assez génériques ce qui permet de réaliser le traitement de signal à faible coût.

Notre passerelle d'audit

Pour nos besoins d'audit, nous avons réalisé notre propre passerelle.

Les développements ont été réalisés en C. Elle nous permet à la fois d'avoir un comportement que pourrait avoir un objet LoRaWAN et à la fois un comportement que pourrait avoir une passerelle LoRaWAN. Les paragraphes suivants détaillent les attaques développées sur cette passerelle.

Ecoute passive.

L'idée est de pouvoir écouter toute trame LoRaWAN sans avoir d'interaction avec l'objet ni avec le réseau LoRaWAN. Pendant cette écoute passive nous nous sommes focalisés sur les messages Join-Request émis par les objets qui sont à portée de la passerelle. En analysant ces messages, nous avons eu accès à l'AppEUI et

au DevEUI ce qui permet de cartographier les types d'objets déployés autour de la passerelle.
Cette première étape nous a également permis de stocker des messages Join-Request afin de mettre en œuvre des attaques complémentaires ultérieurement (cf. section *Attaque par rejeu*.).

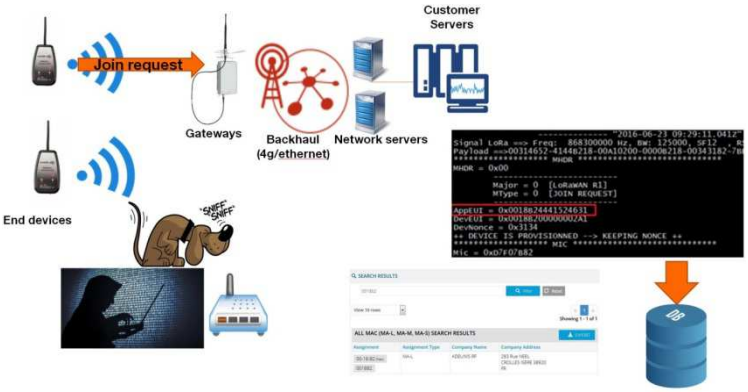


Fig. 5. Ecoute passive

Attaque par rejeu.

Le but ici est de rejouer un message Join-Request précédemment collecté lors de la phase d'écoute passive. On cherche à regarder le comportement du serveur réseau face à cette attaque. Il faut que le serveur réseau puisse détecter ce type d'attaque pour éviter que toute nouvelle dérivation de clés s'opère sur l'infrastructure rendant l'objet légitime isolé du réseau puisque ses clés de sessions ne sont plus synchronisées avec le serveur réseau.

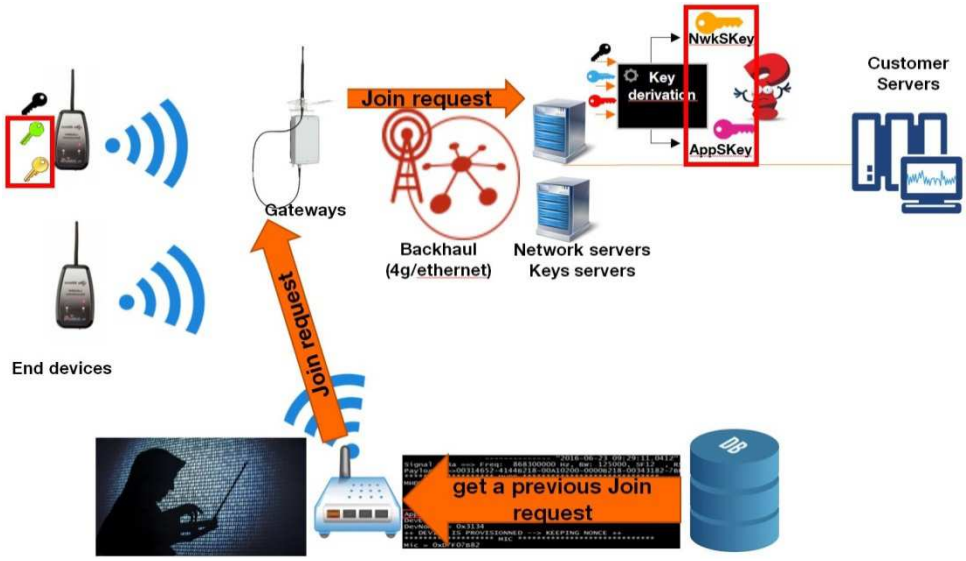


Fig. 6. Attaque par rejeu

Attaque par envoi massif de trames LoRaWAN.

Le but ici est de générer un volume de trafic LoRaWAN conséquent pour étudier le comportement des passerelles et du serveur réseau face à ce type d'attaques.

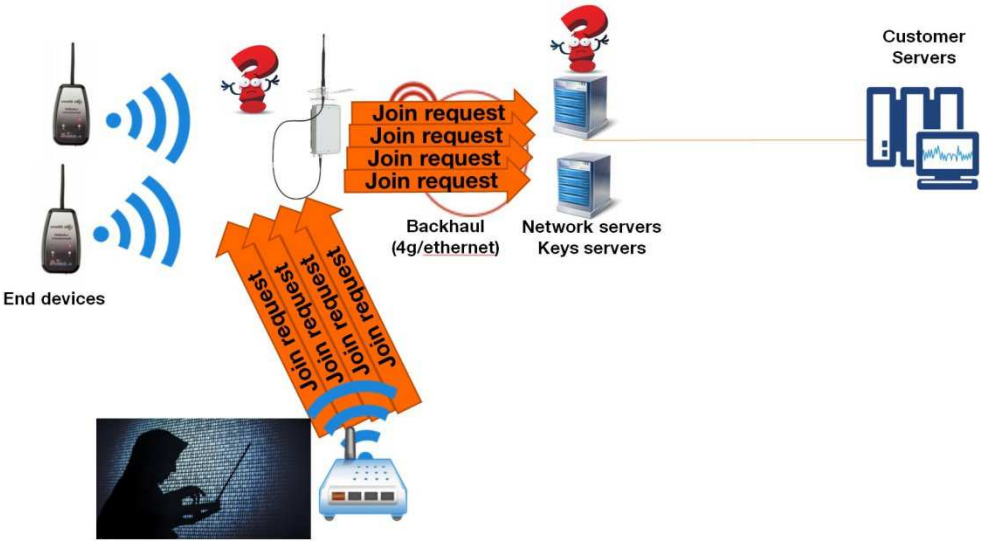


Fig. 7. Attaque de type flooding

Déni de service sur un objet

Le but ici est de renvoyer une trame LoRaWAN à l'objet pendant sa fenêtre d'écoute lorsque celui s'attache au réseau LoRaWAN. La passerelle identifie un message Join-Request et renvoie un « faux » message Join-Accept avant le réseau légitime. On cherche ici à identifier le comportement de l'objet face à ce type d'attaque.

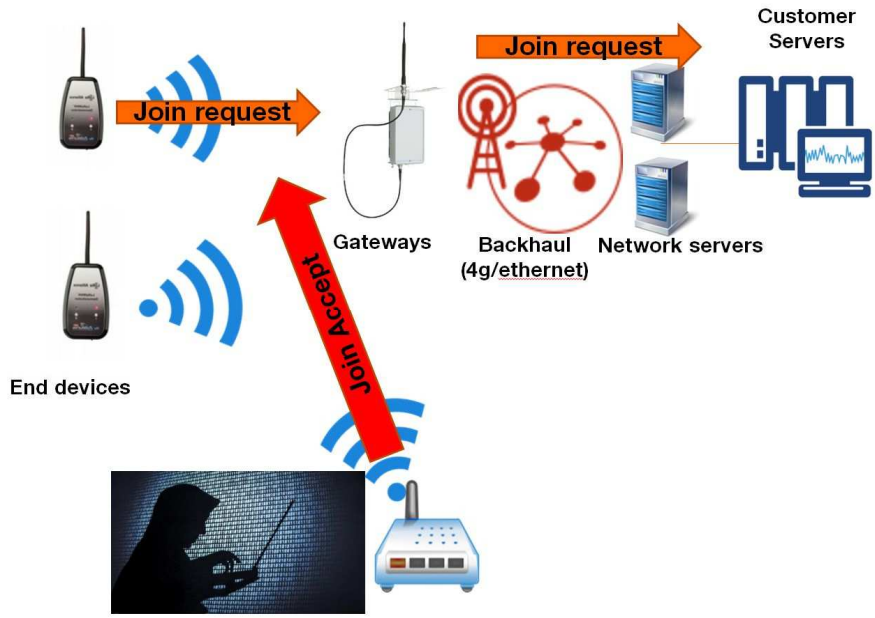


Fig. 8. Attaque de déni de service sur un objet

5 Conclusions

Malgré l'objectif de sécurité partagé dès l'origine par les membres de la LoRa Alliance, le présent article met en lumière les points d'attention à considérer lors de la mise en œuvre d'un service IoT sur LoRaWAN ainsi que quelques imperfections de sécurité propres à la spécification LoRaWAN 1.0. Ces faiblesses sont connues et doivent être considérées selon deux échelles de temps :

- Concernant les déploiements actuels, et pour les services nécessitant un haut niveau de sécurité, des contre-mesures complémentaires existent et doivent être adoptées afin de garantir un niveau de sécurité optimal ;

- Concernant les déploiements futurs, des corrections seront apportées lors de la prochaine version de LoRaWAN. A ce titre et en tant que membre de cette de la LoRa Alliance depuis mai 2016, Orange participe au « *Technical Working Group* » afin de suivre les évolutions de la spécification LoRaWAN mais aussi pour proposer des recommandations basées sur son expérience d'opérateur de réseau.

En ce qui concerne les déploiements Orange, une étroite collaboration a été mise en place entre les équipes sécurité, les équipes chargées de « *sourcer* » les équipements (objets, passerelles) et les équipes projets. Cette collaboration se traduit par :

- La préconisation d'exigences de sécurité en phase amont (approche « *secure by design* ») ;
- La contractualisation d'exigences de sécurité avec les fournisseurs d'objets ;
- Des phases de validation et d'audit de sécurité sur les infrastructures et objets déployés ;
- Un accompagnement sécurité des clients lors de la conception et mise en œuvre de leurs services IoT.

Formally Proven and Certified Off-The-Shelf Software Components: the Critical Links for Securing the Internet of Things

Dominique Bolignano

Prove & Run

77, avenue Niel, 75017 Paris, FRANCE

dominique.bolignano@provenrun.com

Abstract. The Internet of Things is bringing new security challenges that need to be addressed before deployment reaches a larger scale. We believe this can be done using a few key security software components and will illustrate this using a few security use cases that are representative of various Internet of Things market segments**.**

1 Introduction

The Internet of Things (IoT) is gaining traction and will probably accelerate its deployment in the future. Cybersecurity (referred to as "security" in this paper) issues have begun to appear in several areas of the IoT (connected cars [6,7], [10,11,12,13,14,15], smart grids, smart homes [16,17,18], smart offices, smart cities, avionics, adaptive maintenance, Industry 4.0, etc.). Such security problems will tend to grow at an even higher rate than the IoT itself as hackers will also benefit from more profitable business models coming from higher volumes, higher value of individual objects involved, more IoT features, etc. The problems are also very likely to spread across other areas of the IoT (home health care, medical equipment, on-demand manufacturing, etc.) until quickly becoming essential and critical to the successful deployment of the IoT.

In this paper we will first describe and justify the approach we propose in order to achieve a proper level of security. We will in particular demonstrate that the main sources of security issues can be attributed to faulty software where errors in the software architecture, design, implementation or configuration of an IoT system create vulnerabilities that can be exploited to mount a successful attack. The challenge is therefore to produce software that is as close as possible to "zero-bug": this paper will explain how this challenge can be met in a cost-effective way. We will then illustrate the critical part of the approach on a few representative use cases.

2 An Approach to IoT Security

A large number of cyber-attacks have been reported lately. Even if these attacks were mostly targeted to cars and to mobile phones we believe that most of them (i.e. [3,4,5,6,7], [10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26]) are quite general and could happen in virtually any area of the IoT. Whereas the potential for such attacks is still extremely important we believe that such remote attacks on IoT devices could be avoided by following a proper security methodology and using technologies readily available on the market. We will in the sequel present the proposed approach and illustrate it on known attacks.

2.1 Following a Proper Security Methodology

It is first important to follow a proper security methodology. This was obviously not the case for many of the systems affected by reported attacks, probably because the security issues were not taken seriously enough. Many acceptable security methodologies exist and we do not recommend here a particular one. Yet, a proper security methodology should at least involve a proper risk analysis including the identification of assets and a study of the risks inherent to the architecture and associated functionalities for the expected usage environment. Such a risk analysis should lead to the definition of primary and secondary assets with associated security properties (integrity, confidentiality, availability, etc.). It should also lead to the identification of attackers' profile, of threats, of assumptions on the usage environment, of organizational and technical security objectives for the intended usage, etc. It should lead to the definition of a targeted resistance level that is commensurate with the risks at stake and to the potential business models for attackers.

Such a risk analysis should typically be followed by the definition of (product) security requirements.

When such a proper security methodology is followed both the security architect and the development team have a clear framework with proper objectives and requirements to perform and guide their work as well as a way to assess the adequacy of their solution to the security context. In the case of IoT where (massive) remote attacks are the most critical ones following such a methodology almost always leads to distinguishing two phases in an attack.

Identification versus Exploitation Phase of the Attack. There are indeed in many cases two phases to consider for an attack: (1) the identification phase where the attack is identified and prepared and (2) the exploitation phase which corresponds to the use of the analysis, data, technique and tools defined during the first phase. The investment that can be reasonably made by attackers in the first phase is much higher, much more sophisticated tools, than the investment that can be reasonably applied to each single device in the second phase.

Addressing the Identification Phase of the Attack. When addressing the identification phase of the attack, security architects and developers will have to consider potentially sophisticated physical attacks. But technologies and know-how to resist to such attacks are widely available and in the end the only issue is cost. So the main strategy will be to keep to the very minimum the number of assets that need to be protected against sophisticated and expensive hardware attacks, and/or to minimize their value for attackers. Secure elements, HSMs and cryptographic processors can be used for that purpose, provided that the list of assets has been brought down to a few (root) secrets and keys that only need to be stored, and used to perform cryptographic operations. Even more important is to build a security architecture that reduces the value of assets by ensuring in particular that if such assets are compromised during the identification phase on one device, they cannot become key in performing the exploitation attack on a large number of remote devices.

Addressing the Exploitation Phase of the Attack. The importance of logical attacks. Once the problem of resistance with respect to the identification phase has been properly handled, physical attacks don't have to be considered anymore, besides side-channel attacks (mainly pertaining to cryptographic operations). These latter can easily be addressed by using state-of-the-art know-how and technologies that are readily available and that can be kept quite local. Now at the end the protection against logical attacks becomes the main challenge, and this is quite new.

Resisting logical attacks has indeed been until recently the easy and marginal part of the security challenge. This is because the hardware components to secure were both very small and had very small attack surfaces. The smartcard is a very good and representative example of such situations: the hardware features are simple (very few peripherals and interruptions to handle, simple cache mechanisms, etc.) so as a result the corresponding OS is very simple, which in turn exposes a very small attack surface compared to modern operating systems.
But this situation is radically changing, mainly due to the much larger attack surface exposed by IoT devices. An analysis of the attacks reported in conferences such as Black Hat 2013 to 2016 [3,4,5] indeed shows the ever-increasing importance of logical attacks. Hackers typically exploit errors (in the large sense) to break into systems: low-level implementation bugs, protocol or specification flaws, design, configuration or initialization errors, violations of organizational security policy, etc. See [1] for further detail on this topic.

Mobile Phone Security. The smartphone revolution triggered a change: mobile phones started to be used for more than just calling and texting. The new security needs were mainly driven by secure transactions or secure processing requiring the involvement of more peripherals than secure elements and smartcards could possibly handle (at least at an acceptable price). For example, peripherals for the user interface needed to be trusted so as to be sure in particular that "what you sign is what you see". As a result, the part of the device to secure, more precisely the scope of the software and hardware that needed to be trusted, the so-called Trusted Computing Base (TCB), was extended to include the OS kernel, and because of that, became too complex to secure.
In an attempt to cope with the new security needs coming from the mobile phone industry, the concept of the Trusted Execution Environment (TEE) was introduced in 2001 [1]. The TEE concept was mainly driven by the idea that taking a large OS such as Linux or Android out of the scope of the TCB and replacing it with a secure, small and carefully designed microkernel was a major step toward building an architecture that could resist logical attacks [1]. However the TEE concept that is now widely deployed on smartphones was only the first step towards addressing new security needs. Even a small TCB such as the TEE is still too complex and error prone (e.g. [20,21]). The situation is even worse as the IoT brings many new challenges.

IoT: Logical TCB and Logical Attack Surface Becoming Too Large. With the IoT, the TCB of devices involved in connected architectures and their surface of attack become much wider. Devices to secure are indeed more diverse than "just" mobile phones: instead they handle a larger number of various peripherals to be secured. They also include large and complex software stacks, with rich OSs and kernels, some of which are essential to security. The IoT is thus taking the need for security into a new era where sub-systems and peripherals that need to be secured are complex and have a very large surface of attack. In other words the scope of the TCB significantly expands and becomes the new weak link and the one we refer to here as the "resistance to logical attacks".
In addition, IoT use cases create new situations where assets that need to be protected are not just virtual, but also physical: goods, infrastructures, lives, *etc*. The effects of large-scale attacks are no longer limited to tampering with crucial data or creating improper transactions (issues which can usually be avoided or traced back with proper risk management processes), but could also potentially include irremediable physical destruction. The prospects and business model for attackers become much more attractive. In many cases the risk for services and industries may become incommensurate.
It is quite obvious that it is essential to design architectures that rely on TCBs that are as small and simple as possible, and this was the idea behind the TEE concept. But while this is generally possible, even if the TCB becomes small it is always too

complex and too error-prone to achieve the right level of security for the TCB if developed with standard development technologies, at least for the kernel part of them, and this even for a group of security experts.

The Challenge of Securing OSs and Kernels. Various public databases (such as [2]) indeed provide statistics on public bugs or vulnerabilities on all kinds of software. These databases clearly show that current OSs and kernels suffer from a great number of errors and weaknesses, no matter who writes them, and no matter how long they have been in the field. For example new errors are still reported in the thousands every year on "well-known" systems such as Linux.
This situation is basically due to the inherent complexity of such OSs and kernels, which rely more and more on complex and sophisticated hardware. OSs and kernels are by nature concurrent and hugely complex because of the need to support various kinds of peripherals (interruption handling becomes more and more difficult), the performance objectives (*e.g.*, complexity of cache management), the resource consumption issues (*e.g.*, the need for a sophisticated power management), and so on

This complexity increases with time, increases with new IoT architectures and increases when it comes to real microprocessors (as opposed to microcontrollers).
Therefore, in the end, the main issue boils down to being able to produce and demonstrate that the OSs and kernels that are part of the TCB are as close as possible to "zero-bug" *i.e.* are free from errors, either in their design or implementation, that could be potentially exploited for logical attacks.
We believe that the challenge posed by logical attacks can only be addressed by applying deductive formal methods at least on the kernel parts of the TCB so as to get as close as possible to "zero-bug" for these complex software components.

2.2 Using Deductive Formal Methods for OS Kernels Included in the TCB

To the extent that the OS and kernel are included in the TCB, they need to resist hackers who will try to exploit bugs and weaknesses, *i.e.* errors in the security rationale. These software parts need in particular to be as close as possible to "zerobug" with proven and auditable compliance to security properties.
Traditional software engineering techniques such as exhaustive testing or code inspections are clearly not sufficient anymore to bring the level of assurance that is needed to secure complex open kernels. This is due to the fact that there are too many different situations to consider for a kernel designer or tester and no real methods to review the quality of such kernel code in a systematic way, beside the use of proof techniques.
Instead we believe the only valid response to such complexity is a special class of formal methods, which are known as deductive techniques or proof techniques. There are indeed various kinds of formal methods all using rigorous techniques (typically mathematical and/or logical techniques) to prove corrections or compliance with security properties. Three categories of formal methods are usually distinguished:

1. Model-checking techniques, which can only be applied on models that are simple enough to be exhaustively checked. They are not applicable to real kernels, or only on oversimplified abstractions of them.
2. Static techniques, which can be applied on real code, but can only check some limited low-level properties (such as the absence of buffer overflows, or the illegal use of pointers). They are certainly useful, but are far from being sufficient, as they cannot address the high-level properties at stake (integrity, confidentiality, isolation, etc.).

3. Deductive (or proof) techniques, which are the most expensive to apply, as formal proof cannot be done completely automatically. But they are the only ones, which allow proving virtually any security or safety property.

Prove & Run has developed a formal language and a dedicated environment (ProvenTools) that takes advantage of decades of research and advances in the scientific field of formal methods so as to make this formal verification process more efficient and also to bring even more confidence [8,9]. Using this environment, we can fully prove the most complex parts of any TCB, in particular OSs and kernels, so as to leave hackers with no vulnerabilities to exploit, even in large, complex and open systems.
Some parts of the TCB such as the secure boot, secure configuration software or security applications do not necessarily have to be proved: they are quite sequential and their complexity is amenable to traditional validation techniques. They would also be, and this is probably not a surprise, the easiest to prove, if needed. So the decision on which techniques to use for validation of these "easy" parts is up to the security architect and/or up to the security evaluation and certification authorities. These parts will, depending on the situation, be formally verified or not, but in any cases they will have to be brought to the right level of confidence.

2.3 Reusing Proven Building Bricks Across Industries

Producing a security-proven OS kernel using deductive formal methods is not a straightforward and simple task, even if our formal language and dedicated environment facilitate this task. In order to match the cost requirements of industrial deployment, it is critical to be able to reuse the same software components across industries and to mutualize costs.
We believe it is important to provide off-the-shelf reusable proven kernels, *i.e.* COTS, that will make securing IoT architectures simple and cheap, as well as fast to build and evaluate. In line with this vision, Prove & Run has already developed three such off-the-shelf kernels, formally proven and ready to use and certify:

1. ProvenCore, a fully proven and secure OS kernel, POSIX conformant and dedicated to microprocessors. ProvenCore has been optimized to run in the secure part of ARM® Cortex®-A TrustZone®-enabled microprocessors [8].
2. ProvenCore-M, a fully proven and secure OS kernel, providing a large subset of the same POSIX APIs and dedicated to microcontrollers. It has been also optimized to run in the newly announced secure part of ARM Cortex-M TrustZone-enabled microcontrollers.
3. ProvenVisor, a modern fully proven and secure hypervisor, providing similar functionalities as Xen[1], but with a much smaller TCB.

Prove & Run's COTS come for example with Common Criteria (CC) EAL7-ready documents which can be used by any integrator to rapidly go to any level of security certification they wish. Prove & Run's formally proven COTS are proven down to the code itself and go in fact much beyond what is required by the highest levels of CC certification.
We believe that by combining these basic security bricks we can secure virtually any IoT architecture at a very high level of security with reduced cost and time. We will illustrate this in the following part of this paper.

3 Securing the IoT with a Proven and Secure OS Kernel

[1] http://www.xenproject.org

In this section, we illustrate on a few use cases the most innovative and challenging part of the approach, *i.e.* building a very strong TCB by reusing formally proven COTS, and in particular here using ProvenCore.

The way we, at Prove & Run, propose to address this is the following. We use a separate proven secure OS kernel, *i.e.* in our case ProvenCore, to address peripherals that need to be secured and to run secure applications, in a way that allows us to:

- Keep the normal OS (Linux or any other OS on which the main functionalities of the product have been developed) and thus retain all its features and benefits,
- Use a proven OS to perform security functions so that any error in the normal OS cannot be used to compromise the TCB; this pushes the normal OS outside of the TCB,
- Design the proven OS in a way that makes it generic and secure enough to be used as COTS in virtually any IoT architecture.

We describe how this can be done on ARM architectures that account for the vast majority of the market, but the same can be transposed to other architectures. On ARM architectures and in particular on Cortex-A and Cortex-M processors, that is on ARM microprocessors and microcontrollers, there is a security mechanism called TrustZone that provides a low-cost alternative to adding a dedicated security core or co-processor, by splitting the existing processor into two virtual processors backed by hardware-based access control. This lets the processor switch between two states, i.e. two worlds, typically the "Normal World" on one side and the "Secure World" on the other side. TrustZone access control mechanisms together with its software stack (called the Monitor on Cortex-A) act as an extremely small and security oriented two-VM asymmetric security hypervisor that allows:

- The Normal World to run on its own, potentially oblivious of the existence of the Secure World and,
- The Secure World to have extra privileges such as the ability to have some part of the memory, as well as some hardware peripherals, exclusively visible and accessible to itself.

