

WonderCloud, une plateforme pour l'analyse et l'émulation de micro-logiciels ainsi que la composition de pots de miels.

Mathieu GALLISSOT, Maxime PUYS, Pierre-Henri THEVENON
Univ. Grenoble Alpes, CEA LETI, 38000-F GRENOBLE
{prénom}.{nom}@cea.fr

Résumé : La caractérisation de la cybersécurité d'objets connectés ou d'équipement industriels nécessite en général d'en disposer en vue de les analyser dans des environnements de tests prévus à cet effet. Nous présentons dans cet article une alternative permettant de mener un premier niveau d'analyse sans disposer du matériel, basée sur l'étude du micro-logiciel du dispositif physique qui sera alors émulé. Cette émulation est également utilisée afin de réaliser et exploiter des pots de miels, des topologies d'objets connectés typiquement représentatifs d'un environnement réel. Ces pots de miels peuvent être utilisés à des fins de formation, de tests ou d'observation.

Mots clés : Honeypot, Emulation, ICS, IOT

1. Introduction

L'Internet des Objets (Internet of Things - IoT) vise à connecter des objets communicants à Internet afin de les piloter à distance et d'en recueillir des données. L'intérêt du grand public ainsi que des industriels a fait exploser le nombre d'objets connectés. De plus, avec l'arrivée de l'IoT Industriel et l'apparition récente des usines connectées (Factory as a Service – FaaS), on dénombrerait en 2020 plus de 30 milliards d'objets connectés avec une projection à 75 milliards en 2025¹. Ces objets sont unanimement reconnus comme étant des portes d'entrée exploitables par un attaquant, étant généralement moins sécurisés du fait de leur faible coût. Le nombre d'attaques ciblant les objets connectés a donc connu une forte croissance ces dernières années et les outils de découverte tels que Shodan [1] permettent aujourd'hui à n'importe qui d'obtenir l'adresse IP d'objets vulnérables. Pour donner un coup d'avance à la défense, de nombreux travaux ont été initiés afin d'identifier les vulnérabilités de ces objets, et ainsi fournir des briques de sécurité à intégrer dans leurs évolutions. Une alternative consiste à leurrer les attaquants en les incitant à attaquer de faux objets appelés « pots de miels » (*honeypot*), d'une part afin d'éviter qu'ils attaquent de vrais objets, d'autre part afin d'observer et d'analyser leurs techniques et ainsi permettre la sécurisation des cibles. Le challenge principal dans la création de pots de miels est donc la crédibilité de la copie afin que l'attaquant ne détecte pas l'usurpation.

Contributions : Dans cet article, nous proposons une solution permettant d'émuler des objets IoT en se basant uniquement sur leur micro-logiciel. Cette solution d'émulation est basée sur le framework d'émulation Firmadyne [2] que nous avons étendu dans le support de périphériques et d'architectures, elle permet de reproduire fidèlement le logiciel contenu dans l'objet, son véritable comportement ainsi que ses interfaces. Il est ainsi possible d'une part d'analyser la sécurité de l'objet sans risquer de l'endommager et d'autre part de créer des pots de miel réalistes. En effet, le code exécuté étant celui de l'objet étudié, un attaquant ne pourra que très difficilement faire la différence entre un objet réel et sa version émulée. En se basant sur les technologies SDN (Software Defined Networking) afin de connecter plusieurs objets émulés, il est alors possible de reproduire un réseau à grande échelle tel qu'un immeuble de bureau. Nous appliquons notre solution pour montrer comment reproduire fidèlement le modèle d'une maison connectée, intégrant à la fois des objets « consumer » tel que des caméras et des imprimantes, mais aussi des objets plus industriels en reproduisant par exemple le système de chauffage ou celui de la commande des volets roulants.

Etat de l'art : A notre connaissance, peu de solutions permettent d'émuler des objets IoT en se basant uniquement sur leur micro-logiciel. On peut notamment citer FirmPin [3], une suite logicielle développée par DeepBitsTechnology depuis 2018 et intégrant l'outil de fuzzing TriforceAFL, la plateforme d'émulation

¹ Statista, 2016. <https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/>

Firmadyne et l'outil d'analyse dynamique de code DECAF. On peut également citer l'outil Avatar [4] combinant l'émulation avec du véritable matériel en faisant suivre les accès aux entrées/sorties de l'émulateur vers le matériel. Par ailleurs, plusieurs frameworks sont développés afin d'analyser automatiquement la sécurité d'objets IoT. On peut notamment citer Expiot [5] en 2018, Hercules [6] ou encore P-SCAN [7]. Ces outils permettent l'identification de vulnérabilités dans des objets IoT ou IoT industriels en supportant de nombreux protocoles. Ils permettent de localiser des objets sur un réseau, d'écouter les communications, ainsi que d'envoyer ou rejouer des messages malicieux.

Objectifs : La plateforme d'émulation développée a plusieurs objectifs :

- Permettre la recherche de vulnérabilités en exécutant des suites de tests sur le produit émulé. L'émulation a ici un atout qui est de permettre une orchestration de tests divers, et ceci de manière graduée : scanning d'interface réseau, fingerprinting des processus réseau, recherche de vulnérabilités via base de données existantes tel que metasploit. A terme l'émulation devrait permettre l'emploi de techniques plus complètes tels que le fuzzing.
- Elle doit faire le lien avec un dispositif industriel ou un système industriel physique. La mise en place d'une plateforme de tests physique intégrant une multitude de réseaux industriels différents et de dispositifs industriels (relais de protection, automates, contrôle moteur, ...) est limité par le coût, la place et la complexité de câblage. L'utilisation de l'émulation permet donc d'étendre les capacités d'un système ou d'un réseau industriel physique existant sans complexifier l'infrastructure.
- Un honeypot consiste à attirer des personnes sur des ressources et ainsi analyser leur comportement (et éventuellement les neutraliser). Le but est de faire croire à l'intrus qu'il peut prendre le contrôle d'une véritable machine de production, d'une installation domotique, ... et d'étudier ses méthodes pour se prévenir d'attaques futures. La plateforme Wondercloud doit permettre de créer un réseau de dispositifs industriels afin de tester le réseau complet ou un dispositif physique ou émulé à l'intérieur de ce réseau contre des intrusions extérieures. De nombreuses solutions existantes permettent la mise en place d'un tel honeypot ; on peut notamment citer *Conpot* qui permet d'émuler un système industriel intégrant un réseau industriel de type Modbus. Le honeypot ainsi créé rend le réseau Modbus TCP accessible depuis internet. Cependant, l'empreinte de ce type de honeypot est relativement bien connue et très rapidement détecté comme honeypot par un outil d'analyse réseau comme *Shodan* ; il existe même des outils comme *Honeypot Hunter* qui permettent de repérer d'éventuels honeypots. La mise en place d'un honeypot modulaire intégrant des dispositifs physiques et émuls permet de s'affranchir des problèmes rencontrés par les honeypots open source.

2. Approche technologique

L'état de l'art utilise un modèle commun qui se base sur : 1) l'acquisition de micro-logiciels, souvent basée sur l'outil binwalk [8] ; 2) la préparation du micro-logiciel en vue de son émulation, utilisant un kernel approprié ; et 3) l'émulation du micro-logiciel via l'émulateur multi-architectures qemu [9]. Dans cette étude, notre motivation était double, avec en premier lieu la volonté d'améliorer les outils publiés par la communauté en vue d'augmenter le taux de réussite de l'émulation, c'est-à-dire le démarrage et l'exécution du système d'exploitation. L'amélioration de ce taux nécessite d'ajouter le support pour plusieurs architectures matérielles, mais aussi d'ajouter des étapes visant à contourner certaines protections. En second lieu, nous souhaitons pouvoir exploiter les systèmes émuls en vue d'exploiter des pots de miels représentatifs de systèmes cyber physiques complexes, c'est-à-dire composés de plusieurs objets connectés interagissant entre eux.

2.1. Extraction de micro-logiciels

La première étape pour émuler un dispositif est la récupération de son logiciel embarqué. Deux solutions principales permettent de récupérer cette donnée :

- La première solution consiste à parcourir le site web d'un constructeur afin de trouver des versions du logiciel concerné. La disponibilité des micro-logiciels sur internet d'un fabricant de dispositifs IOT et industriels est courante, car elle permet aux utilisateurs de mettre à jour leurs produit. Il existe de nombreux outils permettant de *scraper*, c'est-à-dire de récupérer « par aspiration » un ensemble d'informations sur un site web. Certains de ces scrappers sont dédiés à l'extraction de micro-logiciels d'un site web ; comme ceux proposés au sein du framework Firmadyne, mais également par d'autres

outils équivalents [10]. Certains corpus de références sont également disponibles comme celui du Fraunhofer [11].