In the proposed architecture, the proven secure OS kernel, ProvenCore in our case, is used as the kernel for the Secure World, and the rich but error-prone OS (Linux, Android, etc.) is placed/left in the Normal World. The critical peripherals that have been selected to be controlled by the secure OS are configured, typically during the secure boot, to be only accessible and visible to the Secure World.

So in the architecture we propose, only applications that implement security functions (*i.e.* firmware update, firewalling and more generally security policy enforcement applications, IDS, VPNs, authentication protocols agents, security trace loggers, *etc.*), and potentially the drivers of security-sensitive peripherals need to be adapted. The biggest part of the software stack, *i.e.* the Normal OS and its applications are left untouched.

3.1 Securing Firmware Update Over-The-Air

As a first representative example we present how to secure a Firmware Over-The-Air (FOTA) management system. The ability to update firmware in the field is now a requirement in most markets. Firmware updates are an essential security mechanism with both a curative use to update the firmware when vulnerabilities have been identified, and a preventive use to block unauthorized firmware updates by attackers.

From a high-level point of view, FOTA systems:

- Improve the value of existing devices by enhancing their functionality and performance: Tesla Motors uses firmware updates to add new features to existing cars.
- Eliminate costly recalls/local maintenance or physical replacements because of functional or security bugs. Again Tesla Motors has recently used its FOTA system to fix a vulnerability in their cars before the vulnerability was disclosed publicly.
- Reduce testing and support costs by keeping all devices at the same version, so there is no need to support older versions of the software.

A firmware update is a highly sensitive operation, carrying a massive security risk, as an attacker can misuse it to break or disable the device, unlock restricted features, or load a modified version of the firmware with disabled security and/or safety features.

We show here how to secure a FOTA system (and more generally any firmware update mechanism). The approach can be used to integrate any given FOTA system (for example a third-party one), or to design a new one. As in most security applications, only a part of the application is security sensitive: the so-called TCB. The idea is to port or develop the TCB part of the application as a ProvenCore (*i.e.* secure) process, the other part potentially remaining unchanged and running on the normal OS. By running the secure part of the application on ProvenCore we achieve two objectives: (1) we can make sure that the secure part is actually doing what it is meant to do, because it is relying and executing on a sound foundation, *i.e.* the formally proven ProvenCore and (2) that the secure part cannot be tampered with as it is protected (and isolated) from other applications and from the external world by ProvenCore.

As an example, our Secure Firmware Update (SFU) solution uses a split architecture:

- The SFU Agent, which is the secure part of the FOTA remote application, runs in the Secure World, on top of ProvenCore, and performs the most sensitive operations, such as verifying the authenticity of the update, making the installation decision and installing the update.
- The SFU Client runs in the Normal World, and retrieves the firmware image, for instance from the Internet or from another peripheral, potentially rebuilding it if delta binaries are used for bandwidth considerations. The SFU client has no security responsibility, and other third-party software can replace it.

The security comes here from the isolation of the sensitive update process in the SFU agent. The agent runs independently of the Normal World, and ProvenCore protects it from interference by other software components. In addition ProvenCore provides a solid ground for the execution of the SFU Agent that makes sure that the Agent is effectively doing what it is written and meant to be

In practice, Prove & Run's SFU solution has been designed to strengthen existing firmware management systems by protecting the most sensitive parts of the firmware update operation: the integrity and authenticity checks, and the firmware's activation. Our SFU solution brings many advantages:

- The integrity and authenticity of the firmware is guaranteed by the isolation of the firmware update process, even if the operating system from which the firmware image comes has been compromised.
- Attackers cannot abuse the firmware update mechanism by bypassing integrity checks. If an interface exists with the secure boot solution, they also cannot modify the firmware directly.

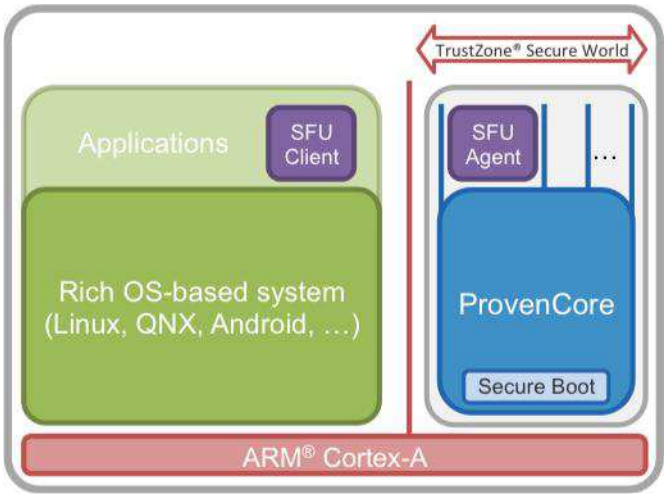


Fig. 1: FOTA Architecture

The way we use ProvenCore to secure the FOTA process typically works for any other security function. In the end it comes down to the following:

- Identifying in a given security function the security sensitive part, *i.e.* the so-called TCB, then developing/porting this part as a ProvenCore process.
- Configuring ProvenCore to control sensitive resources and peripherals whenever needed, potentially rewriting some of the drivers for the secure peripherals into ProvenCore drivers. In the case of the SFU controlled resources were for example the secure memory (only accessible to the Secure World) and potentially the cryptographic functions or peripherals used to enable the secure boot.

3.2 Securing a Virtual Private Network

Applications running on connected embedded devices need to communicate securely with remote peers (other embedded devices, gateways or servers): they must be assured that their communications cannot be listened to or modified on the way. Even on “private” networks, who can be listening is never clear.
More precisely, applications and remote peers must be able to:

- Authenticate each other to make sure they know for sure who is on the other end of the line,
- Authenticate the messages they exchange to make sure that the messages they exchange are not modified on the way,
- Encrypt the messages they exchange to make sure that the content of the messages are not disclosed to any potential listener.

It is hard to provide this level of security, as it requires experience with cryptography and protocol design. For example, SSL:

- Requires the application to embed a large and complex library,
- Requires the application to be modified to make use of this feature,
- Requires the developer of the application to understand how to use this library securely.

Virtual Private Network (VPN) implementations offer a solution to these issues:

- Supported at the OS level, so applications don't have to be modified,
- Simpler configuration, performed once for the whole device,
- Can apply to some or all of the communications going in and out of a device,

Limitations of Traditional VPN Implementations. Nevertheless, the confidentiality and authenticity of messages exchanged across a VPN can only be trusted if the VPN agents running on each device are protected from attacks:

- If an attacker can use a local application to inject a new public key certificate in the certificate store of the VPN agent, the attacker can perform a man-in-the-middle attack.
- If an attacker can use a local application to read the private certificate store of the VPN agent, the certificates can be leaked, enabling the attacker to impersonate this device or to perform a man-in-the-middle attack.
- More importantly if an attacker can perform a privilege escalation or more generally corrupt the hosting OS kernel they can easily perform any of the attacks above and much more.

The proposed architecture consists in placing the VPN agent in the Secure World and developing a proxy in the Normal World that provides the VPN interface to Normal World applications. It is also possible to reduce the scope of the agent by only keeping its Trusted Computing Base (TCB) on the secure side, and putting the non-TCB part of the agent into the proxy. We did this for example for OpenVPN. The Normal World (in our case Linux) uses OpenVPN (or more generally the secured VPN) as if it were a normal Linux-based OpenVPN implementation. But the secure channel is fully secured against remote attacks. The architecture can be summarized in the following figure:

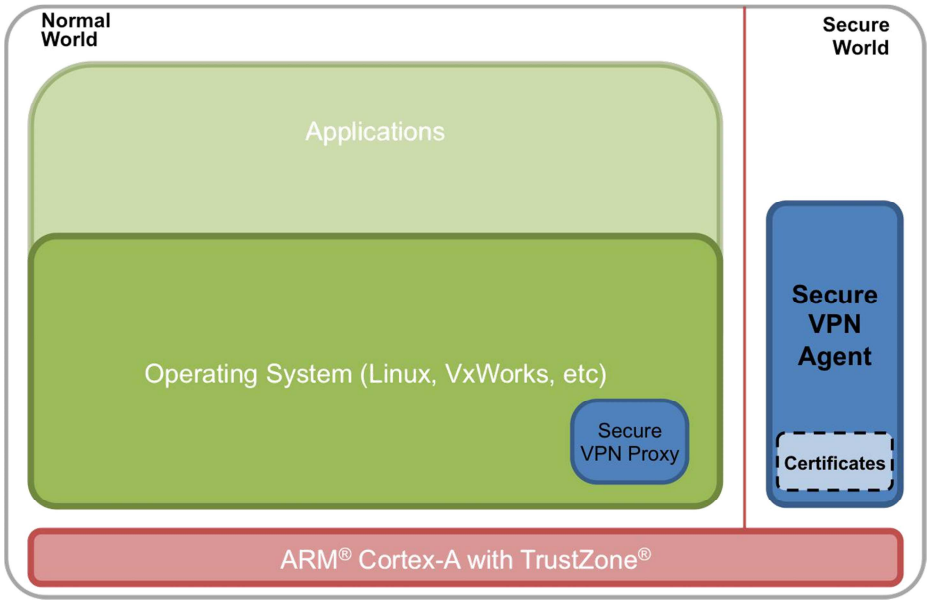


Fig. 2: Architecture for the secured VPN implementation

Security Rationale. VPN agents cannot be fully trusted because they run in the same address space as the OS (Android, Linux, *etc*.), where they are vulnerable to the huge number of local and remote attacks that affect traditional operating systems. This also means that it may not be possible to recover control of compromised devices remotely.
The Secure VPN Solution is secure because:

- Thanks to TrustZone, the Secure VPN Agent executes in a secure area, protected from attacks from applications running on the OS.
- Thanks to TrustZone, the Secure VPN Agent relies on certificates whose authenticity and confidentiality are protected from any attacks coming from the OS.
- The kernel of the Secure VPN is formally proven for higher security.
- The authenticity of the Secure VPN is protected by a secure boot mechanism anchored in the hardware security features of the board.

In order to better explain the importance of using a formally proven secure kernel we consider a simpler use case, that of a secure filter.

3.3 Securing a Command and Data Filter

Let us consider the situation where commands are received from the outside (typically from an administration server) and data is regularly sent from the device to the cloud. This is typical of many IoT endpoints. Let's then identify the complete list, type and constraints that should apply to both incoming commands and outgoing data. Once this is done, it is then possible to develop a security filtering application that ensures that only messages that comply with the identified types and constraints go in and out

Such a security application would have been more than useful to prevent attacks such as [7].

When such a security application runs on a normal OS such as Linux, we get the architecture shown in the following figure:

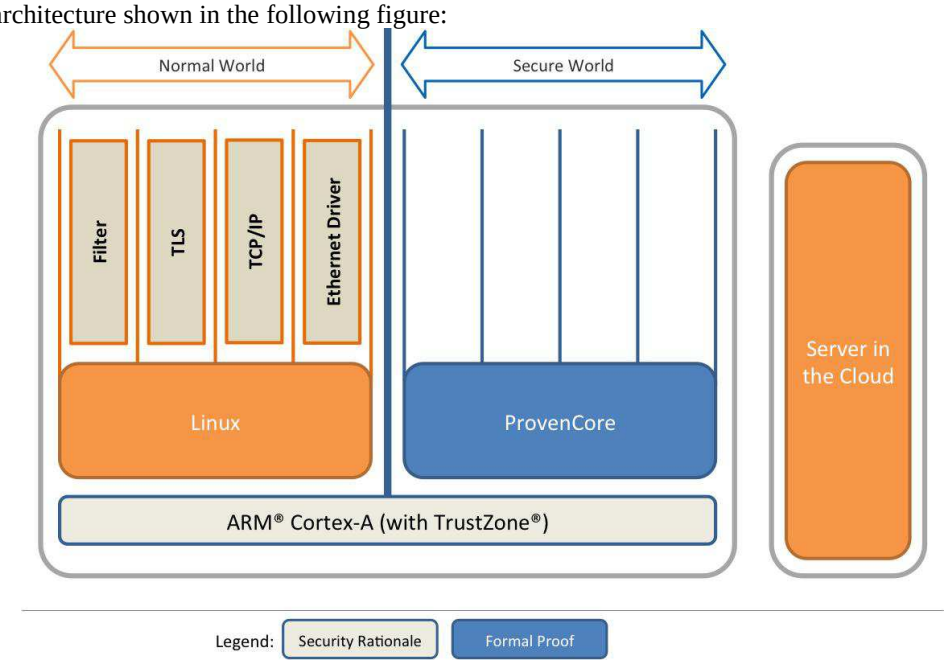


Fig. 3: Traditional architecture for a command and data filter

A hacker will typically try to send corrupted messages (*e.g.*, by injecting messages on the communication link between the trusted server and the device). Depending on the message, the tampering will ideally be detected and blocked either by the communication driver, the TCP/IP stack, the TLS layer (we assume here there exists such a security layer) or by the filtering application, each one checking its part of the message. The problem is that any problem in any of these layers will potentially be exploited to perform a privilege escalation attack, to corrupt or tamper with the underlying OS kernel and to bypass the complete filtering chain, as shown in the following figure:

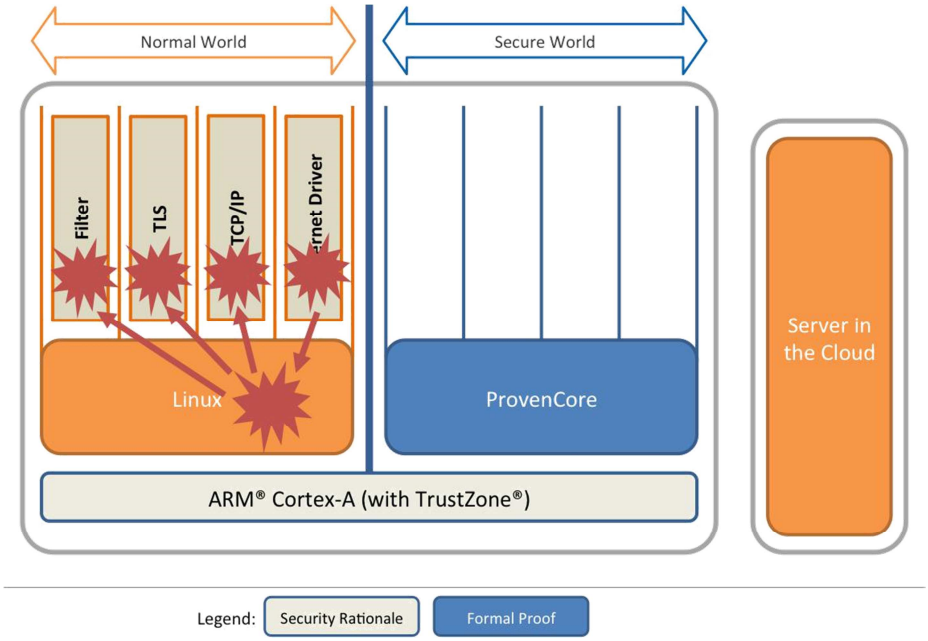


Fig. 4: Vulnerabilities of the traditional architecture

Now in the case of Linux (or of most normal OSs), the privilege escalation is even not required and the attack is even easier because software components such as the drivers or (part of) the TCP/IP stack are executing in privileged kernel mode.
In any case, if the same software components are executed and protected by ProvenCore such remote attacks are not possible anymore, as ProvenCore has been formally proven to resist to such logical attacks in all possible cases. This is explained in the following figure:

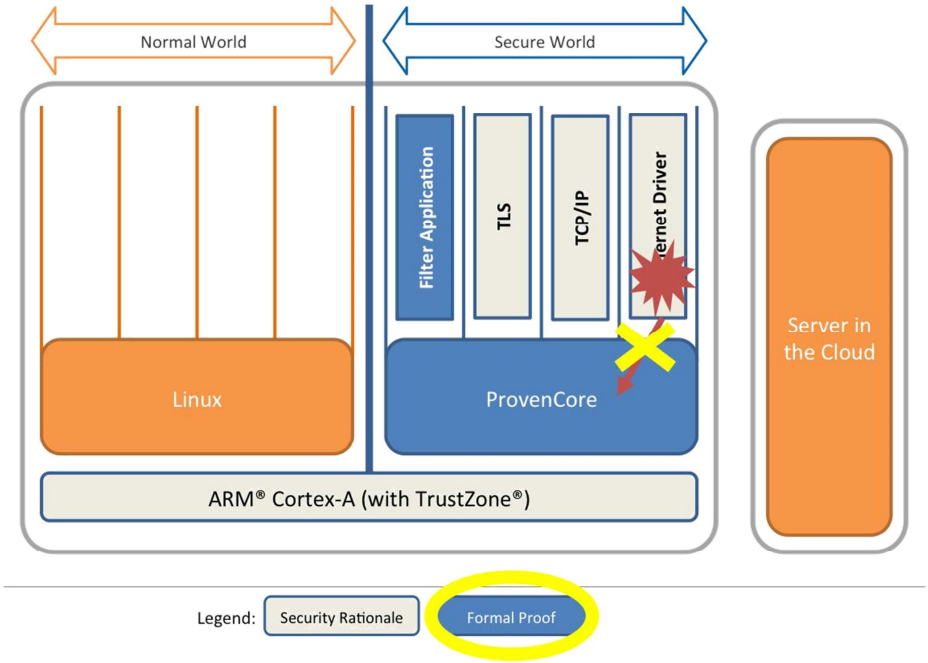


Fig. 5: Secure architecture for a command and data filter

In addition, the Ethernet peripheral is in the Secure World, so, even if the Linux OS in the Normal World is compromised, it cannot get any access to the peripheral without going through the driver, protocol stack and filtering application running in the Secure World

In this setup it is not strictly necessary to prove the Filtering Application, as it is typically a very simple sequential application for which state-of- the-art development and quality-management methodologies alone allow to reach a very high level of quality and trust. Nevertheless proving such applications can be done at a relatively low cost and bring an even higher level of security.

4 Conclusion

In this paper we have shown that IoT architectures introduce new security challenges. We have shown that it is possible to bring a very high level of security to those IoT architectures, in a way that is both compatible with industrial requirements and with time-to-market and cost constraints. This involves reducing the scope of the TCB, which can be achieved using a combination of formally proven, secure and certification ready COTS. By combining these basic security bricks we can secure virtually any IoT architecture at a very high level of security with marginal and acceptable costs.

References

1. D. Bolignano, "Proven Security for the Internet of Things,", Embedded World 2016, Nuremberg, Germany, February 23-25, 2016.
2. National Vulnerability Database. NIST, https://web.nvd.nist.gov/view/vuln/search
3. Briefings, 2013. Black Hat Conference, https://www.blackhat.com/us-13/archives.html
4. Briefings, 2014. Black Hat Conference, https://www.blackhat.com/us-14/archives.html
5. Briefings, 2015. Black Hat Conference, https://www.blackhat.com/us-15/briefings.html
6. C. Miller and C. Valasek, "A survey of remote automotive attack surfaces". http://illmatics.com/remote%20attack%20surfaces.pdf
7. C. Miller and C. Valasek, "Remote Exploitation of an Unaltered Passenger Vehicle". IO-Active, Seattle, WA, Tech. Rep., 2015. http://www.ioactive.com/pdfs/IOActive_Remote_Car_Hacking.pdf.
8. S. Lescuyer, "ProvenCore: Towards a verified isolation micro-kernel," EuroMILS Workshop, 10th HiPEAC Conference, Amsterdam, Netherlands, January 19-21, 2015.
9. P. Bolignano, T. Jensen and V. Siles, "Modeling and abstraction of memory management in a hypervisor," 19th International Conference, FASE, 2016.
10. Beemer, Open Thyself! – Security vulnerabilities in BMW's ConnectedDrive. http://m.heise.de/ct/artikel/Beemer-Open-Thyself-Security-vulnerabilities-in-BMW-s-ConnectedDrive-2540957.html
11. Hack it like you stole it. https://www.pentestpartners.com/blog/hacking-the-mitsubishi-outlander-phev-hybrid-suv/
12. Car Hacking Research: Remote Attack Tesla Motors. http://keenlab.tencent.com/en/2016/09/19/Keen-Security-Lab-of-Tencent-Car-Hacking-Research-Remote-Attack-to-Tesla-Cars/
13. C. Yan, W. Xu and J. Liu, "Can You Trust Autonomous Vehicles: Contactless Attacks against Sensors of Self-driving Vehicle," Defcon 24, August 4-7, 2016.
14. Y. Burakova, B. Hass, L. Millar and A. Weimerskirch, "Truck Hacking: An Experimental Analysis of the SAE J1939 Standard," 10th USENIX Workshop on Offensive Technolo-gies (WOOT 16), Austin, TX, August 8-9, 2016.
15. C. Thuen, "Time to Get Progressive With ICS / IoT Cyber Security," http://www.digitalbond.com/blog/2015/02/02/time-to-get-progressive-with-ics-iot-cyber-security/
16. Neighbor Unlocks Front Door Without Permission With The Help Of Apple's Siri, http://www.forbes.com/sites/aarontilley/2016/09/17/neighbor-unlocks-front-door-without-permission-with-the-help-of-apples-siri/
17. R7-2016-10: Multiple OSRAM SYLVANIA Osram Lightify Vulnerabilities (CVE-2016-5051 through 5059), https://community.rapid7.com/community/infosec/blog/2016/07/26/r7-2016-10-multiple-osram-sylvania-osram-lightify-vulnerabilities-cve-2016-5051-through-5059
18. How Hacked Cameras Are Helping Launch The Biggest Attacks The Internet Has Ever Seen, http://www.forbes.com/sites/thomasbrewster/2016/09/25/brian-krebs-overwatch-ovh-smashed-by-largest-ddos-attacks-ever/
19. C.Wang, X. Guo, Y. Wang, Y. Chen, B. Liu, "Friend or Foe?: Your Wearable Devices Reveal Your Personal PIN", Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, Xi'an, China, May 30 - June 3, 2016.
20. KNOXout (CVE-2016-6584) - Bypassing Samsung KNOX, http://www.vsecgroup.com/single-post/2016/09/16/KNOXout---Bypassing-Samsung-KNOX
21. Extracting Qualcomm's KeyMaster Keys - Breaking Android Full Disk Encryption, http://bits-please.blogspot.fr/2016/06/extracting-qualcomms-keymaster-keys.html
22. Objets connectés : polémique sur la sécurité du réseau français Sigfox, http://www.01net.com/actualites/objets-connectes-le-reseau-francais-sigfox-une-passoire-en-matiere-de-securite-957875.html

23. R7-2016-07: Multiple Vulnerabilities in Animas OneTouch Ping Insulin Pump, https://community.rapid7.com/community/infosec/blog/2016/10/04/r7-2016-07-multiple-vulnerabilities-in-animas-onetouch-ping-insulin-pump
24. S. Skorobogatov, "The bumpy road towards iPhone 5c NAND mirroring".
25. K. Razavi, B. Gras, E. Bosman, B. Preneel, C. Giuffrida and H. Bos,"Flip Feng Shui: Hammering a Needle in the Software Stack", Proceedings of the 25th USENIX Security Symposium, Austin, TX, August 10–12, 2016.
26. V. Sivaraman, D. Chan, D. Earl and R. Boreli, "Smart-Phones Attacking Smart-Homes", Security & Privacy Week 2016, Darmstadt, Germany, July 18-22, 2016.

Formal Verification of a Memory Allocation Module of Contiki with Frama-C: a Case Study

Frédéric Mangano[1], Simon Duquennoy[2,3], and Nikolai Kosmatov[1]

[1] CEA, LIST, Software Reliability Laboratory, PC 174, 91191 Gif-sur-Yvette France
firstname.lastname@cea.fr
[2] Inria Lille - Nord Europe, France
firstname.lastname@inria.fr
[3] SICS Swedish ICT, Sweden

Abstract. Formal verification is still rarely applied to the IoT (Internet of Things) software, whereas IoT applications tend to become increasingly popular and critical. This short paper promotes the usage of formal verification to ensure safety and security of software in this domain. We present a successful case study on deductive verification of a memory allocation module of Contiki, a popular open-source operating system for IoT. We present the target module, describe how the code has been specified and proven using Frama-C, a software analysis platform for C code, and discuss lessons learned.

Keywords: deductive verification, specification, FRAMA-C, Contiki, memory allocation.

1 Introduction

While formal verification is traditionally applied to embedded software in many critical domains (avionics, energy, rail, etc.), its usage for the Internet of Things (IoT) has not yet become common practice, probably because the first IoT applications were not considered as critical. However, with the emergence of the Internet of Things, embedded devices get massively connected to the Internet. In this context, security concerns become of utmost importance as IoT applications today often deal with sensitive data and act on the physical world. The devices are both constrained and network-facing, thus creating new opportunities for attackers and new challenges for verification. One of these challenges is to verify an embedded yet full-fledged low-power IPv6 stack underlying many potentially critical IoT applications. In this paper we focus on the Contiki OS [1], and in particular on its memory allocation module `memb` providing a generic mechanism for allocation of a bounded number of blocks of any given type.