- La seconde solution consiste à extraire physiquement le logiciel d'un dispositif en utilisant l'interface de programmation présente sur la carte électronique. Cette interface, communément appelée JTAG, permet généralement de programmer et déboguer les composants numériques de type processeur, microcontrôleurs ou FPGAs, mais peut être utilisée pour extraire le fichier binaire de la mémoire de programme. Sur un dispositif dit sécurisé, il est primordial que l'accès à cette interface soit donc désactivé afin d'empêcher un attaquant de récupérer des données sensibles dans le code (par exemple des clés, des mots de passe administrateur) ou de reprogrammer le dispositif [12] [13]. Dans la réalité, cette interface est très souvent en accès libre pour permettre une mise à jour du logiciel en usine. Cependant, l'identification des entrées /sorties de l'interface JTAG sur la carte électronique peut être complexe. En effet, les électroniciens rivalisent d'ingéniosité pour camoufler sa présence, par exemple sous la forme de points de tests ou dans un connecteur personnalisé intégrant d'autres interfaces. Des outils matériels tels que evilsplit [14], ou JTAGulator [15] peuvent permettre d'identifier parmi les pins d'un connecteur les E/S du JTAG en testant l'ensemble des combinaisons possibles. Des outils open source tels que openocd permettent ensuite de se connecter au processeur/microcontrôleur afin d'extraire le fichier binaire de sa mémoire.



Figure 1. Principe d'extraction de micro-logiciels avec notamment une sonde "JTAGulator" permettant de découvrir la structure des ports JTAG et Série ainsi que l'utilisation d'une pince de programmation pour extraire le contenu d'une mémoire SPI.

2.2. Émulation

L'émulation du micro-logiciel doit permettre de pouvoir exécuter le code du micro-logiciel dans sa quasi-totalité. Une des premières étapes est la découverte des caractéristiques du matériel sur lequel ce micro-logiciel doit pouvoir s'exécuter, il s'agit en particulier de comprendre l'architecture du microprocesseur, définissant le jeu d'instructions, mais aussi les différents périphériques systèmes tels que la mémoire et en particulier la mémoire non-volatile. Dans le cas où nous disposons du matériel, il est relativement aisé d'identifier le microprocesseur utilisé via les indications fournies sur son packaging. A partir de cette identification, nous pouvons accéder aux données constructeurs et ainsi utiliser le bon environnement d'émulation. Dans l'alternative où nous ne disposons que du micro-logiciel, sans informations sur le matériel, les utilitaires *binwalk* et *file* permettent d'identifier le jeu d'instructions des différents binaires.

Une fois l'environnement physique du micro-logiciel connu, il s'agit de l'émuler via un émulateur compatible. Qemu [9] propose de nombreuses architectures de microprocesseurs permettant ainsi de pourvoir exécuter du code prévu pour diverses architectures courantes telles que ARM, MIPS, PowerPC, Sh4, etc. Un projet alternatif, nommé xPack QEMU ARM [16], permet également de simuler des architectures plus légères que l'on trouve fréquemment sur les objets connectés. Pour faciliter l'émulation, nous optons pour l'utilisation d'un noyau Linux précompilé pour l'architecture plutôt que de tenter d'émuler le noyau original. Cette approche est utilisée dans l'état de l'art et permet un meilleur taux de réussite du démarrage du micro-logiciel sur système Linux. Elle permet, également, d'introduire de nouveaux drivers ou d'en renommer certains afin

de permettre à certains applicatifs en *user-space* de s'exécuter. Ainsi, nous avons étendu le noyau proposé par Firmadyne afin d'une part de supporter des architectures matérielles tel que PowerPC, mais également de pouvoir ajouter des modules ou drivers émulant des périphériques tels que des interfaces WiFi (module MAC80211_hwsim) ou encore des périphériques vidéo (v4l2loopback).