Contributions. This experience report paper advocates the use of formal verification for IoT and presents a case study on verification of the `memb` module performed with FRAMA-C [2], a rich and powerful toolset for analysis of C code. We formally specify `memb` operations and prove it using the deductive verification tool FRAMA-C/WP. We emphasize two specific issues: the generic nature of the module (resulting in heavier pointer arithmetics and casts) and the need to specify the number of available blocks (requiring an axiomatic definition of occurrence counting). Finally, we describe the verification results and discuss lessons learned.

2 Contiki and its Memory Allocation Module

Contiki and Formal Verification. Today's IoT software is highly critical because it runs on hardware that is able to sense or even act on physical things. A compromised IoT device may get access to sensitive or private data. Worse, it might become able to take action on the physical

world, potentially with safety consequences. Examples of such include reconfiguring an industrial automation process, interfering with alarms or locks in a building, or in the e-health domain, altering a pacemaker or other vital devices.

Contiki is an Operating System for the Internet of Things. It was among the pioneers in advocating IP in the low-power wireless world. In particular, it features a 6LoWPAN stack [3], that is, a compressed IPv6 stack for IEEE 802.15.4 communication. This enables constrained devices to interoperate and connect directly to the Internet. Sensors, actuators or consumer devices can be brought together and create applications in various areas such as home automation or the smart grid.

Contiki is targeted at constrained devices with a 8, 16 or 32-bit MCU and no MMU. The devices usually feature a low-power radio module, some sensors, a few kB RAM and tens of kB ROM. Contiki has a kernel, written in portable C, that is linked to platform-specific drivers at compile-time. At the time of writing, it supports 36 different hardware platforms.

When Contiki started in 2003, the focus was on enabling communication in the most constrained devices, with no particular attention given to security. As it matured and as commercial applications arose, communication security was added at different layers, via standard protocols such as IPsec or DTLS. The security of the software itself, however, did not receive much attention. Although a continuous integration system is in place, it does not include formal verification. While formal verification has already been applied to microkernels and Cloud hypervisors (see [4, Sec. 5] for related work), we are not aware of similar verification projects for IoT software.

Contiki's `memb` Module. In this case study, we turn our attention to Contiki's main memory management module: `memb`. To avoid fragmentation in long-lasting systems, Contiki does not use dynamic allocation. Memory is pre-allocated in blocks on a per-feature basis, and the `memb` module helps the management of such blocks. For instance, the routing module provisions for N entries, which are stored in a static memory area N times the size of a routing entry. Entries are managed by allocating and freeing blocks at runtime.

The module `memb` offers a simple API, enabling to *initialize* a `memb` store, *allocate* a block, *free* a block, *check* if a pointer refers to a block inside the store and *count* the number of allocated blocks. `memb` consists in about 100 lines of code but is one of the most critical elements of Contiki, as the kernel and many modules rely on it. A flaw in `memb` could result in attackers reading or writing arbitrary memory regions, crashing the device, or triggering code execution.

The Contiki code base involves a total of 56 instances of `memb`. Not all are included in a given Contiki firmware, but a subset is included depending on the application and configuration. `memb` is used for instance for HTTP, CoAP (lightweight HTTP), IPv6 routes, CSMA, the MAC protocol TSCH, packet queues, network neighbors, the file system Coffee or the DBMS Antelope.

3 Specification and Deductive Verification with FRAMA-C / WP

FRAMA-C [2] is a popular software analysis platform for C programs that offers various static and dynamic analyzers as independent plugins. The plugins include the deductive verification tool WP, abstract interpretation based value analysis, dependency and impact analysis, program slicing, test generation, and runtime verification, among others. FRAMA-C comes with a behavioral specification language ACSL [5]. The user specifies the desired properties of their program by adding ACSL annotations (preconditions, postconditions, loop invariants, assertions, etc.). These annotations are written in special comments `/*@ <annotation>*/` or `//@ <annotation>`. FRAMA-C/WP can be used then to establish a rigorous mathematical proof that the program satisfies its specification. Technically, it relies on automatic provers (SMT solvers) that try to prove the theorems (*verification conditions*, or VCs) automatically generated by WP.

```
1  /*@ requires n ≥ 0 ∧ \valid(t+(0..n-1));
2      assigns \nothing;
3      ensures \result ≠ 0 ⇔
4        (∀ ℤ j; 0 ≤ j < n ⟹ t[j] == 0);
5  */
6  int all_zeros(int *t, int n) {
7    int k=0;
8    /*@ loop invariant 0 ≤ k ≤ n;
9        loop invariant ∀ ℤ j; 0 ≤ j < k ⟹ t[j] == 0;
10       loop assigns k;
11       loop variant n-k;
12   */
13   while(k < n){
14     if (t[k] ≠ 0)
15       return 0;
16     k++;
17   }
18   return 1;
19 }
```

Fig. 1: Function `all_zeros` specified in ACSL (file `all_zeros.c`).

3.1 Specification of C Programs with ACSL

ACSL (ANSI/ISO C Specification Language) [5] is a formal behavioral specification language offered by FRAMA-C and shared by different FRAMA-C analyzers. It allows its users to specify functional properties of C programs similarly to Eiffel [6] and JML [7]. It is based on the notion of function contract. The *contract* of a function f specifies the preconditions that are supposed to be true before a call of f (i.e. ensured by the caller), and the postconditions that should be satisfied after the call of f (and should be thus established during the verification of f). The preconditions are specified in **`requires`** clauses, while the postconditions are stated in **`ensures`** clauses. An additional type of postconditions, specified in an **`assigns`** clause in ACSL and used for the so-called *frame rule*, states a list of locations of the global memory state that may have a different value before and after the call. When the contract of f contains such a clause, all locations that are not mentioned in it must have the same value before and after the call of f. Function contracts can be also represented in the form of different behaviors.

Predicates used in annotations are written in typed first-order logic. Variables have either a C type or a logical type (e.g. $\mathbb{Z}$ or $\mathbb{R}$ for mathematical integer or real numbers). The user can define custom functions and predicates and use them in annotations together with ACSL built-ins. Indeed, ACSL features its own functions and predicates to describe memory states. In particular, regarding memory-related properties, **`\valid`**`(p)` expresses validity of a pointer `p` (i.e. being a non-null pointer which can be safely accessed by the program); `\base_addr(p)`, `\block_length(p)`, and `\offset(p)` express respectively the base address, the size of the memory block containing `p` and the offset of `p` inside it (in bytes), while `\initialized(p)` is true whenever the pointed location `*p` has been initialized. We refer the reader to the ACSL documentation [5] for more details on the ACSL features.

Specification Example. Figure 1 illustrates a C function `all_zeros` specified in ACSL. This function receives as arguments an array `t` and its size `n` and checks whether all elements

```
1 /*@ requires len ≥ 0 ∧ \valid(t+(0..len-1));
2     assigns \nothing;
3     behavior present:
4       assumes ∃ ℤ i; 0 ≤ i < len ∧ t[i] == elt;
5       ensures 0 ≤ \result < len ∧ t[\result] == elt;
6     behavior absent:
7       assumes ∀ ℤ i; 0 ≤ i < len ⟹ t[i] ≠ elt;
8       ensures \result == -1;
9     disjoint behaviors;
10    complete behaviors;
11  */
12 extern int find_value(int *t, int len, int elt);
```

Fig. 2: Function `find_value` specified in ACSL.

of the array are zeros. If yes, it returns a nonzero value, and 0 otherwise. The function contract contains a precondition (line 1) and postconditions (lines 2–4). The precondition states that the input array contains `n` valid memory locations at indices `0..(n-1)` that can be safely read or written, and that the size `n` is non negative. This property must be ensured by the caller and should be thus specified in the precondition. The **`assigns`** clause at line 2 states that the function is not allowed to modify any non-local variable. Without this clause, an erroneous implementation writing zeros in all elements of the array and returning 1 would be considered correct with respect to the contract. Finally, the clause at lines 3–4 states that the result is nonzero if and only if all elements of the array are equal to zero. The loop contract at lines 8–12 will be discussed in the next section.

Figure 2 provides another example of a specified function. This function is only declared and takes as arguments an array `t` of size `n` and some element `elt`. It must return an index `i` such than `t[i] = elt`, or -1 if there is no such index.

The precondition (line 1) and the **`assigns`** clause (line 2) are similar to the ones of the function `all_zeros`. The postcondition is expressed through two named behaviors. They correspond to the two different cases of the contract. First, the behavior `present` states that, if the searched element `elt` is present in the array (line 4), the function's result is an index with the expected property (line 5). The behavior `absent` corresponds to the opposite case (line 7). In that case, the function returns -1 (line 8). Additionally the **`disjoint behaviors`** clause states that these behaviors are mutually exclusive (line 9), while the **`complete behaviors`** clause indicates that they cover all the possible cases of the function (line 10). In other words, being both disjoint and complete guarantees that one and only one behavior applies at each function call.

3.2 Deductive Verification with FRAMA-C / WP

Deductive program verification is one of the software verification techniques that consist in establishing a mathematical proof that a given program satisfies its specification. The weakest precondition calculus reduces any deductive verification problem to establishing the validity of first-order formulas called *verification conditions*. The WP plug-in [2] of FRAMA-C performs weakest precondition calculus for deductive verification of C programs. Various automatic SMT solvers, such as Alt-Ergo, CVC4 and Z3, can be used to prove the verification conditions generated by WP.

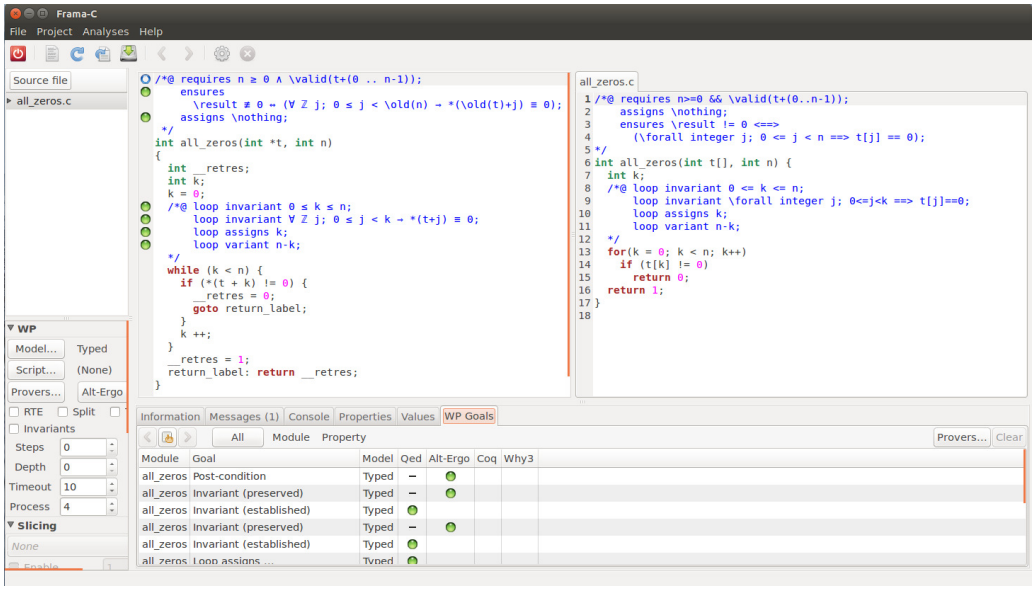


Fig. 3: Successful proof for the program of Figure 1 with FRAMA-C/WP.

Proof Example. Let us illustrate deductive verification with WP on the example in Figure 1. The command `frama-c-gui -wp all_zeros.c` runs the proof with WP on this example and shows the results in the FRAMA-C GUI. Suppose first that the user has specified the contract at lines 1–4 without writing the loop contract at lines 8–12. In this case, the proof of the post-condition will not be successful. Indeed, in presence of loops, since the number of loop iterations is unknown, the deductive verification tool requires a *loop invariant*, i.e. an additional property on the program state that is true before the loop and after each complete loop iteration. It can be specified in a loop contract using **`loop invariant`** and **`loop assigns`** clauses. The clause at line 8 specifies the interval of values of the loop variable `k`. The clause at line 9 specifies that all elements at indices `0..(k-1)` are equal to 0 (that is indeed true after any complete loop iteration otherwise the loop execution was interrupted at line 15). Similarly to **`assigns`**, the **`loop assigns`** clause specifies the variables (but both global and local ones in this case) that may change their value during the loop. The loop contract can also contain a **`loop variant`**, which defines a decreasing natural measure corresponding to an upper bound of the number of remaining loop iterations and is used to prove that the loop terminates. In this example, `n-k` provides such a bound (cf. line 11).

On the complete program of Figure 1 with the loop contract, WP successfully proves that this function respects its specification. In addition, it is possible to make WP check the absence of runtime errors using the option `-wp-rte`. In this case, thanks to the array validity assumed at line 1 and the interval of values specified at line 8, WP successfully proves that array access at line 14 is valid and the arithmetic operation at line 16 does not overflow.

```
1  /* file memb.h */
2  struct memb {
3    unsigned short size; // block size
4    unsigned short num;  // number of blocks
5    char *count;         // block statuses
6    void *mem;           // array of blocks
7  };
8  #define MEMB(name, btype, num)...
9  // macro used to decrare a memb store for
10 // allocation of num blocks of type btype
11
12 void  memb_init(struct memb *m);
13 void *memb_alloc(struct memb *m);
14 char  memb_free(struct memb *m, void *p);
15 ...
```

```
1  /* file demo.c */
2  #include "memb.h"
3  struct point {int x; int y};
4
5  // before preprocessing,
6  // there was the following macro:
7  // MEMB(pblock, struct point, 2);
8
9  // after preprocessing, it becomes:
10 static char pblock_count[2];
11 static struct point pblock_mem[2];
12 struct struct memb pblock = {
13   sizeof(struct point), 2,
14   pblock_count, pblock_mem };
15 ...
```

Fig. 4: (a) Extract of file `memb.h` defining a template macro `MEMB`, and (b) its usage (in file `demo.c`) to prepare allocation of up to 2 blocks of type **`struct`** `point`

4 Verification of `memb` with FRAMA-C

4.1 Declaration of Memory Allocation Data via a Pseudo-Template

A `memb` store is represented by the **`struct`** `memb` structure (see Fig. 4a, lines 2–7). Being written in C, it does not allow polymorphism. Instead, it stores as a field of the structure the size of the block type it is meant to store, which enables the implementation to dynamically compute the addresses of the blocks.

The actual blocks are stored in an array (referred to as field `mem`), statically allocated using a global array definition. This is illustrated in Fig. 4b, lines 10–14, for 2 blocks of type **`struct`** `point`. Since its length varies, the array cannot be declared directly in the structure, that is why the structure contains a pointer that is initialized to point to the global array. The `count` array is allocated in the same fashion.

All the fields of the `memb` structure are initialized at compile-time and shall not change when the program executes. That is conveniently realized using the `MEMB` macro (whose definition is omitted in Fig. 4a), which relies on the preprocessor in order to

- generate the global array definitions for the `count` and `mem` arrays,
- declare an initialized `memb` store.

Lines 10–14 of Fig. 4b show the result of line 7 after the preprocessing.

4.2 Specifying Operations

First, we specify the functions of `memb` in ACSL. Let us describe here the contract for the allocation function `memb_alloc` shown in Fig. 5. Its ACSL contract contains preconditions (**`requires`** clauses) that are assumed to be ensured by the caller, and postconditions (**`ensures`** clauses) that should be ensured by the function at the end of its execution and thus proved by the verification tool. Lines 2–3 specify that the store respects a global validity property before and after the call. A specific behavior can be expressed in ACSL using a **`behavior`** section, which defines additional postconditions whenever the behavior guard given in the **`assumes`** clause is satisfied. The contract may stipulate that the given behaviors are disjoint and complete (lines 16-17), the verification tool verifies it as well. The `memb_alloc` function has two behaviors:

1. If the `memb` store is full, then it is left intact, and `NULL` is returned (lines 12–15).
2. If the `memb` store has at least one free block, then it must be guaranteed that:

```
1  /*@
2    requires valid_memb(m);
3    ensures valid_memb(m);
4    assigns m→count[0 .. (m→num - 1)];
5    behavior free_found:
6      assumes ∃ ℤ i; 0 ≤ i < m→num ∧ m→count[i] == 0;
7      ensures ∃ ℤ i; 0 ≤ i < m→num ∧ \old(m→count[i]) == 0 ∧ m→count[i] == 1 ∧
8        \result == (char*) m→mem + (i * m→size) ∧
9        ∀ ℤ j; (0 ≤ j < i ∨ i < j < m→num) ⟹ m→count[j] == \old(m→count[j]);
10     ensures \valid((char*) \result + (0 .. (m→size - 1)));
11     ensures _memb_numfree(m) == \old(_memb_numfree(m)) - 1;
12   behavior full:
13     assumes ∀ ℤ i; 0 ≤ i < m→num ⟹ m→count[i] ≠ 0;
14     ensures ∀ ℤ i; 0 ≤ i < m→num ⟹ m→count[i] == \old(m→count[i]);
15     ensures \result == NULL;
16   complete behaviors;
17   disjoint behaviors;
18 */
19 void *memb_alloc(struct memb *m);
```

Fig. 5: (Simplified) ACSL contract for allocation function `memb_alloc` (file `memb.h`)

- the returned block is properly aligned in the block array (line 8),
- the returned block was marked as free, and is now marked as allocated (line 7),
- the returned block is valid, i.e. points to a valid memory space of a block size that can be safely read or written to (line 10),
- the states of the other blocks have not changed (line 9),
- the number of free blocks is decremented (line 11, see also Sec. 4.3).

4.3 Keeping Track of Free Blocks

When allocating, we may need to ensure the `memb` store is not full, that is, some blocks are available. To this end, we make assumptions on their number that we compute using a logic function named `_memb_numfree`. For instance, requiring that at least n blocks are free ensures that the n subsequent allocations will succeed. The specification states that the number of free blocks is decremented when allocating, and incremented back when a block is freed. Allocation succeeds if and only if the number of free blocks before the allocation is non-zero.

A `memb` store uses its `count` array to determine whether its i^{th} block is free ($count[i] = 0$) or allocated ($count[i] = 1$). In the logic world of ACSL, counting the number of free blocks, or more precisely counting the number of occurrences of 0 in the `count` array, requires an inductive definition using sub-arrays:

- If $to \leq from$, then $count[from \ldots to[$ obviously contains no zeros.
- If $from > to$ and $count[to - 1] = 0$ then $count[from \ldots to[$ contains one more zero than $count[from \ldots to - 1[$.
- If $from > to$ and $count[to - 1] \neq 0$ then $count[from \ldots to[$ contains as many zeros as $count[from \ldots to - 1[$.

In our specification, we use a more general inductive definition from [4], as well as a few auxiliary lemmas proven by induction in the proof assistant Coq (v.8.4pl6).

4.4 Deductive Verification Results

The current specifications of the `memb` modules are fully proven automatically using FRAMA-C/WP Magnesium-20151002, Alt-Ergo v.0.99.1, CVC4 v.1.4 and Z3 v.4.4.2. The ACSL specification of `memb` is 115 lines of code long, for a total of 259 lines in the header file. To prove

```
15 ...
16 /* file demo.c, continued */
17 /*@
18   requires valid_memb(&pblock) ∧ pblock.num == 2;
19   requires pblock.size == sizeof(struct point);
20 */
21 void main() { // all contracts proven with WP except out-of-bounds pointer line 28
22   memb_init(&pblock);
23   /*@ assert _memb_numfree(&pblock) == 2; */
24   void *obj1 = memb_alloc(&pblock), *obj2 = memb_alloc(&pblock);
25   /*@ assert _memb_numfree(&pblock) == 0 ∧ obj1 ≠ NULL ∧ obj2 ≠ NULL; */
26   /*@ assert \valid((char*) obj1 + (0 .. sizeof(struct point)-1)); */
27   /*@ assert \valid((char*) obj2 + (0 .. sizeof(struct point)-1)); */
28   /*@ assert \valid((char*) obj1 + sizeof(struct point)); */ // UNPROVEN - invalid
29   memb_free(&pblock, obj1);  memb_free(&pblock, obj2);
30   /*@ assert _memb_numfree(&pblock) == 2; */
31 }
```

Fig. 6: Example of ACSL-annotated function using `memb` (file `demo.c`)

it, 32 additional lines of annotations were required in the implementation file, summing up to 154 lines. 126 verification conditions are generated. The total effort required for specification and verification of the module can be estimated as 3 man-months.

Fig. 6 shows an annotated function using `memb` that can be automatically proven with FRAMA-C/WP, except for line 28 that contains an out-of-bounds pointer. Thus, out-of-bounds accesses are automatically detected thanks to the provided specification.

This verification case study also allowed to detect a potentially harmful situation. Function `memb_free` used to decrement the `count` associated to the given block, instead of setting it to 0. An awkward consequence of this is that calling `memb_free` on a block with an unusual `count` (e.g. greater than 2) would not actually free it. While this should not happen under normal circumstances, we have decided to replace that decrement operation by a set to 0. This choice makes `memb_free` both simpler and more robust, easing the verification process, and we recommend to integrate it into the production code.

5 Conclusion and Future Work

IoT software is becoming more critical and widely used. We argue that formal verification should be more systematically applied in this domain to guarantee that critical software meets the expected level of safety and security.

This paper reports on a case study where deductive verification with FRAMA-C/WP has been applied on the memory allocation module `memb`, one of the most critical and largely used components of the Contiki OS. We have described the verification methodology and results. In particular, the presented verification *formally guarantees* the absence of out-of-bounds accesses to the block array in the `memb` module.

We have emphasized two technical aspects. One is related to pointer arithmetics and casts due to the generic implementation of the module for all possible block types. The second one concerns inductive definitions and proofs necessary to count elements in the block status array and to state some properties on the corresponding counting functions. While these aspects could constitute an obstacle for formal verification of real-life C software a few years ago, they can be successfully treated today by modern verification tools like FRAMA-C/WP. This experience report also shows that automatic theorem provers have made significant progress, and that interactive proof, e.g. with Coq proof assistant, can be used in complement on remaining properties that are too complex to be proven automatically.

One future work direction is the verification of `memb` with a slightly more precise specification, including for example stronger isolation properties between blocks of the same store. This would require a better support of ACSL allocation primitives in WP (such as the `frees` clause) in order to better trace validity of individual blocks. Secondly, the results of this case study should facilitate the verification of other components of Contiki relying on `memb`. For some of them (such as `list`, defining chained lists), this could require further extensions of FRAMA-C and ACSL. Finally, specification and proof of other IoT software modules is another future work direction.

Acknowledgment. Part of the research work leading to these results has received funding for DEWI project (www.dewi-project.eu) from the ARTEMIS Joint Undertaking under grant agreement No. 621353. The second author has also been partially supported by a grant from CPER Nord-Pas-de-Calais/FEDER DATA and the distributed environment Ecare@Home funded by the Swedish Knowledge Foundation 2015-2019. Special thanks to Allan Blanchard, François Bobot and Loïc Correnson for advice, and to the anonymous referees for their helpful comments.

References

1. Dunkels, A., Gronvall, B., Voigt, T.: Contiki - A Lightweight and Flexible Operating System for Tiny Networked Sensors. In: LCN 2014
2. Kirchner, F., Kosmatov, N., Prevosto, V., Signoles, J., Yakobowski, B.: Frama-C: A software analysis perspective. Formal Asp. Comput. **27**(3) (2015) 573–609
3. Montenegro, G., Kushalnagar, N., Hui, J., Culler, D.: Transmission of IPv6 packets over IEEE 802.15.4 networks. RFC 4944 (September 2007) http://www.rfc-editor.org/rfc/rfc4944.txt.
4. Blanchard, A., Kosmatov, N., Lemerre, M., Loulergue, F.: A case study on formal verification of the Anaxagoros hypervisor paging system with Frama-C. In: FMICS 2015
5. Baudin, P., Cuoq, P., Filliâtre, J.C., Marché, C., Monate, B., Moy, Y., Prevosto, V.: ACSL: ANSI/ISO C Specification Language. http://frama-c.com/acsl.html.
6. Meyer, B.: Object-oriented Software Construction, Second Edition. Object-oriented Series, Prentice Hall, New York (1997)
7. Leavens, G.T., Cheon, Y., Clifton, C., Ruby, C., Cok, D.R.: How the design of JML accomodates both runtime assertion checking and formal verification. In: FMCO 2002

Automated and Remote Security Fuzz Testing Tool for IoT Devices

Category: specialized

J.C. Fonbonne

CEA-Leti, Minatec Campus, Université Grenoble Alpes
17, Rue des Martyrs, 38054 Grenoble, France
jean-christophe.fonbonne@cea.fr

Abstract. Security testing of IoT devices endeavors to solve detection and identification of hidden vulnerabilities in embedded software. Some additional constraints and problems appear with the economic environment where the security evaluation must be fast, automated and low cost. A convenient testing tool chain includes the three abstraction levels of (i) source code with static analysis, (ii) binary code with image verifier and (iii) operational devices with dynamic analysis. Fuzzing is one of the dynamic analysis techniques consisting in generating a large amount of deliberately malformed or unexpected packets to detect errors. This approach encounters a meaningful success in the hosted software world in the ten last years with major software companies having imposed it in the development lifecycle. To transpose this efficiency in the IoT context, fuzzing tools needs to be refined to be perfectly adapted to IoT devices in their specificity and their multiplicity. This work presents a study in which we describe an IoT oriented security tester providing seamless physical interactions with the system under test and its hardware interfaces.