Dans notre approche, l'émulation est considérée comme suffisante lorsque le micro-logiciel émulé permet d'exécuter les fonctionnalités pour lequel il est prévu. Par exemple, dans le cas d'une caméra de vidéosurveillance, nous nous attendons à ce qu'un flux vidéo soit présent dans le système d'une part et d'autre part au fait que les interfaces homme/machines ou les APIs soient fonctionnelles.

2.3. Composition et simulation d'environnements physiques

Les outils de la communauté mentionnés dans l'état de l'art ont pour finalité la recherche de vulnérabilités au sein des micro-logiciels. Dans ce cas, l'émulation telle que décrite dans le paragraphe précédent sert, selon les capacités, notamment au fuzzing d'interfaces ou de binaires logiciels. Dans notre cas, nous souhaitons utiliser les systèmes émuls afin de simuler des environnements réalistes dits du domaine de l'*"Operational Technologies"* (OT). Plusieurs cas d'usage sont envisagés, tel qu'une utilisation en tant que « *Cyber Range* » [17] pour la formation, l'organisation de challenges en cyber sécurité ou encore la composition de pots de miels pour la veille active de menaces et l'identification de vulnérabilités sur le réseau internet.

Nous avons ainsi organisé l'émulation dans un framework logiciel issu d'OpenStack [18] afin d'avoir un environnement de composition de produits et réseaux virtuels que nous appelons « WonderCloud ». Cet environnement permet à ses utilisateurs de pouvoir instancier différents micro-logiciels émuls et de les interconnecter par le biais de réseaux virtuels. Ces réseaux virtuels peuvent être plus ou moins complexes avec la particularité de pouvoir être également régis par d'autres micro-logiciels émuls comme des micro-logiciels de routeurs ou de commutateurs. Enfin, ces réseaux virtuels peuvent être connectés à des réseaux physiques afin de réaliser le paradigme « hardware in the loop » [19].

Au sein de la plateforme WonderCloud nous avons trois modèles de produits que l'on peut instancier :

- Des micro-logiciels émuls
- Des modèles d'environnement physiques, interagissant avec les produits émuls via des protocoles et standards (Modbus/TCP, KNX/IP, etc)
- Des stations d'analyse avec des outils précompilés pour la recherche de vulnérabilités

3. Cas d'usage : exemple de conception d'un pot de miel pour une maison intelligente

Depuis l'émergence de la domotique dans les années 1980, de plus en plus de systèmes électroniques ont été industrialisés en vue d'apporter des services aux habitants dans le concept de maison intelligente. Longtemps considérés marginaux ou luxueux, ces systèmes connaissent une seconde jeunesse avec l'émergence des objets connectés. Fortement dépendant d'internet et des services distants, ces objets apportent de nouveaux moyens d'interactions entre des systèmes connectés et des habitants.

La particularité d'une maison connectée est qu'elle peut représenter un cas typique de vulnérabilité par l'absence de gestion de la sécurité. Un réseau résidentiel n'est administré, sauf cas exceptionnel, que par ses utilisateurs. Il s'agit du grand public incluant notamment des personnes ayant peu de connaissance en sécurité. Ce contexte rend ces systèmes particulièrement attrayants en vue d'attaques, car ils sont a priori d'une part vulnérables et d'autre part, n'étant pas supervisés, l'attaque ne sera pas forcément découverte. Ils peuvent typiquement être la cible de vers comme ceux à l'origine de l'attaque « Mirai » visant des objets connectés à internet dont le mot de passe constructeur n'a pas été modifié à l'installation.

Nous avons donc utilisé les outils précédemment décrits en vue de concevoir un système entièrement émulé représentant une maison intelligente complète, composé de nombreux objets connectés assemblés selon la topologie suivante :