Keywords: Stateful Fuzzing, Automated Security Testing, Network Interfaces, Security Protocols, Vulnerability Detection, Fault Injection, Embedded Software

1 Introduction

1.1 State of the art of connected devices system security evaluation

Most of the formal methods including model checking, static or dynamic analysis are usually used during the first step of the Secure Development Lifecycle (SDL) to test security in software. Those are conducted either in hosted or in embedded computing including IoT devices, sometimes having more relevance for one domain than for the other. The term model checking designates the approach of performing a formal confrontation of a system against its expected behavior [1]. This technique is particularly efficient when dealing with multithreaded programs running on distributed architecture. The static source code analyzer is closer to the final product in the sense that the production source code is scrutinized by automatic reasoners to predict run-time properties of the program before compiling it. The user should specify the expectation on target behavior by dictating which preconditions are required, which post-conditions are ensured, assuming a range of strict state parameters[2]. The dynamic analyzer aims to focus on the program being run on the processor by profiling the effects of the instruction flow on managed

resources e.g. memory allocations, systems calls, exception signals, thread scheduling [3].

The previously described methods do not address errors involving the functional correctness of the software, meaning that it does not guarantee that the program computes the correct result. The test operation happening in the latest steps of the SDL involves functional, integration and conformance issues. The final IoT product may have suspicious behaviors when integrated in its full working environment and with specific configuration and calibration parameters. The most subtle security vulnerabilities can be detected by misuse, abuse and fuzz testing (Cf 2.1).

1.2 Security test challenges

The list below gives a general glimpse of the constraints and dilemmas encountered in IoT security testing at product level.

Profusion of physical links and communication protocols:
IoT is the realm of smart connectivity. The most commonly deployed wireless solution are based on e.g. Bluetooth (Low Energy above all), 802.15.4 with ZIGBEE or 6LoWPAN for the aggressive resource-constrained devices, but may extend to higher rate such as 802.11 (Wi-Fi) for devices with larger power supply. Some proprietary solutions, mainly UWB oriented can also be relevant thanks to their interesting range, rate and consumption trade-off. The wired links, while of less relevance, are not incidental but generally limited to USB, Ethernet and CAN. Conversely, the transport and application protocols are unlimited in term of standard or proprietary solutions. Unlike traditional wide access network, TCP/IPv4 or v6 and their associated service layers (including security) are not necessarily ultra-dominant. Different adapted solutions are eligible by domains but further evolution must take into account the current practices: MODBUS in industrial systems, CAN in automotive, BACnet in home and building automation for instance.

Heterogeneous security objectives and normative environment
The IoT products cover a large range of applications hence resulting in disparate levels of security objectives: personal access control, digital wallets, health monitoring, automotive, home automation, building automation, smart city, industrial cyber physical systems [4]. The security guidance must adapt to the normative environment which can be anywhere from very strict (e.g. telemedicine), to virtually non-existent (e.g home automation). The adversary model can evolved over time. For instance, the Dolev-Yao model [5] is applied mid-term in industrial system or energy smart grid. But the adversary will have access to devices by stealing or buying it for reverse engineering purpose.

They could have access to first level operator credentials long-term. Additionally, innovative security standards begin to emerge in the security community e.g. PKI-less, decentralized key agreement, identification based cryptography that are more suited to massive deployment of IoT technologies.

Rich life cycle.
Nowadays most manufacturers design products that are connected and can be updated remotely. The product digital profile, including the embedded software plus the calibration and the configuration parameters may evolve for each phase of the lifecycle: personalization, distribution, configuration, operation, maintenance, ownership transfer and decommissioning. The monolithic update shall likely prevail over the package based update which is terribly painful to test because of the proliferation of version combinations. Monolithic update on memory-limited IoT

devices could be possible if the trade-off on power and data rate is acceptable. Software is not the unique source of variation for a given IoT device. A manufacturer can decide to have multiple variants of the same chip and change the source according to silicon vendors' quotation over time. Even if variants are functionally equivalent, one source may expose a security flaw which would not be mitigated by existing configuration.
The continuous deployment process [6] is especially well suited for massive deployment, including partial roll-out on a limited number of end users to experiment modification in real condition (dark launch). The fast, automated, low cost and remote security verification goes along with this novel connected manufacturing novel paradigm.

2 Related works

2.1 Security testing step by step

The first step is the security protocol and standard compliance verification. The most commonly tested primitives are authentication, key agreement, message confidentiality, secure firmware update, access right management and all the other mechanisms described in the security objectives [8].
The second step is the abuse and misuse cases (i.e. scanning) to force interactions on the target that are harmful to the system or to execute malicious processes, while remaining in the authorized perimeter of the product specification [7]. Each scenario can be derived directly from the threat model to achieve exhaustiveness.
The third step is the fuzz testing. We define fuzzing as a black-box testing technique for discovering faults in the system under test by providing unexpected inputs through external interfaces and monitoring for exceptions [9][10].

2.2 Fuzzing tools efficiency

The randomly generated input test may intuitively lead to believe that this technique has a little chance of being optimal [11] which is prima facie correct. But for high data rate interface, well known or well guessed data representation and a protocol-aware guided input generation, the experimental results are indisputable. This may explain the proliferation of fuzzing tools during the past decades [12][13].
Fuzzing tools are usually differentiated using two characteristics:

- Specialization: from the more specialized i.e. dedicated to a unique application or protocol up to the more generic framework practically offering a meta-language adaptable to any execution environment.
- Target: the tester architecture, user interface and execution mode has different flavors for local host programs, network interfaces or web server/browser.

Evaluating the quality of a fuzzing session is not that easy. Beyond the idea of coverage and bug trophies, [14] demonstrates that spectral power and signal entropy can be used as generic metrics to characterize a distribution efficiency. Alongside famous fuzzers such as SPIKE[15], SULLEY[16], SNOOZE[17], major companies from software industry promote or impose fuzzing in their SDL on equal terms with modelling, code auditing, integration and acceptance testing. Security researchers from those companies investigates on specific tools or techniques like SAGE[18].
Feedback-driven fuzzing (FDF) or evolutionary fuzzing tremendously improves the test performance [22]. American Fuzzy Loop (AFL) combines genetic algorithms and compiled markers for precise coverage measurement. Fuzzgrind[19] leverages on Valgrind dynamic binary instrumentation coupled with a constraint solver (STP) to generate winner challenges with an amazing success rate.

All those successful techniques apply to the category of hosted software with tracers, testers and targets running on the same operating system and the same processor. The difficult question is how to achieve a similar performance for remotely connected IoT devices.

3 Objectives and Requirements

3.1 IoT Security testing tool objectives

The IoT automated security testing tool should be:

- dedicated to connected cyber-physical devices,
- working in opaque box mode,
- to detect, classify, preserve and reproduce vulnerabilities,
- irrespective of the physical, link or transport communication layer.

3.2 IoT Security testing tool requirements

Physical adaptability and modularity.
The selected architecture should be agnostic of the target specificities e.g. physical links, communication protocols and channels, memory, power or computing resources, operating system and usage domain. The evaluation probe deployment should not be impacted by the proximity of the target that could be in the same processing nodes than the probe, or running within a connected device close to or far off the probe. The immediate constraint that ensues from this requirement is the strict separation of pattern generation, injection and classification. Additionally, the evaluation probe should be able to simultaneously manage several interfaces to stress the target through its multiple available communication channels.

Challenge generation and mutation.
The challenge generation follows the guidelines captured in the two essential components:

- the protocol description language (PDL) that defines the structure of the message with its vocabulary and grammar. Some specific parts of the message can be computed automatically given the right semantics, including lengths, CRC, hashes, cyphered data or fields whose presence depends on other values. Nested and recursive fields must be supported.
- the scenario manifest that defines the content of the message whether by abuse or misuse case description or by guided mutation in case of a fuzzing session. Because of the FDF feature, the mutation may depend on the previous response or the target behavior, both provided by the monitoring module.

Many security protocols are stateful processes and a naive fuzzing approach is ineffective as the challenge is not generated according to the current state. The packet inspection should provide hints to the packet forgery about state transitions in order to apply the appropriate packet template.

Scenario reproducibility by session replay gives the ability to repeat abnormal events predictively for debugging purpose. This function is achieved by keeping track of the history of numerous past challenges. Enforcing the initial seed initializing by pseudo-random number generator enable the exact challenge replay from the beginning of the session if the history depth happens to be insufficient.

Target monitoring.

During the fuzzing operation and depending on the Target Of Evaluation (TOE), an extensive amount of data is aggregated and the difficulty lies in a judicious analysis of the result, by feeding back only the relevant events and keeping track of the most note-worthy challenge/response pairs.
The inspection side of the session manifest defines (i) the inspected fields after the frame dissection, (ii) assertion expected on those fields and (iii) the associated event generated in case of failure. Nested assertions are an obvious requirement e.g.” check condition P if condition Q is true”.
The general monitoring behavior can be fined-tune in the option list depending on certain target properties e.g.:

- automatic aliveness query if useful to check if the target is still alive after each challenge in case of very quiet target
- automatic confirmation query is useful to check if the challenge is the real cause of the collected response(s) in case of very verbose target
- filtering unwanted or identical constant response is useful to reject pointless signals e.g. heartbeats or acknowledgments

The target monitoring also includes statistical analysis on field values e.g. the detection of recurrent occurrences against rare occurrences. This powerful feature gives some guidance to the forgery. This way to focus on challenge packets that produce rare response drastically hastens the coverage in a FDF session.

Controllability and robustness.
A fuzzing test may last hours, days or weeks. A very robust control center is necessary to obtain exploitable results. Live pausing, resuming, configuration updating are must-have requirements as they enable the evaluator to affect the test trend. It should be also possible to visualize the evolution of observable metrics. Positioning the target into a specific state facilitates the test automation. Any kind of unexpected events can happen during a fuzzing session. As the quality of a fuzzer is its ability to run for a long time period, the overall system has to face and react appropriately against anomalous situation as well as momentary or definitive TOE breakdown. A strong differentiation is needed to separate simple TOE silent state to major disruption. The testing infrastructure has to gracefully restart the overall session, potentially including a hardware reset, while logging the useful collected information for reproducibility and diagnosis.

4 IoT fuzzing assets and challenges

4.1 Dilemmas and hindrance of fuzzing approach

The choice of standard or open source PDL like in Scapy[20] or Wireshark[21] contributions quickens the test bench setup. Unfortunately, proprietary protocols are often used, especially at the application level. The protocol discovery phase can be time consuming to acquire proficiency about new specifications. A pre-fuzzing session consisting in capturing many different transactions can prove helpful especially coupled with a message inference tool.
Target reliability and firmware maturity are also required to achieve a successful fuzz testing. If the target has not been previously scrutinized with functional, integration and stress tests, the fuzzing session is likely to be interrupted too frequently to be effective.
The sensitive point of fuzz testing is the session completeness and how to know when a session can be stopped. Much as the method is good in input coverage, interface coverage and specification coverage, the code coverage is a metric difficult to estimate without having an internal target monitoring

4.2 Types of detected defects

As the operation is performed at later phases of the SDL, the fuzz testing is more suited for rating implementation and operational vulnerabilities. The predominant types of detected bugs belong to the family of memory corruption resulting in stack, heap or buffer overflow but integer wrap around or race condition are also common.

At higher layer, one can find improper neutralization of special element in command (misuse case), execution with unnecessary privileges, path traversal, uncontrolled format string, buffer copy without size checking, missing authentication or authorization, and hard-coded credential. These typical errors appear in the top 25 CVE vulnerabilities and can be prevented with respect of monster mitigation and best practices. Defensive code, “assume the worst” coding habits and comprehensive parser verification are the most efficient measures against random attacks [8]. It’s substantially infeasible for random testing to discover multistage exploit even guided with an intelligent mutation strategy. It can however give hints about the vulnerability entry point to then further exploit it.

4.3 Remote and automated test

A unique remote test infrastructure is appropriate in the context of massive and continuous deployment for a cross-matrix verification with multiple hardware variants, manufacturer configurations, firmware versions and calibration flavors. The security use, misuse, abuse and fuzz test suite can be applied systematically on any device of the test bed, even if it is not located on the same site. The test automation is the logical effect of a continuous deployment with an expanded shipment flow. A diagnostic report can even be required periodically for each device selected for the partial roll-out or dark launch.

5 Practical experiment

5.1 Tester description

Architecture overview.
All random testers use more or less the same identical iterative sequence: generate the input, apply the input to the target, and analyze the result. FDF tools haves a fourth operation to extract metrics used to guide the generation or mutation of future inputs.

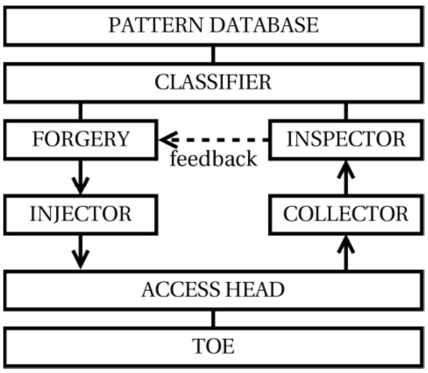


Fig. 1. Experimented tester architecture

The proposed architecture (**Fig. 1**) follows this common flow. The forgery generates challenge packets that are shaped and sent not directly to the TOE but to the intermediate access head by the injector module. The TOE is managed and monitored by this access head and the returned responses are encapsulated and sent to the collector module. Fine response scrutinizing is performed in the inspector module that can detect a faulty or suspicious response. Note that the forgery can be driven by some inspector results in case of FDF. The classifier module sits on the floor above to record and retrieve hit patterns and metadata according to product hardware and software identifiers.

Forgery and inspection capability overview.
The following feature list illustrates the commodities offered by the manifest grammar for challenge building and response monitoring:

- define a frame as stack of including successive layers
- define a layer as represented by a packet or a sequence of packets
- associate a packet with its PDL name
- control the evolution of a field and its malformation during the session
  - e.g. constant random, guided by FDL fitness, recursive mutation
- define conditional field control depending on another field value
- define a ordered or randomized sequence of possible mutation by iteration
- define a composition of chained mutation or generation in a single iteration
- define a packet profile for dissection of replies
- define a events for session management on response type
- define conditions to be asserted, possibly nested
- define events on condition assertion
  - log, file capture, session stop, restart all
- define the fields for which the value is collected for further statistical analysis
- declare analytics on defined collection e.g. histogram
- define the fitness function and the random field impacted in FDF
- define protocol states, the packet template per state, and conditions to move on state

Access head guidelines.
This architecture emphasizes the essential role of the access head that communicates to the tester engine by a standard socket connection, meaning the access head

can be located on the local host, or can be equally accessible through a LAN or a WAN. By definition, the access head is a transparent bridge between:

- a socket communication pipe connected to the injector and the collector
- and the physical channel of the target of evaluation

Each pattern generated by the forgery is forwarded to the target and the captured response from the target is forwarded to the collector for inspection. The choice of hardware and software components obviously depends on the physical link but also on possible objectives of quality conformance, if any. The target layers to be fuzzed have an important impact on the access head implementation. If the media access control layer is preserved, any types of commercial product can be enough, for example a WiFi or Bluetooth USB dongle. If error injection is applied on lower layers, then a setup where the MAC drivers can be bypassed is needed. For example, a Raspberry Pi coupled with a bare RF transceiver evaluation kit is convenient to test a wireless target. For all common wired serial links like UART, SPI, JTAG or I2C, Bus Pirate [24] is a perfect adapter. For NFC interfaces, commercial testers are available, but silicon vendors' evaluation kits like TRF7970A (TI) or CR95HF (ST) are well adapted. For the CAN interface, a wide selection of commercial offering of USB/CAN adapters is available on the market. If the error injection is needed at the lowest possible physical level, then a Software Defined Radio (SDR) should be considered [23]. The previous described solutions are enough for a laboratory environment but must be ruggedized for an industrial environment with quality assurance objectives.

5.2 Test configuration example

In the following, PROSE refers to the prototype implementation of this proposed architecture. It is an acronym for PRobe for Security Evaluation.

CONTIKI micro-kernel on WISMOTE.

The purpose of this experiment is to evaluate the tester on an IoT device. The candidate is the CONTIKI micro-kernel [25] embedded on a WISMOTE sensor node communicating through a 802.15.4 radio channel. The transit service is realized by the IPv6 Low-Power Wireless Personal Area Network (6LoWPAN) and the Low-Power and Lossy Networks routing protocol (RPL-RFC6550) [26][27]. The RPL border router is needed for the initial neighbour discovery and join procedure. Later on the periodic information messages to preserve the routing topology are exchanged between the border router and the sensor node.

CONTIKI distribution offers an interesting option to facilitate development and debug operation. The kernel image is built for a Linux or a Windows operating system. This platform named "native" allows to easily integrate a simulation environment based on a discrete network simulator, WSNET in that case. Three entry points are opened using posix pipes on the simulator instance: the native border router, the leaf target node and the tester access head. The equivalent test bench is realized with a Raspberry PI coupled with a Microchip MRF24J40MC radio add-on board. In that case, the connection to the tester can be done through a socket interface to the gateway implementing the border router and the access head.

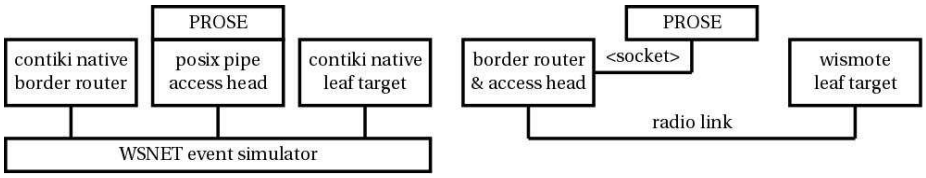


Fig. 2. Simulated and Embedded test platform

The fuzzing configuration builds a 802.15.5 Frame Control Field described by the *"Dot15d4FCS"* SCAPY class with the PAN identifier compression mode and long address, the IPv6 destination address being chosen to reach the target. The fuzzed layers are *"LoWPAN_IPHC"* , *"IPv6"* and *"ICMPv6EchoRequest"* . The inspection is simply configured to trace killer challenge detected by acknowledgment vanishing.
Three defects observed after a short session time from about ten minutes to one hour are listed below.

```
//sicslowpan.C
1482: memcpy(packetbuf_ptr + packetbuf_hdr_len, (uint8_t *)UIP_IP_BUF
+
1483: uncomp_hdr_len, uip_len - uncomp_hdr_len);
```

The term (uip_len - uncomp_hdr_len) can be negative resulting in a buffer overflow.

```
//uip6.c
881: uip_ext_opt_offset += UIP_EXT_HDR_OPT_PADN_BUF->opt_len + 2;
```

For extension header buffer operation, the execution enters in a never ending loop when the variable `UIP_EXT_HDR_OPT_PADN_BUF->opt_len` equals 254 for the killer challenge.

```
// uip-icmp6.c
212: memmove((uint8_t *) … ,
213 … , uip_len - UIP_IPICMPH_LEN - uip_ext_len -
UIP_ICMP6_ERROR_LEN);
```

When the term `uip_len - UIP_IPICMPH_LEN - uip_ext_len - UIP_ICMP6_ERROR_LEN` is negative, the memove can end in a stack overflow error.

Multiple Interface CPS Component.
The purpose of the next platform is to experiment a multiple physical channels injection. The target is a cyber physical system automation server communicating with two wired channels USB and CAN plus two wireless channels WiFi (WLAN) and ZigbePro (ZBP) The same transport layer encapsulates any application services on every physical channel. Four different access heads are deployed to inject erroneous challenges on all available interfaces simultaneously.

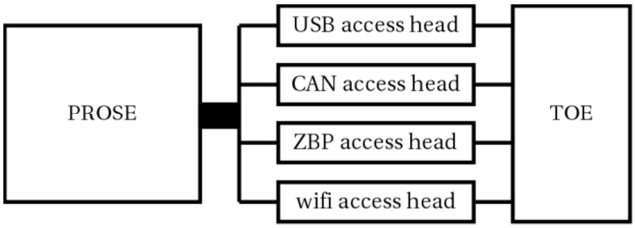


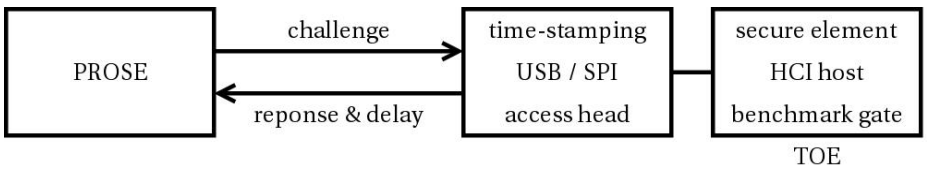
Fig. 3. Multiple channel fuzz testing platform

The fuzzed layers start from the media access control up to the application layer. The inspection is concentrated on suspicious returned error codes from valid replies, without forgetting to filter all the rational error codes e.g "invalid format" frame.

Two unexpected behaviors are observed. First, the intensive rate messaging leads to a memory overflow error code coupled with a significant response time degradation if the four interfaces are flooded simultaneously. This degradation is also observed on all valid messages and can only be fixed by performing a hardware reset. No defective states are observed if the interfaces are not stressed all four together. Secondly, a never ending loop occurs when an error is inserted in the ASN.1 Cryptographic Message Syntax [28] document, providing certificate and signature. This deviation occurs when the message length field is tampered with.

HCI smart-watch secure element.

The purpose of this platform is to prototype a FDF guided by statistical analysis. The target is a secure element of a wearable IoT smartwatch from the H2O project. The communication is based on a simple HCI over SPI protocol with a dedicated benchmark gate implemented in the target [29]. The access head is an USB to SPI repeater with a precise send/receive time stamping, returning the response message and the delay to the collector. The benchmark gate has N possible services. Each service has a different execution duration D_i. A command can activate K services among N. The total delay is the sum of the activated services execution time. Some services are easily visible because they have a very high likelihood of being activated by the incoming command. Other services are somehow hidden because they have a very low likelihood of being activated. The goal of the experiment is to confirm that the fuzzer can explore all the services equally despite their unequal chances of being discovered.



The inspection module is configured to accumulate response delays and to maintain their histogram. Each new egress message is affected with a score depending on the occurrence of its response delay. The score is inversely proportional to the probability density of the delay. The score is used by a non-linear distribution function generating new input values. The chosen distribution function, i.e. the Von Mises distribution, has configurable location and dispersion parameters. Hence, when a

score is high, the distribution is concentrated around the point of interest. When the score is low, the distribution gets closer to the normal distribution.

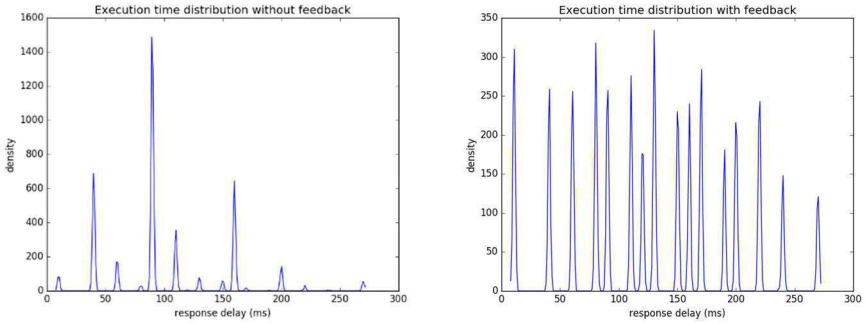


Fig. 4. Response distribution with and without feedback comparison

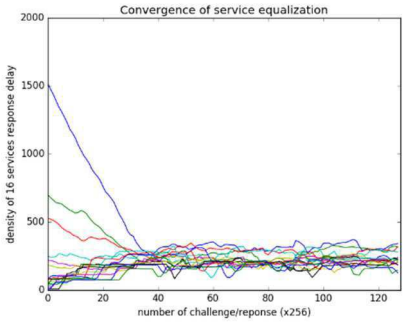


Fig. 5. Convergence factor of 16 services

The experiment is done with 4 possible services combining 16 possible response delays. The observed results confirm that if no feedback is applied, then the response distribution is uneven. The "dominant" services have the highest peaks and some "hidden" services are not visible (**Fig. 4** left). If the FDF feature is enabled, the services are equally invoked after some iterations (**Fig. 4** right). The convergence trend leading to an even distribution of service responses is shown on **Fig. 5**.

6 Conclusion and future works

Scanning and fuzzing are by nature systemic security tests adequate to the latest steps of the security development life cycle i.e. operation, integration and maintenance. Despite the heterogeneity of IoT devices in terms of physical interface, communication protocol, application domain, normative environment and extended life cycle, a unified security tester providing use, misuse, abuse cases as well as fuzzing operation has been demonstrated as being the most efficient. The major requirement resides in achieving sufficient modularity so that the target is easy to connect through any physical channels. According to a protection profile defining an adversary model with a low potential of physical access, this approach is proven low cost, fast, automated, scalable and easy to put in place. The remote property enables security evaluation laboratories as well as industrial stakeholders to share the same test environment. The adoption by security evaluation scheme and certification methodology would be a valuable perspective. In the future, the combination of scanning and fuzzing operation could be an interesting way to enhance the target coverage. On the other side, combining logical and physical error injection with automation could cover higher threat levels.

Acknowledgments

The author would like to thank the partners of the H2O -Human to Objects- project labelled for the CATRENE 7th call and referenced CAT209-H2O, covering more particularly secure element in IoT ecosystem as explained in paragraph **"HCI smart-watch secure element"**

References:

1. Clarke, E. M., Emerson, E. A., & Sifakis, J. (2009). Model checking: algorithmic verification and debugging. *Communications of the ACM*, *52*(11), 74-84.
2. Kirchner, F., Kosmatov, N., Prevosto, V., Signoles, J., & Yakobowski, B. (2015). Frama-C: a software analysis perspective. Formal Aspects of Computing, 27(3), 573-609.
3. Nethercote, N. (2004). Dynamic binary analysis and instrumentation (Doctoral dissertation, PhD thesis, University of Cambridge).
4. Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. Future Generation Computer Systems, 29(7), 1645-1660
5. Cervesato, I. (2001, June). The Dolev-Yao intruder is the most powerful attacker. In 16th Annual Symposium on Logic in Computer Science—LICS (Vol. 1)
6. Olsson, H. H., Alahyari, H., & Bosch, J. (2012, September). Climbing the" Stairway to Heaven"--A Mulitiple-Case Study Exploring Barriers in the Transition from Agile Development towards Continuous Deployment of Software. In 2012 38th Euromicro Conference on Software Engineering and Advanced Applications (pp. 392-399). IEEE.
7. Sindre, G., & Opdahl, A. L. (2005). Eliciting security requirements with misuse cases. Requirements engineering, 10(1), 34-44.
8. Dowd, M., McDonald, J., & Schuh, J. (2006). The art of software security assessment: Identifying and preventing software vulnerabilities. Pearson Education.
9. Sutton, M., Greene, A., & Amini, P. (2007). Fuzzing: brute force vulnerability discovery. Pearson Education.
10. Takanen, A., Demott, J. D., & Miller, C. (2008). Fuzzing for software security testing and quality assurance. Artech House.
11. Myers, G. J., Sandler, C., & Badgett, T. (2011). The art of software testing. John Wiley & Sons.
12. DeMott, J. (2006). The evolving art of fuzzing. DEF CON, 14.
13. McNally, R., Yiu, K., Grove, D., & Gerhardy, D. (2012). Fuzzing: the state of the art (No. DSTO-TN-1043). DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION EDINBURGH (AUSTRALIA).
14. Abdelnur, H., Lucangeli, O. J., & Festor, O. (2010). Spectral Fuzzing: Evaluation & Feed-back (Doctoral dissertation, INRIA).
15. Aitel, D. (2006). MSRPC fuzzing with SPIKE 2006. Immunity Inc, August.
16. Devarajan, G. (2007, August). Unraveling SCADA protocols: Using sulley fuzzer. In Defon 15 Hacking Conf.
17. Banks, G., Cova, M., Felmetsger, V., Almeroth, K., Kemmerer, R., & Vigna, G. (2006, August). SNOOZE: toward a Stateful NetwOrk prOtocol fuzZEr. InInternational Conference on Information Security (pp. 343-358). Springer Berlin Heidelberg.
18. Godefroid, P., Levin, M. Y., & Molnar, D. (2012). SAGE: whitebox fuzzing for security testing. Queue, 10(1), 20.
19. Campana, G. (2009). Fuzzgrind: an automatic fuzzing tool. Hack. lu.
20. BIONDI, P. (2005). Scapy: explore the net with new eyes. Technical report, EADS Corporate Research Center, http://www. secdev. org.
21. Combs, G. (2007). Wireshark. Web page: http://www.wireshark.org 12-02.
22. DeMott, J., Enbody, R., & Punch, W. F. (2007). Revolutionizing the field of grey-box attack surface testing with evolutionary fuzzing. BlackHat and Defcon.
23. Dillinger, M., Madani, K., & Alonistioti, N. (2005). Software defined radio: Architectures, systems and functions. John Wiley & Sons.

24. Bus Pirate from http://dangerousprototypes.com/docs/Bus_Pirate
25. Dunkels, A., Gronvall, B., & Voigt, T. (2004, November). Contiki-a lightweight and flexible operating system for tiny networked sensors. In Local Computer Networks, 2004. 29th Annual IEEE International Conference on (pp. 455-462). IEEE.
26. Kushalnagar, N., Montenegro, G., & Schumacher, C. (2007). IPv6 over low-power wireless personal area networks (6LoWPANs): overview, assumptions, problem statement, and goals (No. RFC 4919).
27. Hui, J. W. (2012). The routing protocol for low-power and lossy networks (rpl) option for carrying rpl information in data-plane datagrams.
28. Housley, R. (1999). Cryptographic message syntax.
29. ETSI, T. (2008). ETSI TS 102 622, smart cards; UICC— contactless front-end (CLF) interface; host controller interface (HCI). Technical specification.

Hit the KeyJack: stealing data from your daily wireless devices incognito

Guillaume Fournier[1], Pierre Matoussowsky[1], Pascal Cotret[1,2]

[1] CentraleSupelec - Rennes, France
guillaume.fournier@supelec.fr, pierre.matoussowsky@supelec.fr

[2] IETR, SCEE Research Team - Rennes, France
pascal.cotret@centralesupelec.fr / @Pascal r2

Abstract. Internet of Things (IoT) is one of the most fast-growing fields in high technologies nowadays. Therefore, lots of electronic devices include wireless connections with several communication protocols (WiFi, ZigBee, Sigfox, LoRa and so on). Nevertheless, designers of such components do not take care of security features most of the time while focusing on communication reliability (speed, throughput and low power consumption). As a consequence, several wireless IoT devices transmit data in plaintext creating lots of security breaches for both eavesdropping and data injection attacks. This work introduces KeyJack, a preliminary proof-of-concept of a solution aiming to eavesdrop wireless devices and hopefully perform injection attacks afterwards. KeyJack operates on widely-used devices: our keyboards! This solution is based on low-cost embedded electronics and gives an attacker or a white hat hacker the possibility to retrieve data from John Doe's computer. This work also shows that this approach could be used to any wireless device using 2.4GHz radio chips like the NRF24L01 from Nordic Semiconductor.

1 Introduction

Nowadays, most of the people have IoT devices at home or even at work: for instance, it could be included in a fridge, a set-top box or computers. IoT is a highly-growing field and it will get even bigger in the next decade (a Verizon report [21] foresees that the IoT market will hit the $1 trillion limit by 2019). In the same document, it can be seen that the IoT market targets several applications such as healthcare and home monitoring. Most of these connected devices were designed to perform tasks fast and in a cheap way (in other words, communication links have to be fast and low-power). Unfortunately, without security, each device is a vector of threats: malevolent people could use breaches in wireless protocols to steal or inject data using man-in-the-middle (MITM) attacks.

In this context, this work focuses on wireless keyboards that are widely-spread components. This work presents KeyJack that is a kind of wireless key-logger implemented in a tiny electronic board aiming to retrieve data from a remote computer. Furthermore, this work gives some clues about using it as a malicious injection device.

Section 2 presents some related works of both classic keyloggers and embedded electronics solutions with similar goals. Section 3 presents KeyJack, its requirements and implementation details. Then, Section 4 presents an eavesdropping scenario where Keyjack is used and gives some hints about the feasibility of injection attacks using such hardware components. Finally, Section 5 presents some conclusions and perspectives about this work.

2 Related works

2.1 Keyloggers

When a hacker wants to retrieve data from a user keyboard, keyloggers could be used [18,19,12,5]: these components are usually softwares running in the back-ground of the target computer. In more recent works such as Damopoulos et al. [4], keyloggers are also implemented for touchscreen that are widely-spread interfaces on our smartphones and tablets. Such keyloggers aim to keep a copy of each keyboard hit made by the victim. On the other side, several works such as [10,17] present countermeasures in order to implement systems resilient to such attacks. Therefore, IoT designers could imagine to create embedded systems immune to standard keylogger implementations.

However, most of software keyloggers are not so easy to use:

- Advanced features are rarely included in free versions. As keyloggers may be used for "bad" purposes, developers keep the most intrusive options for paid licenses.
- Basic keyloggers are not 100% discrete as they appear in the task manager. For some of them, administrator rights may be required by the operating system (for instance, installing a driver).
- Furthermore, results take up space on the target computer and may increase its power consumption.

2.2 Other interesting works

In the context of pure keyloggers, there are other interesting alternatives in the hardware community:

- KeyGrabber USB made by KeeLog [20].
- USB Rubber Ducky, a USB tool by Hak5 [9]. This USB key includes a 60MHz programmable microcontroller and a SD slot. It behaves like a keyboard: therefore, nearly anything can be performed (from a Rick Roll hack to keyloggers as well).

Both solutions look like USB ash drives: it can be easily hidden on a computer port. Even if some countermeasures exists (for instance, KeyScrambler[14] encrypts of all keyboard hits in Firefox), it is not 100% satisfying as it still needs some physical access to the target. Furthermore, even if such a tool may be hidden in the task manager, it is assumed that its power consumption may be revealed with physical measurements.

KeyJack wants to tackle those problems using an alternative breach: nowadays, most keyboards are wireless and may be affected by eavesdropping and MITM attacks. The next subsection presents some works using wireless connection to reveal security breaches.

2.3 Embedded electronics solutions

There are several works aiming to steal information from wireless computer devices. The most "industrial" solution is MouseJack from Bastille Networks[1]. MouseJack is an exploit used in several wireless (non-Bluetooth) keyboards that can be used to perform eavesdropping and relay attacks. However, a laptop is required to

[1] https://www.mousejack.com/

run MouseJack in contradiction to KeyJack which aims to be a discrete and standalone solution [11].

Other people tried to make things smaller. Digital Security (@iotcert) implemented related eavesdropping methods for Bluetooth devices running on a single-board computer such as Raspberry Pi [2,3]: Cauquil et al. implemented a Bluetooth sniffer on a RaspberryPi Zero single-board computer that can be used to track people and steal secrets (as it was shown in their Nuit du Hack'16 talk[1]).

Samy Kamkar (@samykamkar) developed Keysweeper [13,7], a solution based on a small tiny microcontroller similar to Teensy or Arduino Nano. Even if this solution is the most similar to KeyJack, it does not take into account the feasability of data injection (Kamkar only focused on data listening). Furthermore, KeyJack plans to use a GSM chip to perform multiple eavesdropping with target localization in a use case where several KeyJack devices would be implemented.

3 KeyJack

3.1 Threat model

There are several keyboard manufacturers. This work focuses on the two main ones: Microsoft and Logitech (it is assumed that generic/low-cost keyboards may work as well). Microsoft/Logitech keyboards uses a classic Wifi-based protocol working at 2.4GHz (for European versions at least). When this 2.4GHz link is left unencrypted, a classic MITM scheme could be used as shown in Figure 1.

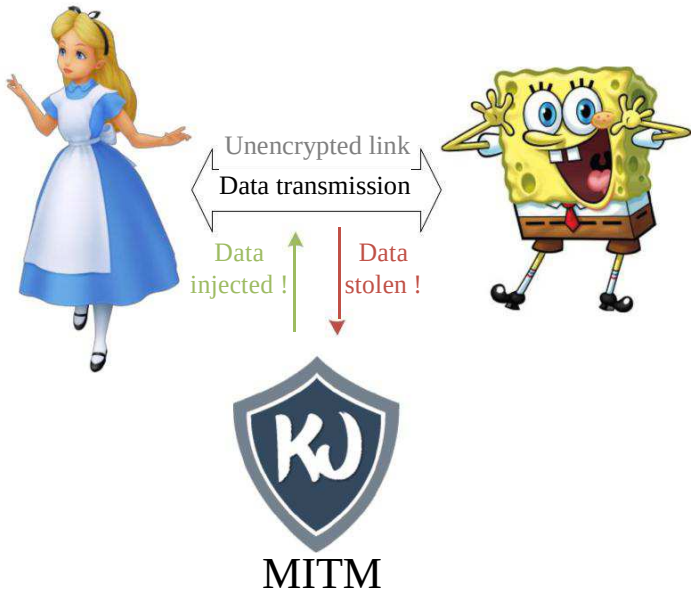


Fig. 1. MITM scheme with KeyJack as the eavesdropper/injection device

For some recent keyboards, the communication between Alice and Bob is encrypted: even if such schemes may be broken using brute force or other advanced techniques, this work assumes that the link is left in plaintext:

– For basic models, security was not implemented in the wireless protocol.

[1] http://virtualabs.fr/ndh16/ndh16-mass-pwning-bug.pdf

– For future perspectives, KeyJack could be adapted to other plaintext protocols working at different frequencies (5GHz band, for instance).

3.2 Requirements

A KeyJack device must have the following requirements:

– No physical access to the target device/computer. It means there will not be any USB connection or malicious software installed.
– Tiny implementation: in other words, KeyJack must be a small-sized board, easily transportable and autonomous in terms of energy.
– "Tracking-friendly": KeyJack end-users must be able to get eavesdropping results from a remote device (website, smartphone. . .).
– Injection-enabled: this work aims to propose a solution which enables not only eavesdropping but also data injection (from a remote interface as well).
– This work focuses on a Microsoft Wireless Keyboard 8001. Mainly because its protocol was unecrypted.
– For further implementations, a GSM chip in order to retrieve locations of KeyJack nodes in case we want to install a network of such devices.
– And, of course, something that is low-cost and easily reproductible!

3.3 Hardware implementation

KeyJack components are shown in Figure 2:

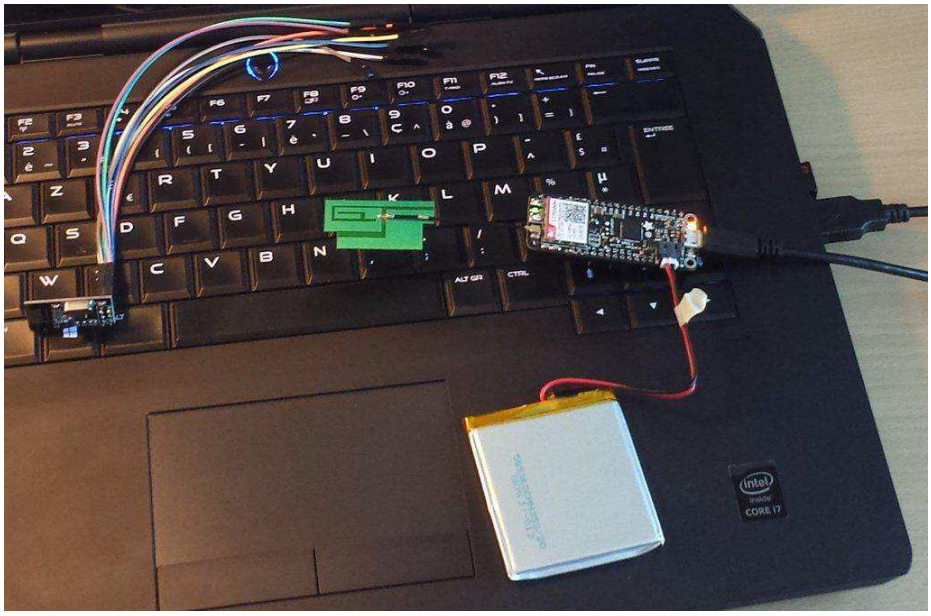


Fig. 2. Electronic components used in KeyJack

From the left to the right side:

– A 2.4GHz board based on a NRF24L01 chip from Nordic Semiconductors.
– An antenna.
– Adafruit Feather FONA as the microcontroller board: this board includes, in a tiny form factor, an ATmega32u4 running at 8MHz and a GSM chip.

[1] https://www.microsoft.com/accessories/en-gb/products/keyboards/ wireless-keyboard-800/2vj-00006

– And a battery.
– (the USB cable is just here for programming purposes)

3.4 Software layer

Figure 3 shows an example of a KeyJack network. Each node is the Adafruit platform described in the previous section running a given Arduino code. When each node collects information, it is transmitted to the self-hosted server where an internal website was developed.

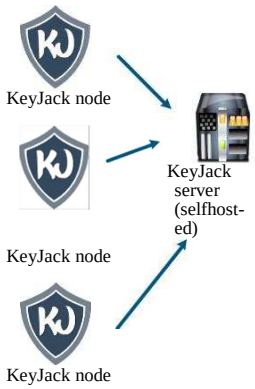


Fig. 3. Example of a KeyJack network with three nodes and a server

On the server side, KeyJack interface looks as follows:

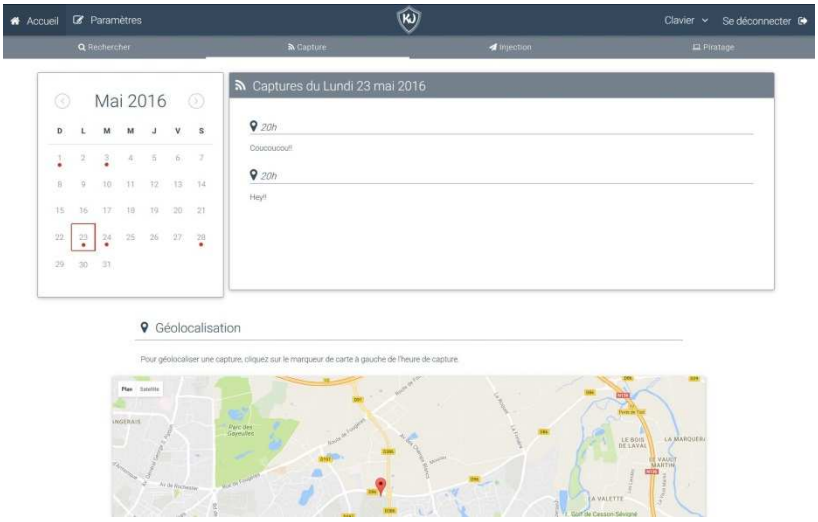


Fig. 4. KeyJack server interface

4 Case study: eavesdropping a Microsoft keyboard

As it was said in Section 3, this case study is focused on a Microsoft Wireless Keyboard 800. Furthermore, we only used a single KeyJack node as shown in Figure 2. When the user enters the KeyJack server, an interface as shown in Figure 6 appears.

Each keyboard is identified by its MAC address and has a dedicated menu:

– *Search*. In this part, the user can read logs of former measurements.
– *Capture*. Reading captures filtered by their date.
– *Injection*. Transmitting keyboard keys.
– *Hacking*. This last tab allows to launch attack scripts.

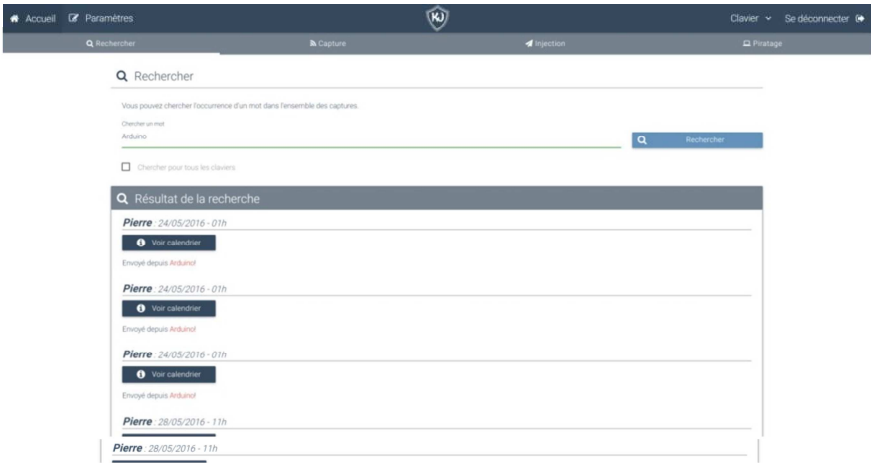


Fig. 5. KeyJack keyboard interface

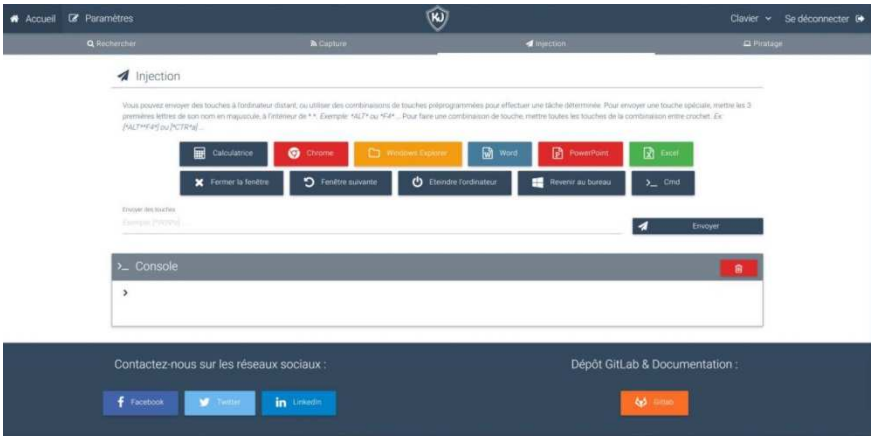


Fig. 6. KeyJack injection interface

Normally, the NRF24L01 chip is not able to work as a sniffer: in fact, the target MAC address is needed and it is not possible to scan the frequency spectrum around 2.4GHz to find one. However, as Samy Kamkar explained in its Keysweeper project, we can send fake information about the MAC address in order to swindle the wireless chip.

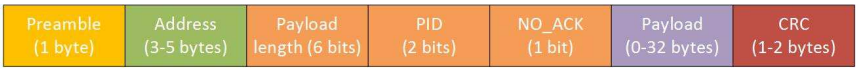


Fig. 7. NRF24L01 packet structure

With previous works of Travis Goodspeed, Samy Kamkar discovered that if we enter an incorrect value regarding the MAC address size (writing the preamble itself), the upset NRF24L01 chip considers all preambles as the target MAC address. However, all data after the MAC address should be the payload. Therefore, we get all traffic packets with the MAC address and everything else up to the CRC. From there, we can proceed to keyboard detection.

Each brand has its own protocol to deal with the USB dongle on the computer. Microsoft does not encrypt data sent by its keyboards (it is only done since 2015!). The only security measure is a XOR performed on the payload with the MAC address. As we know how to get MAC address, this is not a problem for us. As a consequence, we only have to detect Microsoft protocols and perform a XOR.

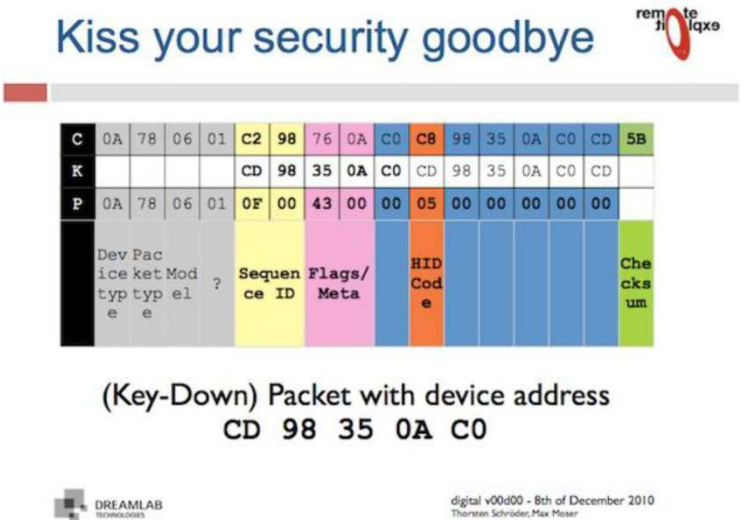


Fig. 8. Packet structure

Samy Kamkar also discovered that all Microsoft keyboards used a MAC address beginning with 0xCD (this is the only byte we need in further measurements). In fact, Microsoft is done in a way that the key value is in the 10th position. As the MAC address is 5 bytes wide and 4 first bytes are not encrypted, the key value is always XORed with 0xCD.