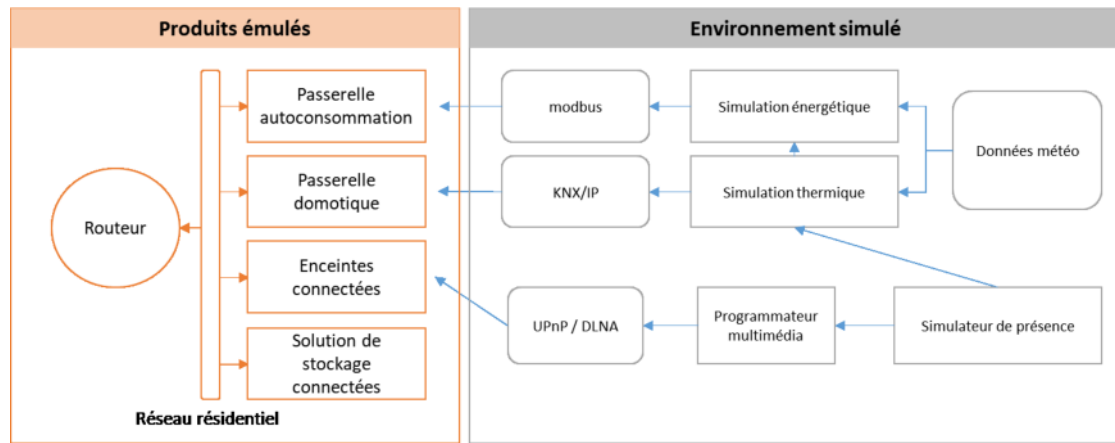


Figure 2. Topologie fonctionnelle du système émulé composé de produits grand public en orange, émulés à partir de leur micro-logiciel et couplés avec différents sous-systèmes physiques simulés afin de produire une interaction forte.

Dans ce cas d'usage, nous cherchons à émuler les systèmes cyber physiques afin de proposer une forte interaction entre le système émulé et l'utilisateur du système en l'occurrence l'attaquant. L'interaction est apportée par la mise en œuvre de simulation de sous-systèmes propres à chaque objet. Par exemple, un objet émulé permettant de contrôler un système de chauffage sera associé à un simulateur thermique persistant.

3.1. Émulation d'un système de gestion du chauffage

Le système domotique choisi pour ce cas est basé sur le protocole KNX. Ce protocole est très utilisé dans l'industrie car il facilite la communication entre systèmes d'automatismes dits à « intelligence répartie ». Le produit central émulé est une passerelle du commerce ayant une interface Ethernet ainsi que des interfaces séries pour les bus de terrain. Son micro-logiciel est disponible sur le site du constructeur et utilise un système compatible avec notamment une plateforme matérielle ARM ainsi qu'un système d'exploitation basé sur Linux.

Le réseau domotique est un bus à état ; c'est-à-dire que les valeurs fournies par ce bus représentent un état physique de l'installation. Pour recréer au mieux un système de gestion, nous avons émulé le comportement thermique d'une maison en utilisant les modèles prédictifs dits « RC » [20]. Dans ce type de modèle, chaque source de chaleur est vue comme une source de tension, chaque isolant est vu comme une résistance et chaque masse thermique est vue comme une capacité. Des interrupteurs permettent d'activer ou non des branches du circuit en fonction de l'état des automates.

Ce modèle constitue la logique nécessaire à la simulation de l'installation. L'interaction avec l'utilisateur se fait via un bus virtuel permettant d'implémenter des automates sous forme de programme informatique [21]. Ce programme intègre le modèle thermique et en gère les parties actives, par exemple l'état haut ou bas des stores correspond à l'état ouvert ou fermé d'une branche active du modèle, soit apportant de l'énergie soit en l'atténuant. Le modèle comportemental de chaque automate, c'est-à-dire la machine à états ainsi que les temps de séquences sont inclus dans le standard [22] et permettent de s'approcher du modèle comportemental réel des objets physiques. Les données extérieures, telles que l'ensoleillement et la température, sont données par un *webservice* météo et permettent elles-aussi d'ajuster pas à pas le modèle thermique, apportant ainsi à l'attaquant une interaction aussi réaliste et persistante que possible.