The last criteria we use to detect Microsoft keyboards is the value of the first unencrypted bytes. As it is shown in Figure 8, the device type is always 0x0A (for keyboards). Then, the packet type indicates the key category (we only focus on keystrokes or idles). Related codes are 0x78 and 0x38. We are finally ready to scan, for all frequencies from 2403MHz to 2480MHz:

- We check if the MAC address of the transmitter begins by 0xCD.
- We test if the payload begins by 0xA78 or 0xA38.

5 Conclusion and perspectives

This work presented KeyJack, a low-cost solution for basic eavesdropping of a specific Microsoft wireless keyboard. The proof-of-concept is a standalone device which can be left in any open-space and small enough to be hidden from people sight. Even if this study focuses on a specific keyboard model, there are opportuni-

ties with other models/vendors when communications are not encrypted. Furthermore, KeyJack can be easily modified for data injection as the server side is already implemented. As each keyboard is clearly identified, the next perspective is to make a network of KeyJack node and a single server where we could monitor everything from a remote location. Finally, KeyJack may be adapted for other protocol where security matters in the context of Internet of Things.

References

1. Bastille Networks, L.: Mousejack faq (Feb 2016), https://www.bastille.net/sites/all/themes/professional_theme/media/bastille_mousejack_faq_2_12_16-2.pdf
2. Cauquil, D.: Bluetooth low energy device spoo_ng library (Aug 2016), https://github.com/DigitalSecurity/mockle
3. Cauquil, D.: Btlejuice: The bluetooth smart mitm framework (Aug 2016), https://speakerdeck.com/virtualabs/btlejuice-the-bluetooth-smart-mitm-framework
4. Damopoulos, D., Kambourakis, G., Gritzalis, S.: From keyloggers to touchloggers:Take the rough with the smooth. Computers & Security 32, 102 { 114 (2013),http://www.sciencedirect.com/science/article/pii/S0167404812001654
5. Dorne, M.K.: La menace keylogger (Jul 2016), http://korben.info/la-menace-keylogger.html
6. Express, J.: Vous utilisez des objets connectés? gare à vos données (Jul 2016), http://lexpansion.lexpress.fr/high-tech/vous-utilisez-des-objets-connectes-gare-a-vos-donnees_1813593.html
7. Goodin, D.: Meet keysweeper, the 10usd usb charger that steals mskeyboard strokes (Jan 2015), http://arstechnica.com/security/2015/01/meet-keysweeper-the-10-usb-charger-that-steals-ms-keyboard-strokes/
8. Hacker, B.: La sécurité dans l'internet des objets (Jul 2016), http://www.leblogduhacker.fr/la-securite-dans-internet-des-objets/
9. Hak5: Usb rubber ducky wiki (Jan 2016), http://usbrubberducky.com/#!index.md
10. Herley, C., Florencio, D.: How to Login from an Internet Cafe Without Worrying about Keyloggers . Association for Computing Machinery, Inc (2003)
11. Higgins, K.: Mousejack attack bites non-bluetooth wire-less mice (Feb 2016), http://www.darkreading.com/endpoint/mousejack-attack-bites-non-bluetooth-wireless-mice/d/d-id/1324404
12. Holz, T., Engelberth, M., Freiling, F.: Learning More about the Under-ground Economy: A Case-Study of Keyloggers and Dropzones, pp. 1-18. Springer Berlin Heidelberg, Berlin, Heidelberg (2009), http://dx.doi.org/10.1007/978-3-642-04444-1_1
13. Kamkar, S.: Keysweeper (2015), http://samy.pl/keysweeper/
14. KeyScrambler: Keyscrambler's features (Jan 2016), https://www.qfxsoftware.com/ks-windows/features.htm
15. Lifchitz, R.: Zigbee security review of a famous french set-top box (Jul 2016), https://speakerdeck.com/rlifchitz/zigbee-security-review-of-a-famous-french-set-top-box
16. Moine, F.: Huit bonnes pratiques pour bien sécuriser les objets connectés (Jul 2016), http://www.journaldunet.com/solutions/expert/64817/huit-bonnes-pratiques-pour-bien-securiser-les-objets-connectés.shtml
17. Ortolani, S., Crispo, B.: Noisykey: Tolerating keyloggers via keystrokes hiding.In: Presented as part of the 7th USENIX Workshop on Hot Topics in Security.USENIX, Berkeley, CA (2012), https://www.usenix.org/conference/hotsec12/workshop-program/presentation/Ortolani
18. Sagiroglu, S., Canbek, G.: Keyloggers. IEEE Technology and Society Magazine 28(3), 10{17 (Fall 2009)
19. Subramayam, K., Frank, C., Galli, D.: Keyloggers: The Overlooked Threat to Computer Security. The Vespiary (2003)

20. USB, K.: Hardware keylogger - keygrabber usb (Jan 2016), https://www.keelog.com/usb_hardware_keylogger.html
21. Verizon: State of the market: The internet of things 2016 (Jan 2016), https://www.verizon.com/about/sites/default/files/state-of-the-internet-of-things-market-report-2016.pdf

IoT and Physical Attacks

Helene Le Bouder[1], Jean-Louis Lanet[1], Ronan Lashermes[1], Thierno Barry[2], and Damien Courousse[2]

[1] LHS-PEC TAMIS INRIA Campus Beaulieu 35000 Rennes, France (contact: helene.le-bouder@inria.fr)
[2] Univ. Grenoble Alpes, F-38000 Grenoble, France; CEA, LIST, MINATEC Campus,F-38054 Grenoble, France

1 Introduction

The Internet of Things (IoT) is about to become ubiquitous in our work and home environments. IoT is the trending term to describe the ubiquity of devices with a computing system in it. From smart light-bulbs to CCTV (video surveillance), from thermostats to wearable heart rate monitors, etc. The emergence of IoT introduces new devices massively interconnected as intelligent houses, connected cars, medical devices... IoT must guarantee the protection and security of personal data and that it does not constitute a new source of threats. It must respect the current and future safety requirements. Physical aspect can lead to forget that some connected objects are information systems. Constraints in resources often leave little place for the implementation of security mechanisms.

When discussing about the security of devices, generally we think about cryptography and/or password. So it is usual to think that to protect a connected object, the designer just has to add a PIN code or to encrypt data. Yet an IoT device has defining features that expose it to new threats, namely the attacker can physically take it and put it into her lab. Therefore, specific knowledge is required to correctly design the IoT device in order to resist to physical attacks.

In this contribution, a quick state of the art of physical attacks is given in 2. Then the particularly case of PIN codes is presented in 3. Countermeasures are presented in 4 to defend the system. Finally the conclusion is drawn in section 5.

2 Physical Attacks

The threat of physical attacks arises when the attacker has physical access to the device.

2.1 State of the art

When discussing about the security of a cryptographic algorithm, numerous mathematical tools allow the cryptographers to ensure the security of a cipher.

Unfortunately those tools do not consider the interaction of the computing unit with its physical environment. Physical attacks are a real threat, even for cryptographic algorithms proved secure mathematically. Physical attacks are divided in two families: the Side-Channel Attacks (SCA) and the Fault Injection Attacks (FIA). Obviously, physical attacks assume a physical access to the device.

SCA are based on observation of the circuit behaviour during the computation, for example in [1]. Side-channel attacks exploit the fact that some physical values of a circuit depend on intermediary values of the computation. This is the so-called leakage of information of the circuit. The most classic leakages are timing [2, 3], power consumption [4, 5] and electromagnetic emissions (EM) [6].

Fault Injection Attacks consist in disturbing the circuit behaviour in order to alter the correct progress of the algorithm [7]. Faults are injected into a device using various means such as laser [8], clock glitches [9], spikes on the power supply or electromagnetic perturbations [10].

The current trend is to mix the two kinds in order to build hybrid attacks, as for example in [11, 12].

Physical attacks are threats for all standard cryptosystems like DES [1], AES [7], RSA [13] or cryptography based on ECC or Pairing [14]. They are also able to reverse engineer them [15-17].

More and more complex devices are attacked from the simplest to desktop computers [13].

It is difficult to anticipate physical attacks. That is why, many frameworks [18 - 24] try to describe them.

2.2 Faustine and EMA

In the High Security Laboratory (LHS) of INRIA, in the TAMIS team, we have at our disposal two experimental benches Faustine and EMA (shown on figures 1 and 3) to investigate vulnerabilities with respect to physical attacks.

Faustine, is the EM fault injection bench.

An electromagnetic pulse is sent to the target thank to the inductive coupling of an EM probe with the target metal layers. The pulse must be precise: it must occur at the moment when a fault is desired and it must be short as to not interfere with other operations on the chip. Our bench is able to inject a pulse of ≈ 3 ns at the minimum and to repeat this pulse in order to achieve multi-faults if wanted.

EMA is the LHS electromagnetic emissions analysis bench. This bench is composed of an EM probe from Langer (RF-R0,3-3) to capture the leakage, a preamplifier from Langer (PA 303) and several oscilloscopes to measure it. The first oscilloscope is our "low cost" oscilloscope. It is a 3405A Picoscope (USB oscilloscope) with 8-bit resolution, a 1GS/s sampling rate and 100MHz higher cut-off frequency. The second oscilloscope is the “high end” one: a DSOS404A from Keysight. This one achieves 10-bit resolution with a 20GS/s and 4GHz bandwidth. This tool becomes necessary to measure leakage from fast targets.

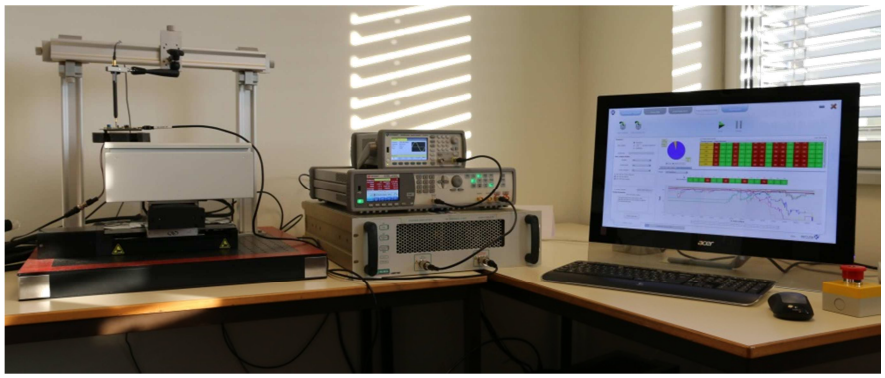


Fig. 1: Faustine, the EM fault injection platform.

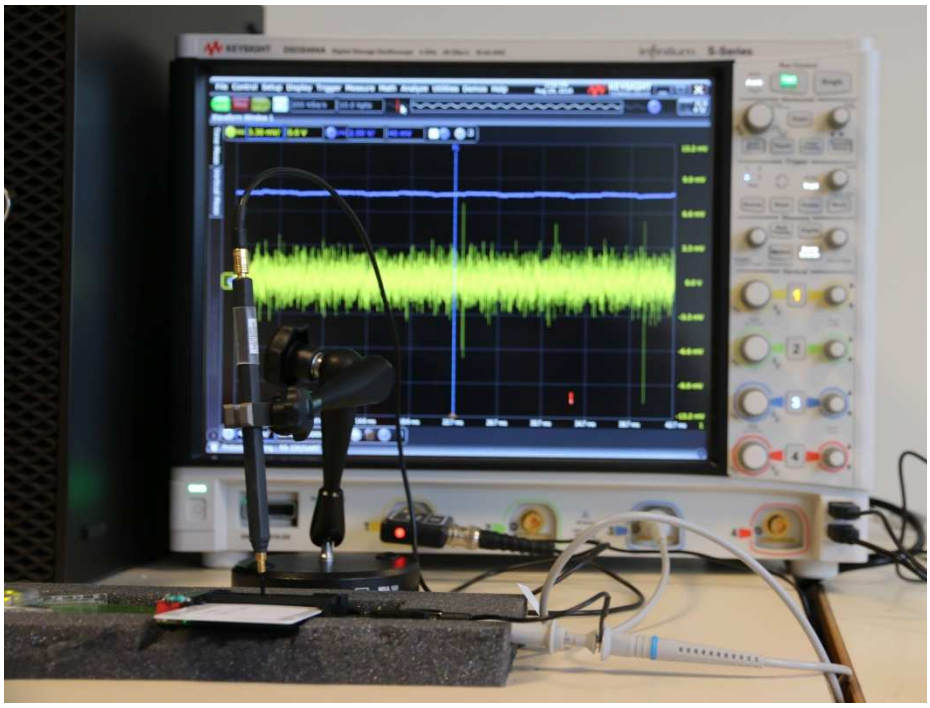


Fig. 2: EMA, the EM leakage observation platform.

Finally a control computer is used to orchestrate the measurements and perform the analysis, with home-made tools. The analysis example described in section 3 was done with the low cost platform (with picoscope). Excluding the computer, the hardware necessary to mount this attack is under 2000€.

The experimental bench protocol is summarized in Fig. 3

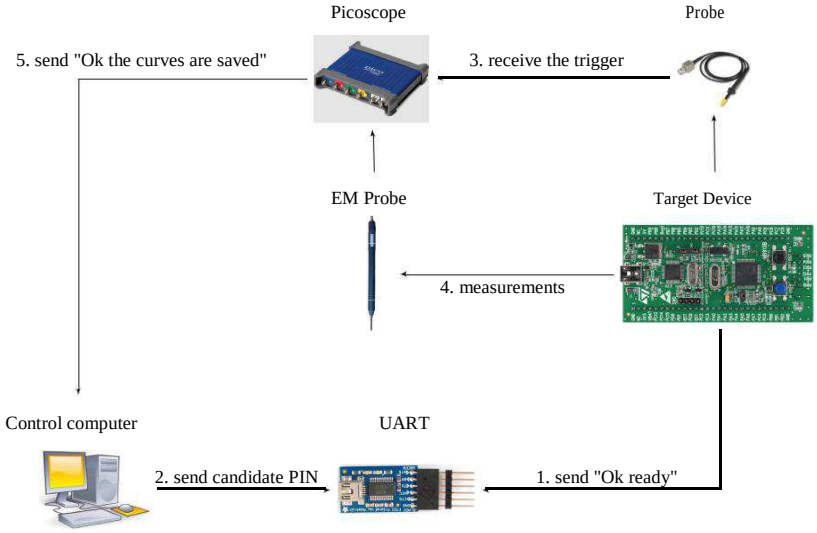


Fig. 3: Experimental bench

2.3 Targeting IoT devices

All IoT devices have a computing system: a microcontroller or a low end processor, sometimes with an internet connection. Because they derived from originally offline, and/or restricted access systems, the security level of these devices has not caught up with their new exposition. Proof is the emergence of IoT botnets, often created by taking control of the devices with the default login/password pair. In [25], the authors show the vulnerabilities of most security protocols for wireless sensor networks. In [26], the authors show that the standard IEEE 802.15.4 used by a variety of higher-level protocol in IoT is not protected against power analysis.

In the High Security Laboratory (LHS) lab we investigate a second class of threats: IoT devices may be captured by an adversary and therefore must sustain physical attacks.

Our target of choice, analyzed in section 3 is the STM32VLDISCOVERY board. Our application is implemented on an ARM-based STM32F100RB microcontroller embedding a Cortex-M3 core and running in our case at 24MHz. This chip does not embed any hardware countermeasures against side channels but it is a popular choice for IoT applications.

3 Side channel Analysis on PIN code

In the LHS lab we are interested in vulnerabilities of PIN codes with respect to side-channels. A new attack against PIN code was presented to Secrypt 2016 [27] and these results are chosen for illustration.

3.1 PIN code

In many smart cards, a Personal Identification Number (PIN) code is used to authenticate the user. These PIN codes are used, for example, in payment cards or SIM cards. Hence they are targets of choice for malicious adversaries. The main protection in PIN code comes from in the fact that the user has a limited number of trials. So it was believed to be protected against side-channel attacks, since an at-

tacker can exploit only a few EM traces; therefore most protections are against fault injection attacks.

A PIN code is an array of *m* digits. A True PIN is embedded in the device; and a Candidate PIN is proposed by the user. Here, the different values on the PIN code are defined in $[[0, 9]]^m$. The Verify PIN is the algorithm which test if the Candidate PIN is correct or not. A good countermeasure against fault attack is to compare true and Candidate PIN twice.The implementation of the PIN comparison is developed in 1, it is the version proposed by [28].

Algorithm 1 Comparison of two PIN codes

```
1:  procedure Comparison(candidate PIN V , true PIN U)
2:      status = FALSE
3:      diff  = FALSE
4:      fake = FALSE
5:      for b = 0 to 3 do
6:          if U_b ≠ V_b then
7:              diff  = TRUE
8:          else
9:              fake = TRUE
10:         end if
11:         if (b = 3) and (diff  = FALSE) then
12:             status = TRUE
13:         else
14:             fake = TRUE
15:         end if
16:     end for
17:     return status
18: end procedure
```

3.2 Description of the attack

The ultimate goal of our work was to test the resistance of a Verify PIN against a SCA with only few traces. That is why we were interested in template attacks. To implement a template attack, a pair of identical devices is needed. One is called the profiling device, the attacker has full control of it; and the other is the targeted device. It is supposed that an attacker can:

- obtain one (or a few) trace on the targeted device;
- change the True PIN in her profiling device;
- obtain many traces on her profiling device.

The attack is divided in two phases:

- profiling phase,
- attack phase.

The attacker starts by several measurement campaigns on the profiling device. Template attacks use a divide and conquer approach, the digits of PIN code are attacked separately. The digits of the True PIN code take all values k in [[0, 9]] and the other digits stay to zero. Digits of Candidate PIN code are fixed to a chosen value *v*. For each (*k, v*) many traces are collected: $M_{v,k} = \{xk_{(i,j)}\}$, *i* for trace, *j* for time.

The detection of points of interest selects the moment of computation for the comparison between the two PIN codes.

The first step is to compute the covariance matrix $S_{v,k}$, a square matrix of size $p \cdot p$. The elements of $S_{v,k} = \{sk_{(j,j')}\}$, are computed with equation:

$$sk_{(j,j')} = \frac{1}{n-1} \cdot \left(xk_j - \overline{xk}_j\right)^t \left(xk_{j'} - \overline{xk}_{j'}\right)$$

The attack phase works on the targeted device and starts by collecting a few traces $T_v = \{x_j\}$ The True PIN digit is **unknown**, it is the target; Candidate PIN digit is equal to *v*.

Then the attacker confronts the trace T_v to the template matrix $S_{v,k}$, with the general formula in template attack:

$$F_v\left(T_v | S_{v,k}, \overline{xk}\right) = \frac{1}{\sqrt{2\pi^p \cdot |S_{v,k}|}} \cdot \exp\left(-\frac{1}{2} \cdot \left(T_v - \overline{xk}\right) \cdot S_{v,k}^{-1} \cdot \left(T_v - \overline{xk}\right)^t\right)$$

The attack returns the guess k_v for which F_v is maximal for a given T_v, or rank the guesses *k* according to the value of $F_v(T_v, k)$.

3.3 Results

The Verify PIN algorithm is implemented on an ARM-based STM32F100RB microcontroller embedding a Cortex-M3 core and running in our case at 24MHz. The bench used is the low cost bench of EMA as explain in 2.2. The bandwidth of the measurement setup is limited by the Picoscope with an upper frequency of 100MHz. The traces obtained are composed of p = 3700 points, with a 1GS/s sampling rate. We remind that 100MHz is a cutoff frequency marking an *attenuation* of the relevant signal, not its total removal.

The experience shows an important result: if a Candidate PIN digit is equal to the corresponding targeted True PIN, it is always successfully detected. In the other cases the success rate is 20%. Improvements of the attack are still possible. To conduct a full attack, first the attacker sends a Candidate PIN with all digits to 0. The analysis is able to detect if a True PIN digit is 0 and will return the most likely value otherwise. Then she tests the PIN code returned by the first attack. The process is then repeated until all values are found. In the worst case: in 8 trials the PIN code is retrieved, as shown in the Table 1.

digit of traces:		1	2	3	4	5	6	7	8
n = 100000	1 Comparison	27:70	41:47	53:84	63:99	73:07	81:33	88:51	100
	2 Comparison	31:71	46:56	57:82	67:76	76:63	84:36	90:68	100
n = 200000	1 Comparison	29:28	44:27	56:79	67:41	76:66	83:91	90:68	100
	2 Comparison	32:72	49:52	61:96	72:05	80:49	87:53	93:23	100

Table 1: Success rate to retrieve a digit of True PIN according to the size *n* of the templates and the number and the choice of traces

The attack succeeds with very few traces. The True PIN is retrieved in 8 trials at most. It becomes a new real threat, and it is feasible on a low cost and portable platform. Some protections against fault attacks introduce new vulnerabilities.

4 Countermeasures

Fortunately countermeasures exist against physical attacks. Some are designed against side channel attacks other against fault injection analyses. Some are generic and may protect various applications with only minor tweaks. But specific countermeasures can always be devised specifically for an application.

4.1 Generic hardware countermeasures

Generic countermeasures against SCA include masking: instead of handling a secret value directly, a random mask is created and the chip handle the masked value (function of the secret and the mask) on one side, and the mask on the other side. The two results are finally recombined after the critical operations to get the result (example in figure 4).

f is a linear function which leaks information, *s* a secret and *m* a random value. In order to compute *f(s)* without leakage, compute:

$$s' = s + m$$
$$f(s) = f(s') - f(m)$$

Fig. 4: Simplest masking example

Another possibility is to use of desynchronization. Most analyses accumulate executions to reduce the measurement noise. If the measurements are desynchronized, reducing the noise becomes much harder and the attacker must increase the number of necessary executions.

Against fault attacks, generic solutions are based on redundancy principles. To detect a single fault, compute a function twice and check that the two results are identical. Variations on the redundancy theme are possible, one may use error detecting/correcting codes to achieve higher security (examples are show in figure5).

f is a function that must not be faulted, s a secret.

1. Twice the same computation $f(s) == f(s)$,
2. use the reciprocal $s == f^{-1}(f(s))$,
3. error detecting code $s' = encode(s)$, $decode(f'(s'))$ is valid?

Fig. 5: Simple redundancy examples

The protections are valid with respect to a threat model, for instance the example in Fig 4 supposes that the attacker can measure only one leakage. An attack would be possible otherwise (the so called second order attack).

Often, countermeasures against faults introduce vulnerabilities with respect to side-channels. To compute twice the leaking function provides twice the data for the SCA. The opposite is also true, masking schemes introduce a new attack surface with respect to fault attack: now faults can also be injected on masks.

All protection mechanisms have an overhead. Therefore a good protection is a trade-off in term of resources, threat model that can be sustained, vulnerability to FIA and SCA, etc.

4.2 Generic software countermeasures

Generally the best solutions against physical attacks are hardware protections. But IoT devices hardly have dedicated hardened chips. If a device was not built to ensure security, is it possible to still protect it? It is the subject of ANR project Cogito [29], among others, where authors use polymorphic codes to introduce software countermeasures against physical attacks. This polymorphism is a software solution and it is based on masking, noise addition, register shuffling... The authors show that the use of code polymorphism is an effective solution to improve security, while complying with the computing and memory constraints of embedded devices, and is easily generalizable to a large set of embedded computing platforms.

4.3 Specific countermeasures for verify PIN

Apart from generic countermeasures, it is possible to be protected against the presented attack by modifying how the PIN codes (true and candidate) are used.

Assuming that the device possess a dedicated secret key K, when setting the true PIN code U, the value $S = HM\,AC_K\,(U)$ is stored (instead of U) on the device. To verify the candidate PIN V , the device must compare $HM\,AC_K\,(V\,)$ and S. This solution offers several security features: the secret U is never handled directly (in verify PIN), it becomes impossible to use a divide-and-conquer strategy to recover individual bytes of the secret individually. Additionally the use of the chip specific key K prevents the use of a profiling device.

4.4 Security hygiene

The security hygiene is the basic principles that designers must follow to not introduce the simplest vulnerabilities. A few can be easily listed:

- avoid security through obscurity (Kerckhoffs' principle),
- if a device is compromised, the whole system must not be (do not hide a global secret on all devices),
- secrets should not be handled directly by the device (reduce physical attacks exposure),
- impose first access security configuration (to avoid the mere existence of default login/password pairs).

5 Conclusion

Security must be thought and included in the design process from the start. Our PIN code attack example shows that new threats arise as soon as the device falls into the attacker's hands, in our case with physical attacks. Unfortunately many devices now on the market fail to ensure even basic security. Adding security is not a trivial thing, and must be thought from the ground up. Protocols and implementations must be correct. Trade-off must be made between resource use and security. Dedicated chips embedding security features are a must-have for IoT devices to be truly considered secure.