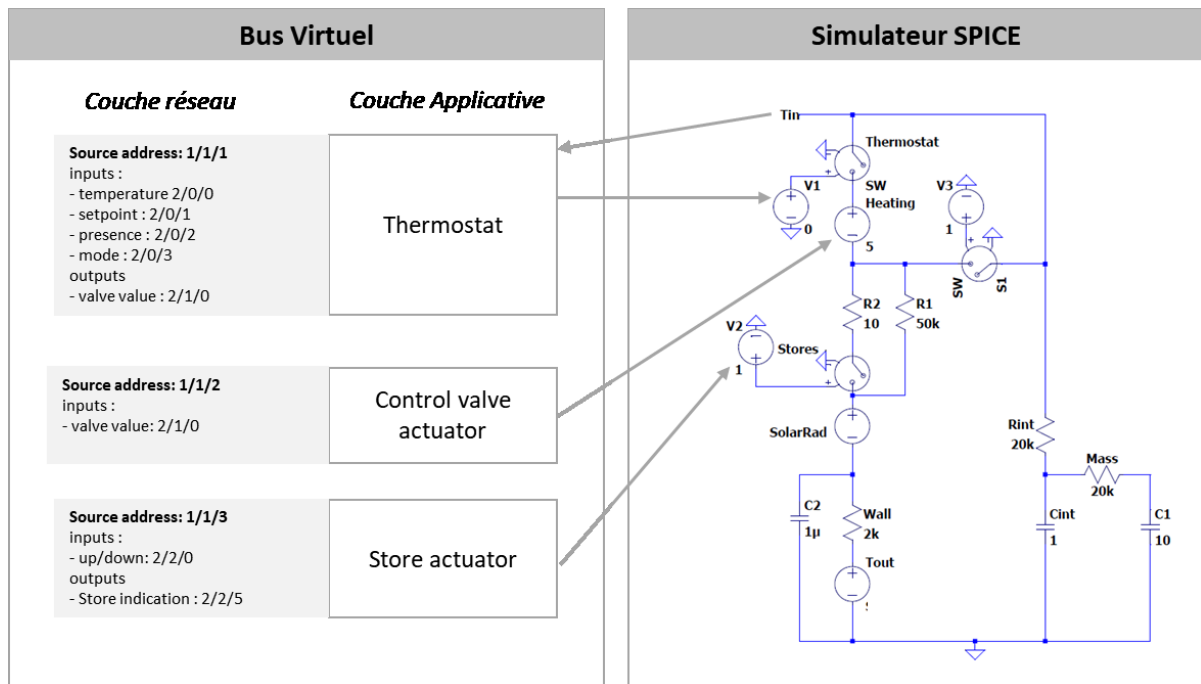


Figure 3. Modèle de simulation analogue au comportement thermique d'un logement.

Sur la Figure 3 nous voyons le modèle de simulation où les générateurs V1, V2 et V3 correspondent à un état binaire du bus virtuel. Le bus virtuel est composé de processus applicatifs permettant de reproduire en langage riche les machines à états d'objets physiques. Ces processus applicatifs sont adressés via une couche réseau propre au protocole utilisé, reproduisant fidèlement les échanges sur la couche réseau avec notamment le plan d'adressage. Les valeurs des générateurs Tout et SolarRad sont données par un webservice météo. Au final, le résultat du modèle est la température intérieure (Tin) générée à partir de conditions externes, mais également à partir de l'interaction humaine reçue par des commandes du bus.

3.2. Émulation d'objets connectés natifs IP

De plus en plus d'objets qui étaient historiquement des périphériques d'un ordinateur sont devenus directement connectés au réseau résidentiel, que ce soit en filaire ou en wifi. En utilisant les technologies citées en section 2, nous avons ainsi pu identifier des écosystèmes d'objets fonctionnant par groupe, typiquement des enceintes connectées et des systèmes de stockage en réseau (NAS).

Pour l'émulation du stockage en réseau, nous avons ciblé spécifiquement un produit commercial connu dont le micro-logiciel était disponible publiquement. Sans protection particulière, nous avons utilisé la méthodologie proposée par le framework Firmadyne puis avons manuellement configuré une machine virtuelle propre à la topologie du produit physique : architecture du processeur (PowerPC) ainsi que contrôleur RAID pour la gestion de disques multiples.

Pour l'émulation d'enceintes connectées, nous avons utilisé deux approches. En premier lieu, les mises à jour des produits ciblés se faisaient de manière authentifiée et chiffrée. La méthode d'authentification était relativement faible et nous avons pu obtenir du serveur de mises à jour plusieurs micro-logiciels mais ces derniers étaient chiffrés. Nous avons donc utilisé une extraction du contenu de la mémoire via une connexion JTAG telle que présentée au paragraphe 2.1. En complément, durant la phase de découverte de mise à jour, nous avons remarqué que les versions les plus anciennes de ces produits n'utilisaient pas le micro-logiciel chiffré, nous permettant ainsi une émulation directe, sans manipulation physique.

Ces deux classes d'objets sont pilotables nativement via le protocole UPnP/DLNA, permettant à une personne extérieure au système de découvrir ses objets connectés et d'interagir avec eux via ce standard. En complément, nous implémentons un simulateur de présence simplifié à partir des recherches proposées par [23] permettant de programmer des diffusions en fonction de la présence supposée et de l'heure de la journée, par exemple des émissions d'informations le matin et des programmes musicaux le soir.

3.3. Émulation d'un système d'autoconsommation

Les systèmes d'autoconsommation deviennent également connectés et certains ont déjà fait l'objet de publication de vulnérabilités, tel que CVE-2019-13529. Ces systèmes sont destinés à la gestion de la production

et éventuellement au stockage d'énergie à l'échelle d'une maison. Comme pour le système domotique décrit en 3.1, deux réseaux de communications peuvent être sollicités : un bus de terrain pour la synchronisation des objets et une passerelle pour la supervision du système depuis l'extérieur.

Nous avons utilisé la même méthodologie qu'en section 3.1. Nous avons modélisé le fonctionnement d'une installation en autoconsommation à partir d'un jeu de données issu d'une expérimentation passée [24] et produit un automate virtuel communicant sur un bus de terrain Modbus. Des travaux sont en cours afin d'extraire le micro-logiciel de la passerelle système ou d'en cloner l'applicatif (interface web).

3.4. Émulation d'un routeur neufbox

Afin d'émuler le réseau IP résidentiel, nous avons utilisé le micro-logiciel d'un routeur typique d'une *box* ADSL, basé sur la distribution OpenWRT. Nous avons ciblé un produit dont le micro-logiciel est disponible en open-source : une *neufbox* de l'ancien fournisseur d'accès à internet « Neuf ». Les sources de ce micro-logiciel sont en effet toujours disponibles [25] bien que l'opérateur ait cessé son activité. L'intérêt d'utiliser un micro-logiciel open-source est de pouvoir émuler les différentes interfaces réseau spécifiques à un routeur de type résidentiel, avec notamment le modem ADSL, le point d'accès WiFi et le réseau téléphonique en utilisant des drivers particuliers au sein du noyau, comme évoque dans la section 2.2.

Pour parfaire l'émulation de ce produit et du réseau résidentiel qu'il gouverne, nous avons sensiblement modifié quelques paramètres notamment au niveau des couches réseau afin d'apporter plus de réalisme à l'ensemble du système émulé. Nous avons en particulier introduit deux réglages permettant de rendre plus crédible la simulation du réseau résidentiel : 1) l'ajout d'un taux d'erreurs aléatoires et uniquement sur un seul des 4 ports physiques du switch émulé en vue de simuler un périphérique probablement défectueux et 2) l'ajout arbitraire de latences sur les connexions physiques internes afin d'introduire des délais propres à un canal de propagation physique.

4. Conclusion

Nous proposons dans cet article différentes techniques permettant d'émuler des objets connectés, illustrées par un cas d'usage : la maison connectée. Ces techniques sont issues de divers domaines, en particulier pour les outils d'analyse de micro-logiciels. Cette émulation de micro-logiciel est associée à des simulateurs d'environnements afin de permettre la composition de systèmes à interaction forte. Ces systèmes composés peuvent servir ensuite à la formation par la simulation d'attaques (cyber-ranges) ou à l'exploitation de pots de miels.

5. Remerciements

Ces travaux ont reçu du financement par le programme national français Programme d'Investissements d'Avenir, IRT Nanoelec ANR-10-AIRT-05. Ces travaux ont également reçu du financement de la part de la Commission Européenne dans le cadre du projet H2020 M-Sec. Le projet M-Sec est financé conjointement par le programme de recherche et d'innovation Horizon 2020 de l'Union Européenne (contrat n° 814917) et par la commission de recherche de l'institut national des technologies de communications et de l'information (NICT) du Japon (contrat n° 19501)