Acknowledgment This work was partially funded by the French National Research Agency (ANR) as part of the program Digital Engineering and Security (INS-2013), under grant agreement ANR-13-INSE-0006-01

References

1. Eric Brier, Christophe Clavier, and Francis Olivier. Correlation Power Analysis with a Leakage Model. In CHES, pages 16{29, 2004.

2. Denis Foo Kune and Yongdae Kim. Timing attacks on pin input devices. In Proceedings of the 17th ACM conference on Computer and communications security, pages 678{680. ACM, 2010.

3. Paul C Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Advances in Cryptology|CRYPTO'96, pages 104{113. Springer, 1996.

4. Stefan Mangard, Elisabeth Oswald, and Thomas Popp. Power analysis attacks: Revealing the secrets of smart cards, volume 31. Springer Science & Business Media, 2008.

5. Stefan Mangard, Elisabeth Oswald, and Thomas Popp. Power analysis attacks: Revealing the secrets of smart cards, volume 31. Springer Science & Business Media, 2008.

6. Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (EMA): Measures and counter-measures for smart cards. In Smart Card Programming and Security, pages 200{210. Springer, 2001.

7. Christophe Giraud. Dfa on aes. In H. Dobbertin, V. Rijmen, and A. Sowa, editors, Advanced Encryption Standard - AES, volume 3373 of Lecture Notes in Computer Science, pages 27{41. Springer, 2005.

8. Sergei P Skorobogatov and Ross J Anderson. Optical fault induction attacks. In Cryptographic Hardware and Embedded Systems-CHES 2002, pages 2{12. Springer, 2003.

9. Endo Sho, Sugawara Takeshi, Homma Naofumi, Aoki Takafumi, and Satoh Akashi. An on-chip glitchy-clock generator for testing fault injection attacks. J. Crypto-graphic Engineering, 1(4):265{270, 2011.

10. Nicolas Moro, Amine Dehbaoui, Karine Heydemann, and Emmanuelle Robisson, Bruno andEncrenaz. Electromagnetic fault injection: towards a fault model on a 32-bit microcontroller. CoRR, abs/1402.6421, 2014.

11. Li Yang, Sakiyama Kazuo, Shigeto Gomisawa, Toshinori Fukunaga, Junko Takahashi, and Kazuo Ohta. Fault Sensitivity Analysis. In CHES, volume 6225, pages 320{334, 2010.

12. Thomas Roche, Victor Lomne, and Karim Khalfallah. Combined fault and side-channel attack on protected implementations of aes. In CARDIS, pages 65{83, 2011.

13. Daniel Genkin, Lev Pachmanov, Itamar Pipman, and Eran Tromer. Stealing keys from pcs using a radio: Cheap electromagnetic attacks on windowed exponentiation. In International Workshop on Cryptographic Hardware and Embedded Systems, pages 207{228. Springer, 2015.

14. Ronan Lashermes, Jacques J. A. Fournier, and Louis Goubin. Inverting the Final Exponentiation of Tate Pairings on Ordinary Elliptic Curves Using Faults. In Cryptographic Hardware and Embedded Systems - CHES 2013 - 15th International Workshop, Santa Barbara, CA, USA, August 20-23, 2013. Proceedings, pages 365{ 382, 2013.

15. Manuel San Pedro, Mate Soos, and Sylvain Guilley. Fire: fault injection for reverse engineering. Information Security Theory and Practice. Security and Privacy of Mobile Devices in Wireless Communication, pages 280{293, 2011.

16. Christophe Clavier. An improved scare cryptanalysis against a secret a3/a8 gsm algorithm. Information Systems Security, pages 143{155, 2007.

17. Matthieu Rivain and Thomas Roche. SCARE of secret ciphers with spn structures. IACR Cryptology ePrint Archive, 2013:636, 2013.

18. Helene Le Bouder , Ronan Lashermes , Yanis Linge , Bruno Robisson and Assia Tria. A Unified Formalism for Physical Attacks. Cryptology ePrint Archive, Report 2014/682, 2014.

19. Francois-Xavier Standaert, Tal G Malkin, and Moti Yung. A formal practice-oriented model for the analysis of side-channel attacks. IACR e-print archive, 134, 2006.

20. Francois-Xavier Standaert, Francois Koeune and Werner Schindler . How to compare profiled side-channel attacks? In Applied Cryptography and Network Security, pages 485{498. Springer, 2009.

21. Francois-Xavier Standaert, Tal Malkin and Moti G Yung,. A unified framework for the analysis of side-channel key recovery attacks. In Advances in Cryptology-Eurocrypt 2009, pages 443{461. Springer, 2009.

22. Stefan Mangard, Elisabeth Oswald and Francois-Xavier Standaert. One for all{all for one: unifying standard differential power analysis attacks. IET Information Security, 5(2):100{110, 2011.

23. Ingrid Verbauwhede, Dusko Karaklajic and Jörn-Marc Schmidt. The Fault Attack Jungle-A Classification Model to Guide You. In Fault Diagnosis and Tolerance in Cryptography (FDTC), 2011 Workshop on, pages 3{8. IEEE, 2011.

24. Alessandro Barenghi, Luca Breveglieri, Israel Koren and David Naccache. Fault Injection Attacks on Cryptographic Devices: Theory, Practice, and Countermeasures. Proceedings of the IEEE, 100(11):3056{3076, 2012.

25. Alexander Becher, Zinaida Benenson, and Maximillian Dornseif. Tampering with Motes: Real-World Attacks on Wireless Sensor Networks. In Sicherheit 2006: Sicherheit - Schutz und Zuverlässigkeit, Beiträge der 3. Jahrestagung des Fachbereichs Sicherheit der Gesellschaft für Informatik e.v. (GI), 20.-22. Februar 2006 in Magdeburg, pages 26{29, 2006.

26. Colin O'Flynn and Zhizhang Chen. Power analysis attacks against IEEE 802.15.4 nodes. In Constructive Side-Channel Analysis and Secure Design - 7th International Workshop, COSADE 2016, Graz, Austria, April 14-15, 2016, Revised Selected Papers, pages 55{70, 2016.

27. Helene Le Bouder, Thierno Barry, Damien Courousse, Ronan Lashermes, and Jean-Louis Lanet. A template attack against VERIFY PIN algorithms. In Secrypt, 2016.

28. Lionel Riviere. Securit des implementations logicielles face aux attaques par injection de faute sur systemes embarques. PhD thesis, Telecom Paris Tech, 2015.

29. Damien Courousse, Thierno Barry, Bruno Robisson, Philippe Jaillon, Olivier Potin, and Jean-Louis Lanet. Runtime code polymorphism as a protection against side channel attacks. In WISTP, 2016.

Cyber-Physical Protections for IoT Devices

Jean-Luc Danger[1;2], Sylvain Guilley[1;2],
Sylvie Herbel[1], Thibault Porteboeuf[1], and Michaël Timbert[1;2]

1 Secure-IC S.A.S., 15 rue Claude Chappe, 35 510 Cesson-Sévigné 2 TELECOM-ParisTech, 37 rue Dareau, 75 014 Paris

Abstract IoT devices offer a large coverage for potential attacks, arising either from the network or from the physical layers.
In this submission, we exhibit the commonalities between cyber- and physical-attacks. Next, we show that it is possible to resist them, using code and data confidentiality and integrity mechanisms, effective even in adversarial environments. Data protection is a topic well studied, e.g., in communities such as CHES. However, quite surprisingly, code protection at hardware level is less studied. We present in this contribution innovative results on this topic.

This paper belongs to the specialized category of C&ESAR 2016.

Keywords: Cyber-physical systems, passive and active attacks, security of IoT devices, protection of data and code.

1 Introduction

Gartner says 6.4 billion connected “things” are in use in 2016, with a growth of 30%. Thus, in 2017, there will be more “things” than human beings. Those are as many targets for attacks, with goals such as device hijacking, cloning, misuse, etc. or network infiltration for intelligence, information theft, etc.

The target of attack in an IoT network can be either the connected devices or the infrastructure. Obviously, the connected devices are the weakest link, thus more likely to the considered as attackers’ preferred target. Clearly, these devices can be continuously be monitored. However, in practice, some attacks (be them passive or active) can be perpetrated below the radar, hence being unnoticed by any supervision mechanism.

It is preferable to implement low-level protections, e.g., at the hardware level. Indeed, some IoT devices have no operating system (OS). Besides, even when there is an OS, it is recommended not to rely only on high-level security enforcement, since those mechanisms consume resources, and are potential easy targets of attacks aiming at bypassing them.

Table 1: Examples of cyber- and physical-attacks.

World / Attack	Cyber-attack	Physical-attack
Passive (SCA)	Memory disclosure (Heartbleed), bypassing ASLR	Recovering keys by monitoring power [13], electromagnetic emanations, etc.
Active (FIA)	Memory corruption (e.g., stack smashing)	Skipping tests (e.g., with laser injection), differential fault analysis [11], bug attack [2], etc.

Outline. In section 2, we survey the newly published threats on IoT devices. Innovative protections are discussed in section 3. Eventually, conclusions and perspectives are given in section 4.

2 Cyber-physical attacks

The peculiarity of an IoT device is that attacks can come indifferently from the network (cyber attacks) or from the bottom (physical attacks). Still, in both case, one can categorize attacks as either passive or active. Some attack examples are reported in Tab. 1.

- Passive attacks (or Side-Channel Attacks, SCA) consist in spying some observable quantities, hopefully somehow connected to a secret.
- Active attacks (or Fault Injection Attacks, FIA) deduce information about the IoT device by monitoring input/output relationships or by modify the code (e.g., via instructions skip).

One can notice that cyber- and physical-attacks can sometimes be related. For instance:

- cyber-attacks can turn into physical-attacks, like in the case of the RowHammer [12, 9], or
- physical-attacks can turn into cyber-attacks, as in the case of buffer over-flow triggered physically [7].

All those attacks (passive/active, cyber/physical) can be combined, which calls for holistic protections.

3 Protections against cyber-physical attacks

3.1 Frameworks: SGX, TEE, etc.

Generic protections have thus appeared. For instance, Intel proposed SGX (Software Guard Extensions), where a protected process has its data encrypted and authenticated [10].

A standardization attempt is the TEE (Trusted Execution Environment). The goal of TEE is to provide to the wild world (called Rich Execution Environment, or REE) an API to execute secure applications. The TEE has a protection profile [8], suited for practical evaluations (e.g., CC).

In the rest of this section, we give an overview of protections for data (Sec. 3.2) on the one hand, and code (Sec. 3.3) on the other hand. We have specific contributions about code protection.

3.2 Classical protections against side-channel and fault injection attacks on data

There are three ways to protect data against passive observation:

1. balance it (hiding) [14, Chap. 7],
2. randomize it (masking) [14, Chap. 9],
3. tolerate leakage, e.g., by frequently updating keys (solution known *as leakage resilience*).

Notice that even those countermeasures can be bypassed. For instance, it is possible to conduct high-order timing attacks [3] on a masked program. Other powerful attacks are high-order cache attacks, on masked implementations. Those have not been demonstrated yet, but are plausible.

3.3 Protection of the code

We provide in this section results from the CyberCPU RAPID project, which is followed by CyberCPU+ (award at the Concours Mondial de l'Innovation).
For the sake of clarity, we divide attacks on code in two categories:

1. attacks which attempt to modify (statically) the code[1],
2. attacks which attempt to hijack (dynamically) the code, i.e., which manipulate pointers to disrupt the execution flow.

In the first case, the asset to be protected is the code (memory), whereas in the second case, the asset is the control flow graph (CFG[2]).

From our experience, there are two strategies to protect against attacks which modify code and CFG:

Table 2: Four hardware-level technologies against attacks on code

Protection type	Protected asset	REV [6]	PCX [5]	SCALL	HCODE [4]
Preventive	Code	X			
Preventive	CFG		X		
Detective	Code				X
Detective	CFG			X	X

1. Preventive protections, which intend to make it hard for an attacker to execute the attack. The used mechanisms are secrecy of code and the of branch-

[1] Even if the code is checked at loading time, an attack can alter it during its execution.
[2] In this paper, we designate by CFG the full execution graph, including intra- and inter-procedural execution paths; such CFG is also sometimes referred to as a super-CFG.

ing destination. This way, an attacker ignorant of the involved secret is only able to inject random code or addresses, thereby being deceived most of the time.

2. Detective protections, which consist in checking the code and/or CFG integrity using a pristine a priori information, typically collected during compilation. The working factor is to compare some information measured during (dynamic) execution with its redundant precharacterized value (statically collected).

We will now describe four hardware countermeasure techniques, called REV, PCX, SCALL and HCODE. Their protection principle is summarized in Tab. 2.

REV. REV consists in encrypting the code with a stream cipher, keyed with a secret value which depends on:

- a secret key, proper to each device, or even changed at every launch of the process,
- the program counter, so as to avoid splicing attacks.

PCX. PCX specifically protects return addresses saved on the stack, by encrypting them with a secret key. It prevents an attacker from forging valid return addresses by the exploitation of a buffer overflow on the stack frames.

SCALL. In order to detect control-flow hijacking, we have implemented SCALL. It consists in a hardware shadow stack, which contains a copy of each PC saved on the stack. When returning from a function, this copy is checked against the return address stored on the stack.

Notice that SCALL consists in the detective counterpart of PCX. The advantage of SCALL is that a mismatch between return address on the stack and in the shadow stack raises an interruption immediately; it can be used to either stop the program or fix the corrupted return address value.

Real-world tests have been done on a LEON processor running an Nginx web server over GNU/Linux: a bu□er overflow is stopped with SCALL activated. Besides, we checked that the performances have not be a□ected by the countermeasure.

HCODE. To check that each basic block is executed as intended in the source code, HCODE [4] can be mobilized. This protection computes hashes (ideally, the hashes are keyed, to avoid forgeries — if in addition, the hashes are salted with the start address of the basic block, slicing is also made impossible) for each sequence of straightline code, and compares it with a genuine hash which is computed beforehand (and stored as a reference in some memory[1]). Therefore, if a cyber attack has managed to overwrite the code, or a laser injection has also corrupted it, the protection will detect the problem. This protection enables some kind of control flow integrity (CFI), insofar as any jump (due to ROP or JOP attacks) in the middle of a basic block will be detected, since its signature will be incorrect.

Now, the attacker can also force the program to jump to a legal location, namely the beginning of a basic block. In this case, protections such the pure software CFI (e.g., that of Abadi [1]) is able (in software or in hardware) to detect that the control flow has been altered.

[1] The hashes need not be saved in a tamper-resistance memory, since they are encrypted with an unknown key — therefore, an attacker cannot adaptively recompute a hash which would yield a valid signature for a forged execution path

4 Conclusions and perspectives

In this paper, we illustrate that cyber- and physical-attacks both consist in spying on or modifying some data. Therefore, similar protections can be enforced. We illustrate them, for both a protection of the data and of the code. In order to grant code integrity, specific techniques are possible, such as HCODE. They require some minor modifications to the processor and to the compiler, with an acceptable (< 10%) penalty in terms of speed.

References

1. Martín Abadi, Mihai Budiu, Úlfar Erlingsson, and Jay Ligatti. Control-flow integrity principles, implementations, and applications. ACM Trans. Inf. Syst. Secur., 13(1), 2009.

1. Eli Biham, Yaniv Carmeli, and Adi Shamir. Bug attacks. In CRYPTO, volume 5157 of LNCS, pages 221–240. Springer, 2008. Santa Barbara, CA, USA.

2. Jean-Luc Danger, Nicolas Debande, Sylvain Guilley, and Youssef Souissi. High-order timing attacks. In Proceedings of the First Workshop on Cryptography and Security in Computing Systems, CS2 '14, pages 7–12, New York, NY, USA, 2014. ACM.

3. Jean-Luc Danger, Sylvain Guilley, Thibault Porteboeuf, Florian Praden, and Michaël Timbert. HCODE: Hardware-Enhanced Real-Time CFI. In Proceedings of the 4th Program Protection and Reverse Engineering Workshop, PPREW-4, pages 6:1–6:11, New York, NY, USA, 2014. ACM.

4. Jean-Luc Danger, Sylvain Guilley, Thibault Porteboeuf, Florian Praden, and Michaël Timbert. Hardware-enforced protection against buffer over-flow using masked program counter. In Peter Y. A. Ryan, David Naccache, and Jean-Jacques Quisquater, editors, The New Codebreakers - Essays Dedicated to David Kahn on the Occasion of His 85th Birthday, volume 9100 of Lecture Notes in Computer Science, pages 439–454. Springer, 2016.

5. Jean-Luc Danger, Sylvain Guilley, and Florian Praden. Hardware-enforced Protection against Software Reverse-Engineering based on an Instruction Set Encoding. In Suresh Jagannathan and Peter Sewell, editors, Proceedings of the 3rd ACM SIGPLAN Program Protection and Reverse Engineering Workshop 2014, PPREW 2014, January 25, 2014, San Diego, CA, page 5. ACM, 2014.

6. Pierre-Alain Fouque, Delphine Leresteux, and Frédéric Valette. Using faults for buffer overflow effects. In Sascha Ossowski and Paola Lecca, editors, Proceedings of the ACM Symposium on Applied Computing, SAC 2012, Riva, Trento, Italy, March 26-30, 2012, pages 1638–1639. ACM, 2012.

7. GlobalPlatform Device Committee. TEE Protection Profile Version 1.2, November 2014. http://www.commoncriteriaportal.org/files/ppfiles/anssi-profil_PP-2014_01.pdf.

8. Daniel Gruss, Clémentine Maurice, and Stefan Mangard. Rowhammer.js: A remote software-induced fault attack in javascript. In Juan Caballero, Urko Zurutuza, and Ricardo J. Rodríguez, editors, Detection of Intrusions and Malware, and Vulnerability Assessment - 13th International Conference, DIMVA 2016, San Sebastián, Spain, July 7-8, 2016, Proceedings, volume 9721 of Lecture Notes in Computer Science, pages 300–321. Springer, 2016.

9. Shay Gueron. A memory encryption engine suitable for general purpose processors. IACR Cryptology ePrint Archive, 2016:204, 2016.

10. Marc Joye and Michael Tunstall. Fault Analysis in Cryptography. Springer LNCS, March 2011. DOI: 10.1007/978-3-642-29656-7 ; ISBN 978-3-642-29655-0.

11. Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji-Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors. In ACM/IEEE 41st International Symposium on Computer Architecture, ISCA 2014, Minneapolis, MN, USA, June 14-18, 2014, pages 361–372. IEEE Computer Society, 2014.

12. Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, CRYPTO, volume 1666 of Lecture Notes in Computer Science, pages 388–397. Springer, 1999.

13. Stefan Mangard, Elisabeth Oswald, and Thomas Popp. Power Analysis Attacks: Revealing the Secrets of Smart Cards. Springer, December 2006. ISBN 0-387-30857-1, http://www.dpabook.org/.

Enhanced IoT security through orchestrated policy enforcement gateways

Hamza Attak, Ludovic Jacquin, Adrian L. Shaw, Marco Casassa-Mont, and Yolanta Beresna

Hewlett Packard Labs, Security and Manageability lab, Bristol (UK).

{hamza.attak, ludo, als, marco_casassa-mont, yolanta.beres}@hpe.com

Abstract. The increasing number of IoT devices raises concerns about the amount of data they generate and – more importantly – their content, having security and privacy implications. The Things are mostly constrained by typical embedded design limitations from non-extensible functionalities to poor or non-existent configuration; adding security features to these devices is therefore impractical. This paper presents a network security infrastructure suitable for IoT devices, which aims at offloading the security from the devices to the nearest network edge they are connected to. First, the SECURED architecture for the network edge device (NED) is detailed: its components, security policy refinement and translation, and the way it addresses mobility of the things. Then, the SHIELD architecture proposes to extend and strengthen the security of the IoT devices by leveraging the dynamic deployment of security controls with analytics, which permits orchestrated security at the entire infrastructure level – allowing a new threat detection paradigm.

Keywords: SDNFV infrastructure, network security, trusted computing, big data

Category: specialised.

1 Introduction

The rate of new devices and platforms connecting to the internet is causing concern throughout the security industry. Firstly, the number of different platforms to secure is akin to the mobile-device-management problems already faced today, only the quantity is far larger and capability gaps between devices is even greater. Secondly, some IoT devices need to operate within such tight operational constraints (such as power or form-factor restrictions) that even if manufacturers wanted to add sufficient security features – such as data or traffic encryption, access control, etc. – , they could not. Finally, the heterogeneity of the IoT landscape, and the potential size of its deployment, requires an automated management of its security.

This paper presents two solutions to these challenges by leveraging the Network Function Virtualisation (NFV) – and the underlying Software-Defined Network (SDN) – paradigms. Specifically, section 2 presents the two main results of the EU SECURED project [1], namely, (i) the security-enhanced NFV infrastructure which allows to of-fload some IoT security into the edge of the network they are connected to, (ii) the solutions for an easy to use security: high level security policies – and the translation services to transform them into enforceable low level configuration –, but also (iii) a mobility-aware solution – whereby the infrastructure can move the security controls to follow the IoT devices. Section 3 focuses on the security improvement allowed by flexible deployment of security controls in the network,

coupled with analytics which combine multiple security controls dynamically to secure the end devices; these improvements will be demonstrated in the EU SHIELD project. Section 4 discusses the results obtained during the research work. Section 5 concludes this paper, highlighting the major contribution and next steps of these two projects.

2 SECURED: security at the network edge

Through the presentation of the SECURED project, this section proposes solutions to enforce security compatible with IoT devices, which avoids altering the actual devices, by offloading the security at the network edge. In this context, SECURED focuses on three main goals: (i) ensure the security of IoT devices; (ii) provide an easy-to-use and configurable way to specify the security policy for a user or an IoT device; (iii) and define a mobility paradigm for geographically moving subjects (such as wearables or drones for instance).

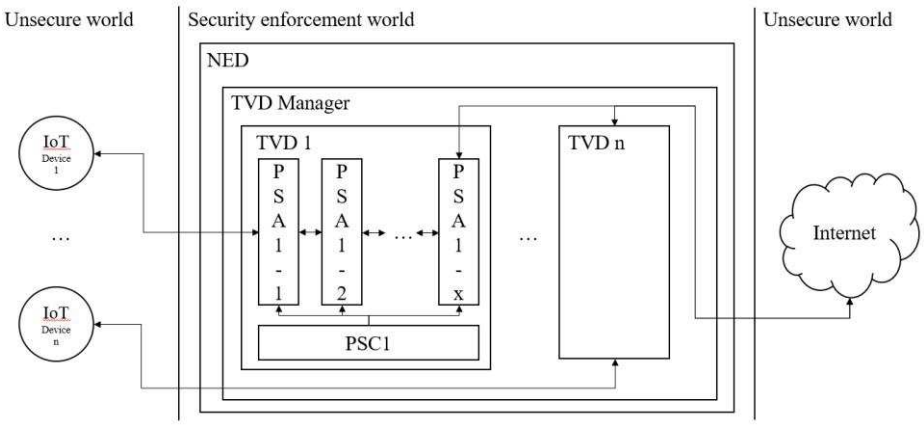


Fig. 1. Overview of the NED and its component

2.1 The SECURED architecture and design

SECURED is a complete solution for securing existing IoT devices and networks which strongly emphasises on usability: it enforces the security at the nearest network edge instead of on the device itself. Standing between the devices and the outer – un-secured – network, it can therefore not be bypassed. Moreover, the security being of-floaded from the Thing, it does not require any modification of the client devices.

It is architectured around three main components in order to answer the security challenges: the Network Edge Device (NED), the Trusted Virtual Domain (TVD) and the Personal Security Application (PSA). An overview of the architecture and the different components' interactions can be seen on figure 1; each of these components is detailed in this section, starting with the PSA concept up to their organization.

In SECURED, security is accomplished in a software-driven fashion, where instances of security applications are dynamically deployed to perform their specific function. These functions are represented in the architecture by the PSAs: a PSA being a self-contained application that performs one or more security functions.

A PSA enforces a set of security functions, but a device might need more than one PSA to achieve its security requirements. The TVD concept embraces the set of chained PSAs needed by a tenant (similarly to a network service chain in ETSI NFV [2]) and combines it with a Personal Security Controller (PSC). TVDs are tenant specific and

provide isolation of the security for each IoT device. The order and configuration of the PSAs inside a TVD are critical to enforce the correct tenant security: it is defined through a policy model, which is described in section 2.3. The PSC represents the controller of the tenant PSAs; it allows to continuously dialogue and monitor them.

Finally, the NED brings together all the previous components into an instantiation of the SECURED concept. Physically, it is the device – or set of devices in the case of a distributed deployment – that gates the Things from the unsecure outer network. It hosts the TVD of every IoT device connected to it.