6. Références

- [1] Shodan, «The search engine for Internet-connected devices,» [En ligne]. Available: <https://shodan.io/>. [Accès le 23 06 2020].
- [2] D. D. Chen, M. Egele, M. Woo et D. Brumley, «Towards Automated Dynamic Analysis for Linux-based Embedded Firmware,» *Network and Distributed System Security Symposium*, 2016.
- [3] DeepBitsTechnology, «FirmPin,» 2018. [En ligne]. Available: <https://github.com/DeepBitsTechnology/FirmPin>. [Accès le 23 06 2020].
- [4] J. Zaddach, L. Bruno, A. Francillon et D. Balzarotti, «AVATAR: A Framework to Support Dynamic Security Analysis of Embedded Systems' Firmwares,» *NDSS*, vol. 14, pp. 1-166, 2014.
- [5] A. Jakhar, «EXPLIoT,» 2018. [En ligne]. Available: <https://explot.readthedocs.io/en/latest/index.html>. [Accès le 23 06 2020].

- [6] OnwardSecurity, «Automated Vulnerability Assessment Tool HERCULES SecDevice,» [En ligne]. Available: https://www.onwardsecurity.com/product_cate/item/29. [Accès le 23 06 2020].
- [7] T. Maurin, L. F. Ducreux, G. Caraiman et P. Sissoko, « IoT security assessment through the interfaces P-SCAN test bench platform,» chez DATE, Dresden, 2018.
- [8] ReFirmLabs, «Binwalk,» 2013. [En ligne]. Available: <https://github.com/ReFirmLabs/Binwalk>. [Accès le 23 06 2020].
- [9] F. Bellard et e. al., «QEMU, the FAST! processor emulator,» 2011. [En ligne]. Available: <https://qemu.org>. [Accès le 23 06 2020].
- [10] Fraunhofer FKIE, «<https://github.com/fkie-cad/FirmwareScraper>,» [En ligne]. Available: <https://github.com/mellowCS/FirmwareScraper>.
- [11] «Fraunhofer FKIE Embedded Device Evaluation Corpora,» [En ligne]. Available: <https://github.com/fkie-cad/embedded-evaluation-corpus>.
- [12] V. Gopal et L. Wonjun, «Exploiting JTAG and Its Mitigation in IOT: A Survey».
- [13] S. Vasile, D. Oswald et T. Chothia, «Breaking All the Things—A Systematic Survey of Firmware Extraction Techniques for IoT Devices,» chez CARDIS, 2018.
- [14] Y. L. Chui et M. M. Wan, «Evilsploit – A Universal Hardware Hacking Toolkit,» *Blackhat*, 2017.
- [15] «JTAGulator,» [En ligne]. Available: <http://www.grandideastudio.com/jtagulator/>.
- [16] xPack\$, «xPack QEMU Arm,» [En ligne]. Available: <https://xpack.github.io/qemu-arm/>.
- [17] NIST, «What are Cyber Ranges?,» 2018. [En ligne]. Available: https://www.nist.gov/system/files/documents/2018/02/13/cyber_ranges.pdf.
- [18] «OpenStack,» [En ligne]. Available: <https://www.openstack.org/>.
- [19] S. Mocanu, M. Puys et P.-H. Thevenon, «An Open-Source Hardware-In-The-Loop Virtualization System for Cybersecurity Studies of SCADA Systems,» chez C&esar - *Virtualization and Cybersecurity*, Rennes, 2019.
- [20] E. Atam et L. Helsen, «Control-oriented thermal modeling of multizone buildings: methods and issues: intelligent control of a building system,» *IEEE Control Systems Magazine*, vol. 36, n° %13, pp. 86-111, 2016.
- [21] M. Gallissot et F. Jambon, «Proposition et mise en œuvre d'un modèle d'interopérabilité pour le bâtiment intelligent,» *Ingénierie des systèmes d'information*, vol. 17, n° %12, pp. 1211-42, 2012.
- [22] KNX Association, «Chapter 7 - "Application Descriptions",» chez *The KNX Standard*.
- [23] D. R. N. M. J.-L. S. J. Page, «A generalised stochastic model for the simulation of occupant presence,» *Energy and Buildings*, Vols. %1 sur %240, Issue 2,, pp. 83-98, 2008.
- [24] I. Nanoelec, «Vivre en autonomie énergétique avec le projet Reli2,» 2016. [En ligne]. Available: <https://www.youtube.com/watch?v=OhQTU10c5uM>.
- [25] StidOfficial, «Sources neufbox,» [En ligne]. Available: <https://github.com/StidOfficial/neufbox>.