The TVDs are managed by a TVD manager whose role is to create, destroy the TVDs and also to monitor – through to the PSC – that all the TVDs are working correctly.

Following the SDN model, a control plane and a data plane are setup in order to isolate the different communication flows occurring inside the NED. The data plane is represented by the PSAs and their chaining with the tenant. The control plane is present at two locations in SECURED: (i) between the TVD manager and each TVD (actually through its PSC), and (ii) between the PSCs and their associated PSAs.

The NED being the point where the security functions are executed, it has to be trusted by the tenants. SECURED leverages trusted computing techniques such as remote attestation [3] with the use of discrete Trusted Platform Module (TPM) to ensure of the identity of the NED and the state of its software stack – validated against known-good states –; SECURED can also verify the SDN rules applied by the network switches [4] in the case of the user's traffic being redirected (e.g. when the current NED cannot enforce every PSAs of the user).

2.2 Implementation

In practice, PSAs can be implemented as virtual machines (VM) or containers. There are two technical requirements for a PSA: first, a PSA needs to be isolated from the other components (i.e. PSAs, PSCs, TVDs, TVD manager) running on the same NED's host; second, a PSA need at least three network interfaces: two defined as chain-head and chain-tail, in order to plug the PSA between the device to secure and the outside unsecure network; another one is reserved in order to control the PSA from the TVD – through the PSC –, representing the control plane.

Similar to PSAs in practice (VM or container), the role of a PSC is to control and monitor the PSAs inside the same TVD. It is also the bridge between the TVD manager, running on the host system of the NED, and the PSAs. Its main function is to start, stop, configure or get the status of PSAs. It is the abstraction layer for interacting with the TVD (either by the tenant or by the TVD manager). Similar to a PSA, the PSC must respect storage and execution isolation on the system it is running on. It also needs two network interfaces: one to control the different PSAs and one to receive higher level requests from the TVD manager running on the host system. The PSC is only present on the control plane.

The TVD being the association of a set of PSAs and one PSC, it is defined mostly as a conceptual abstraction. Nonetheless, its practical parts appear at TVD runtime. As a matter of fact, when the TVD manager instantiates a TVD, it has to create all the networking links between the PSAs – representing the data plane service chain – and attach itself to the PSC in order to manage it.

The TVD manager represents the isolation layer between the different TVDs - and associated components; it also includes the management components for creating and managing the TVDs. In practice, it can be implemented through a virtualisation hypervisor or a container engine.

The SECURED architecture ensures isolation between TVDs in order to respect privacy between users. In practice, it is mainly done by the fact that PSAs isolation is already guaranteed by the hypervisor or container engine and the PSA chain is ensured to be linked without leaks with a well-understood and trusted network configuration. For more advanced TVDs topologies, human verification does not scale. There is a need for automated verification in more dynamic and complex virtual network configurations to ensure that isolation is applied properly.

Open vSwitch [5] and OpenFlow [6] technologies are used for their flexibility in order to dynamically chain the PSAs together and therefore create a TVD between a given IoT device connected to a specific port and the outer unsecured requested resource.

2.3 Policy Model

SECURED can enforce security at the edge for any device or user with multiple devices and applications. In the context of this paper, the security for an IoT device can be represented as nothing else but a TVD instantiated for a specific user having a single device. As a user-centric concept, it is therefore very important to provide good interfaces to interact with the SECURED components. These interfaces and their organization are the subject of this section.

As security functions, PSAs need to be configured to work correctly. One of the problems between non-technical users and software in general is that it is a tedious task to configure software. This statement is even truer with security software.

Through SECURED, a high-level way to define security policies is proposed. Simply called the High-level Security Policy Language (HSPL), it is aimed to be very close to the natural language.

The challenge here is to provide HSPL for the user and at the same time achieve a coherent security chain in the TVD with this high level configuration.

For the sake of flexibility and compatibility, SECURED does not intend to modify the security applications running on the PSAs to be compatible with HSPL (even though the PSAs still need to be configured with their specific syntax and languages). Instead, one more level of abstraction is introduced in order to bridge HSPL with the low level PSA-specific configuration. This is the Medium-level Security Policy Language (MSPL).

Its goal is to extract from the HSPL, the different security capabilities required by the policy. It also set up an application graph which is a proposal of a service chain with different available PSAs to realise the security policy defined by the user. Anomalies can occur at this stage, SECURED warns the user if one happens (unavailable PSAs for example).

The interactions between the different abstraction levels can be seen on figure 2.

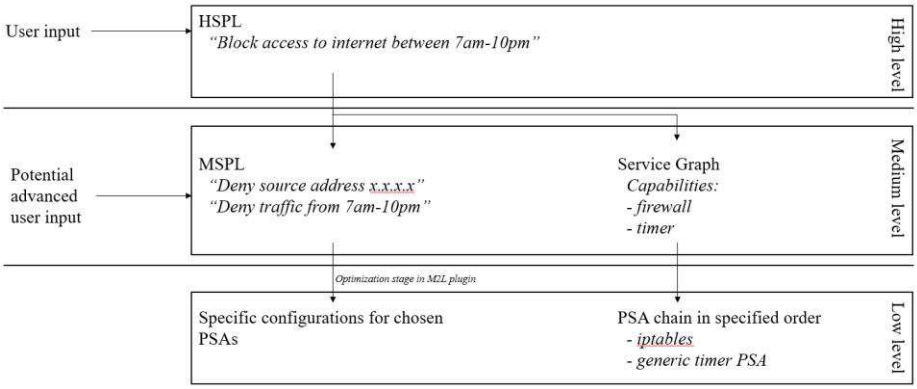


Fig. 2. Workflow example for a given policy

In order to propose PSAs coherent with the defined high level policy, a set of security capabilities is defined beforehand. The HSPL implements these capabilities and each PSA exposes its set. Depending on the capability, a PSA will be place before another on a specified service chain. For example, it makes no sense to put a traffic encryption PSA at the beginning or in the middle of a chain.

SECURED also tries to address the scaling problem. With this idea in mind, SECURED authorizes the network administrators or advanced users to modify their MSPL configuration and specify the specific PSAs they want to use or their orders. Of course, in this situation, the coherency and correctness of the PSAs security chain is not guaranteed by the architecture. Nonetheless, SECURED could still detect certain anomalies on the edited service chain and therefore warn the user (encryption PSA not at the end of a chain for example, or PSAs with redundant capabilities in the chain).

Different constraints can appear when a user is setting up his policy. These constraints are the results of higher priority policies that can be set up by a higher authority entity (for example, parents for children, company for employees…). Before translating the MSPL to low level configurations for the PSAs, a policy reconciliation step is to be conducted in order to verify the compliance of the user-defined policy with the potential higher priority policies (the reconciliation process detailed in [7]). Before applying the policy, its reconciled version is displayed to the user on the SECURED configuration dashboard.

On one hand, the application graph is specifying which PSAs to instantiate and their order, on the other hand, their configuration is still to be translated from the generated MSPL. In order to facilitate the translation, a configuration template is defined for each PSA. This permits the translation process to only fill in the variable parts of the configuration. The PSA developer is responsible for providing this template and the translation plugin between the standardized MSPL and the low-level PSA-specific configuration. The same way SECURED is device-agnostic, this also makes it application-agnostic.

2.4 Mobility scenario

Given the dynamic nature of some IoT deployment, SECURED proposes a solution to maintain a user security context throughout the different NED a device can connect to. When a secure channel to a new NED is established, SECURED detects the user profile and immediately starts to migrate its TVD from the last NED to the new one, which allows a seamless handover from one NED to another.

Let an IoT device wirelessly connected to a SECURED node A with certain signal strength. Let it be moving away from A and therefore detects a decrease in node A's signal. From this situation, three cases are defined in the mobility context:

- The device meets another SECURED access point, in order to maximize performance, the SECURED architecture will start to transfer the tenant's TVD configuration to the second access point (see figure 3). It permits to have the newly instantiated TVD fully ready for when the new device will connect in order to cut down initialization and starting delays. This can be triggered by comparing signal strength and starting the transfer when the first access point's signal is under a defined threshold and the second access point's signal is strong enough.
- The device does not meet another SECURED access point, the device will simply lose connection and the security features added by SECURED. The tenant's TVD will be destroyed as soon as the connection with the SECURED node is terminated.
- The device meets another SECURED but never connects to the new access point. In that case, the device is still connected to the first access point but its TVD has started to be migrated to the next access point. In order not to keep its unused TVD on the second access point, a timeout is defined so that if the device does not transfer to the new TVD, it is automatically destroyed to avoid wasting resources on the second access point.

This description can be generalized to scale to more than two access points. As a matter of fact, the TVD configuration can be downloaded to all the nearby SECURED nodes or a more sensible solution would be to only transfer the configuration to the second strongest access point in terms of signal.

Also it has to be noted that the mobility scenario implies two major points: (i) the IoT device must be updated with the access point decision algorithm, it is therefore intended to more capable type of devices; (ii) there must a repository of tenant's configurations and their PSAs available to the SECURED nodes, it is out of scope of this paper but it is part of the SECURED project and is described in [1].

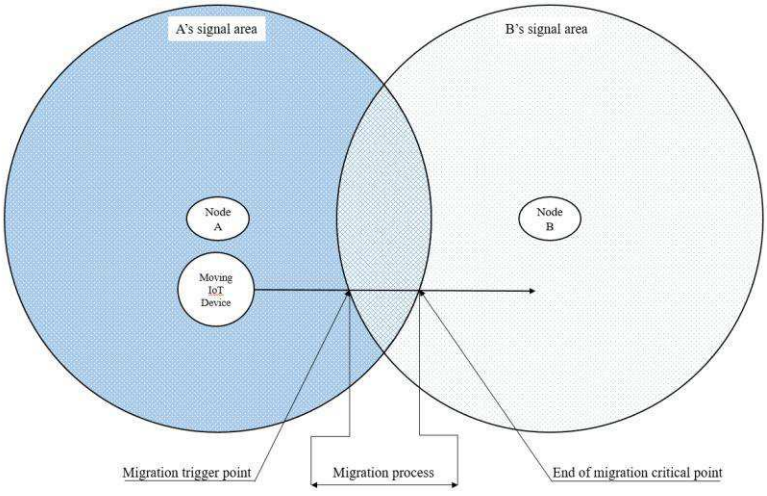


Fig. 3. Migration stages in the mobility scenario

3 SHIELD: leveraging big-data analytic for flexible security

SHIELD introduces big data analytics and remediation capabilities, on top of the SECURED infrastructure, to detect both known and unknown security attacks and remediate them. The core goal of SHIELD is to leverage a network-wide security

view so that it can address distributed attacks such as Advanced Persistent Threats (APT).

In SHIELD, IoT devices consist of a very high number of distributed networked devices that connect to the network edge devices (e.g. in campus size networks). As previously mentioned, these IoT devices might lack basic security capabilities and rely on the underlying networking capabilities to share with other devices the data they locally collect (and potentially process).

In SHIELD, the edge networking devices are instrumented to further collect information about IoT devices, in particular about their networking and behaviour. To achieve this, they run virtual Network Security functions (vNSFs – conceptually similar to SECURED PSAs) under the control of a user/administrator and are orchestrated by SHIELD Management & Orchestration modules. They collect overall networking events (e.g. netflow, DNS, etc.), including the one generated by IoT devices, locally process it and share it with centralized big data analytics solutions. They also play a key role as threat remediation end-points whereby actions to mitigate security risks and attacks are executed within these devices, with impact on locally networked systems.
The current high-level SHIELD architecture is shown in the following figure:

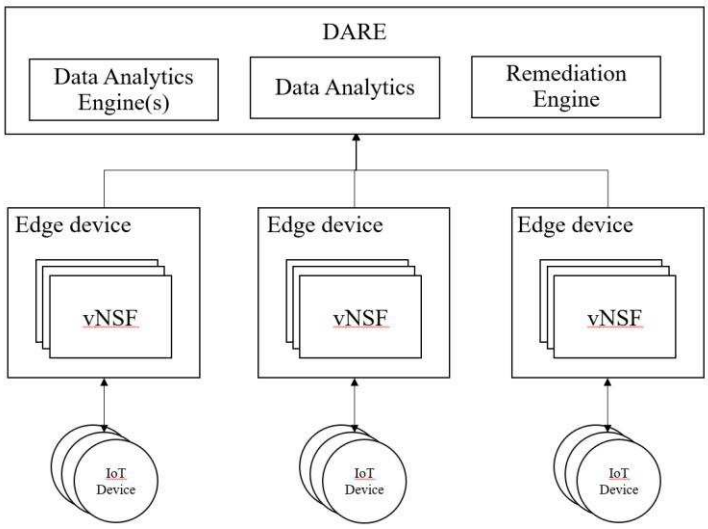


Fig. 4. High-level SHIELD architecture

The IT Infrastructure, inclusive of distributed edge devices, hosts and networks, represents the controlled and managed environment in SHIELD. Work done in SECURED, with regards to infrastructure attestation and verification, is further applied to SHIELD to continuously check for the integrity of edge devices, hosts, software and vNSFs.

On top of the IT Infrastructure, SHIELD runs a standard NFV Infrastructure, compatible with the one described in ETSI NFV Reference framework and architecture [3].

Specifically, SHIELD provides a set of vNSFs (as instances of Virtual Network Functions – VNFs) to support the collection of security events at the network level and security remediation activities (vNSFs Layer). Finally, a centralized, big data analytics solution is provided (via a set of analytics engines) to process collected data, monitor systems detect known/unknown threats and instruct remediation engines to remediate/recover attacks.

The following figure provides additional details about SHIELD components:

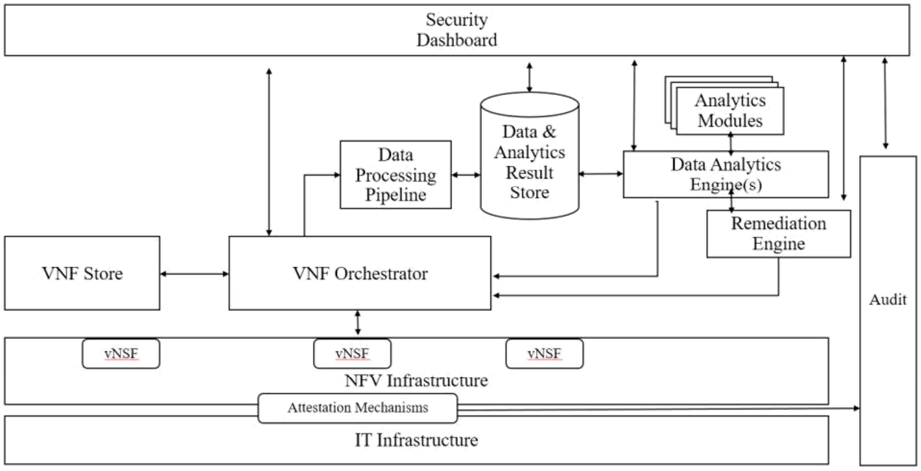


Fig. 5. SHIELD architecture components

The big data analytics solution consists of: network data/event collection points; data warehouse; analytics engines; a library of security analytics and UI.

It interacts with the underlying infrastructure via a VNF Orchestrator which also coordinates a VNF Store, storing available vNSFs functions.

Data collection points are implemented by a set of vNSFs which collect specific types of networking traffic (from IoT devices) via packet capturing techniques, including DNS events, netflow, DHCP, etc. These events are cleaned, enriched with additional metadata (e.g. geo-location of IP addresses, flagging suspicious IP addresses and domains based on threat intelligence, etc.) and stored in high-performance data repositories (inclusive of SQL and noSQL ones). Open sources event brokering and processing frameworks, including Apache Kafka [8] and Apache Storm [9] are used in the data processing pipeline.

A set of Analytics Engines processes the collected data both in near-time and on a historical basis (e.g. spanning from data collected in the last few hours back to weeks/months). They include rule engines and Apache Spark-based analytics engine.

This engine supports a wide range of threat detection mechanisms, driven by pattern-based analytics and machine learning analytics. The former checks for well-known attack patterns and mechanisms, driven by deep security knowledge and expertise in the field; the latter uses machine learning techniques to identify abnormal entity and user behaviours. Both types of analytics detect new security threats that happens at different stages of complex, advanced attacks, during potentially long time periods (weeks/months), including: initial compromise of a system/device; command & control by an external attacker site; lateral movements within an organization network; exploitation/damage.

Alerts are triggered from detected threats, further aggregated (e.g. across various engines) to determine their level of evidence/relevance and visualized to security analysts via a Security Dashboard UI.

A Remediation Engine is fed with top priority alerts and contextual information to determine a plan for mitigating existing threats and risks. This engine uses the alert

details and context (e.g. infected device, network location, business priority, etc.), coupled with existing playbooks (workflows of steps and actions) to determine which remediation actions need to be carried out. Examples of actions might include: getting security personnel authorization to automatically remediate the threat; contact device owners; intervene at the networking level e.g. by blocking networking flows, redirecting them or enabling further logging activities at the networking level. Network-level remediation level automatically happens when the Remediation Engine contacts the NFV Management & Orchestration modules and activates relevant vNSFs to carry out the desired networking activities.

Finally, and Audit Service is used in SHIELD to log various collection, detection and remediation steps to provide an audit trail for future forensic analysis and incident management purposes.

The end-to-end security approach provided in SHIELD ensures that known/unknown security threats within an organisation are quickly detected by using trustworthy data, collected from a programmable NFV infrastructure and automatically remediated (or risks mitigated) by using the same NFV infrastructure and programmable vNSFs functions. The key value proposition is drastically reducing the time needed to remediate an attack, from its initial detection, hence reducing risk exposure and damages within an organisation.

4 Evaluation

In order to demonstrate the SECURED architecture, a NED prototype is implemented [10] towards an Intel Core i7-4770@3.40GHz (4+4 cores), 32GB of RAM with Fedora 20 integrating a 3.18.7-100.fc20.x86_64 kernel. The prototype also runs Open vSwitch in order to dynamically configure the network architecture (principally the TVD) between the NED internal components. PSAs are instantiated KVM-based VM or Docker images.

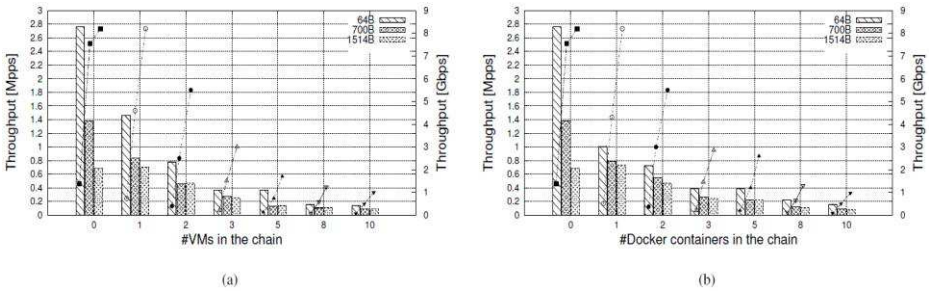


Fig. 6. Throughput of a single chain (of growing length) with VNFs deployed in: (a) KVM-based virtual machines; (b) Docker containers.

Figure 6 shows the throughput of a service chain depending on the PSA nature, the packet size and the chain length. This benchmark shows that virtualization is slightly faster than the container-based solution. It also shows that the throughput is inversely proportional to the number of running PSAs. This problem can be slightly overcome by grouping the device type into one chain, one TVD, and avoid of instantiating one chain per device. This also eases the administration of IoT devices as they are mainly data-centric. An example would be to group all the temperature sensors together so that they share they are secure the same way. This extension could easily ease the computing load on the SECURED node and need no adaptation whatsoever.

On the contrary, latency is reduced when using Docker. This is explained by the overhead implied by the virtualization and the fact that Docker has access the actu-

al kernel driver functions whereas the packets have to cross two kernel stack in virtualization.

A second NED prototype also exists, it is built around an Intel Core-i5 3427U processor, 16GB of RAM and M.2 128GB flash storage. It runs a very thin and customized version of Debian along with Xen hypervisor. It has a very small form factor (seen on figure 7) and shows the NED as a home router appliance.

Fig. 7. 4"x4" NED prototype (~10.1x10.1 cm²)

Concerning the mobility scenario, Montero et SerralGracià [11] shows that the connection handover and the migration of a chain of two 512 MB PSAs variates from 7 to 10 seconds, and an average of 7 seconds without the migration. In fact, the connection handover delay is inherent and cannot be avoided, but the PSAs can either start to be transferred during the connection handover or their image not transferred at all, as only the configuration could be downloaded from the first access point to the second.

At the time of writing this paper, the SHIELD project has just begun and can therefore not provide results or a prototype. Nevertheless, the recent DDoS attack using a large number of IoT devices is the perfect example of what SHIELD on top of SECURED is protecting against. The NEDs and the DARE infrastructure enables SHIELD to aggregate the IoT devices network behaviour in a single decision component. This allows to make network wide decisions on different kind of attacks; for example in the recent DDOS IoT-based attack, SHIELD could have leverage a simple analytics looking for distributed sources trying to connect to the same destination. Based on a specified threshold, some malicious connections could have been terminated at the NED level, protecting the core network and the destination from the flooding.

Similarly, the vNSFs are able to provide protection to those IoT devices connected to a managed network edge and could have prevented their tampering through the network in the first place.

5 Conclusion

This paper presents a security architecture for IoT devices, whereby security controls are hosted as close as possible to the user's device: at the network edge. SECURED results show how to bridge the trust gap of offloading security to the network, define a high-level common language to help setting up policies and translate them into enforceable configurations; and address the mobility challenge inherent to the IoT world.

Building on the SECURED architecture for running security controls in the network, the SHIELD vision is presented: the agile deployment of security functions,

allowed by the use of a NFV-based infrastructure, enables the development of new security analytics that can orchestrate the security applications to either capture more (or less) security events, or address the attacks by deploying the appropriate remediation functions.

These new security paradigms do not rely on the end devices to embed heavy security mechanisms, which address the main challenges of the IoT: constrained and heterogeneous devices.

Acknowledgement.
This work has been partially supported by the SECURED project - which is co-funded by the European Commission under the ICT theme of FP7 (grant agreement no. 611458), and by the SHIELD project – which is co-funded by the European Commission under the call "Digital security: cybersecurity, privacy and trust" of the H2020 program (grant agreement no. 700199).

References.

1. Montero D, Yannuzzi M, Shaw A, Jacquin L, Pastor A, Serral-Gracia R, Lioy A, Risso F, Basile C, Sassu R, Nemirovsky M, Ciaccia F, Georgiades M, Charalambides S, Kuusijarvi J, Bosco F (2015) Virtualized security at the network edge: a user-centric approach. IEEE Communications Magazine 53:176-186. doi: 10.1109/mcom.2015.7081092
2. ETSI-NFV architectural framework (2013). http://www.etsi.org/deliver/etsi_gs/nfv/001_099/002/01.01.01_60/gs_nfv002v010101p.pdf. Accessed 6 Sep 2016
3. Coker G, Guttman J, Loscocco P, Herzog A, Millen J, O'Hanlon B, Ramsdell J, Segall A,
   Sheehy J, Sniffen B (2011) Principles of remote attestation. International Journal of Information Security 10:63-81. doi: 10.1007/s10207-011-0124-7.
4. Jacquin L, Shaw A, Dalton C (2015) Towards trusted software-defined networks using a hardware-based Integrity Measurement Architecture. Netsoft
5. Pfaff B, Pettit J, Koponen T, Jackson E, Zhou A, Rajahalme J, Gross J, Wang A, Stringer J, Shelar P, Amidon K (2015) The design and implementation of open vswitch. NSDI'15 Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation:117-130.
6. McKeown N, Anderson T, Balakrishnan H, Parulkar G, Peterson L, Rexford J, Shenker S, Turner J (2008). OpenFlow: enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review, 38(2):69-74.
7. Basile C, Lioy A, Pitscheider C, Zhao S (2015) A formal model of policy reconciliation. Parallel Distributed and Network-Based Processing (PDP) 2015 23rd Euromicro International Conference:587-594. doi: 10.1109/PDP.20201515.42.
8. Kreps J, Narkhede N, Rao J (2011). Kafka: A distributed messaging system for log processing. Proceedings of the NetDB:1-7.
9. Apache Storm. http://storm.apache.org. Accessed 6 Sep 2016.
10. Bonafiglia R, Cerrato I, Ciaccia F, Nemirovsky M, Risso F (2015). Assessing the Performance of Virtualization Technologies for NFV: a Preliminary Benchmarking. 2015 Fourth European Workshop on Software Defined Networks:67-72. IEEE.
11. Montero D, Serral-Gracià R (2016). Offloading personal security applications to the Network Edge: A mobile user case scenario. Wireless Communications and Mobile Computing Conference (IWCMC), 2016 International:96-101. IEEE.