MASTER EN SCIENCES INFORMATIQUES

Élaboration d'une méthodologie d'évaluation de la sécurité des objets connectés

Favereaux, François

Award date:
2021

Awarding institution:
Universite de Namur

Université de Namur
Faculté d'informatique
Année académique 2020-2021

Elaboration d'une méthodologie d'évaluation de la sécurité des objets connectés

FAVEREAUX François

Promoteur : _____ (Signature pour approbation du dépôt - REE art.40)
COLIN Jean-Noël

Mémoire présenté en vue de l'obtention du grade de
Master en Sciences Informatiques

Résumé

L'IoT (Internet of things) est une dénomination qui reprend un grand nombre d'objets qui diffèrent par leur taille, leur poids, leur fonctionnalité, etc. Bien que pouvant être forts différents, les objets IoT possèdent cependant une série de propriétés en commun : ils sont connectés à internet, ils fournissent des services qui améliorent ou facilitent notre quotidien et, ce sont des capteurs ou des actuateurs. Les montres connectées, les ampoules intelligentes, certains défibrillateurs cardiaques, Google home et bien d'autres font partie des équipements IoT.

De plus en plus de domaines d'application différents utilisent les équipements IoT et leur nombre ne cesse d'augmenter d'année en année. Pourtant, même si ce domaine est en plein essor, de nombreux manquements sont relevés au point de vue sécurité informatique. Ce travail a pour but de proposer une méthodologie d'évaluation de la sécurité IoT.

Pour ce faire, nous présentons brièvement certaines notions théoriques sur les systèmes IoT et sur les méthodes d'évaluation dédiées aux systèmes informatiques standards. Nous présentons ensuite les vulnérabilités présentes dans les équipements IoT, les attaques auxquelles ils font face et les méthodes d'évaluation qui existent actuellement pour attester de leur sécurité. Nous proposons enfin une méthode d'évaluation de la sécurité que nous appliquerons sur des exemples pratiques.

Mots clés : IoT, évaluation sécurité, méthodologie, vulnérabilités, attaques.

Préface

Nous avons choisi de rédiger ce travail en français. Certains termes techniques, propres au domaine IoT et au domaine de la sécurité informatique, n'ont cependant pas été traduits de l'anglais vers le français. Nous estimons que ces dénominations techniques font partie du paysage et doivent apparaître en anglais.

Remerciements

J'aimerais remercier sincèrement

Mr Jean-Noël COLIN
promoteur de ce mémoire,
pour sa disponibilité, son suivi et ses précieux conseils.

Vincent BRETTON
mon employeur,
pour avoir fait de mes études une priorité.

Pierre NENNEN
pour ses commentaires avisés.

Cédric et Jessica
pour leurs relectures attentives.

Claire
pour son indéfectible soutien lors de ce Master.

Ma famille et mes amis
qui n'ont cessé de m'encourager.

Table des matières

Liste des tableaux

Table des figures

Chapitre 1

Introduction

1.1 IoT - Les objets connectés

Il nous parait tout d'abord opportun de définir le terme IoT. Cette dénomination IoT (Internet of Things) serait apparue pour la première fois en 1999 lors d'un discours donné par Kevin Ashton, un informaticien londonien [38]. Ashton a envisagé en 1997 la possibilité d'utiliser des étiquettes d'identification par radiofréquence (RFID) pour suivre les produits tout au long de la chaîne d'approvisionnement de la société Procter and Gamble. Les étiquettes RFID étaient utilisées pour lire et identifier des objets, puis pour transmettre les informations par un réseau sans fil. Il y a encore quelques années, le terme IoT était utilisé pour désigner un système où les objets étaient connectés à internet. De nos jours, cette définition a évolué et de nouveaux équipements dans les systèmes IoT sont apparus comme par exemple, des capteurs GPS, des smartphones, le cloud computing etc. Même si le terme IoT est fréquemment utilisé, il n'est cependant pas facile de trouver une définition formelle et générale.

Plusieurs organismes reconnus - tels que l'*European Telecommunications Standards Institute* (ETSI), l'*International Telecommunication Union* (ITU) , *the Internet Engineering Task Force* (IETF), *the National Institute of Standards and Technology* (NIST) - ont donné leur propre définition de ce qu'est l'IoT. Bien que similaires, ces définitions proposent des concepts parfois différents. L'ETSI parle de *"machine-to-machine"*, l'ITU parle *"d'infrastructure mondiale pour la société de l'information"*, l'IETF parle *"d'objets composés d'électronique, de software, de capteurs et d'actuateurs"*, etc. En synthétisant et en vulgarisant ces différentes définitions, on pourrait définir l'IoT comme suit : l'IoT est un système composé d'objets connectés qui peuvent communiquer entre eux en temps réel via une plateforme située dans le cloud afin de permettre une collecte ou un échange de données issues du monde réel. L'objectif de ce système étant d'offrir des services uniques aux personnes afin d'améliorer leur qualité de vie.

1.2 Diversité des systèmes d'objets connectés

Maintenant que nous avons défini le terme d'IoT, nous allons découvrir dans cette section qu'il existe une multitude de systèmes d'objets connectés différents. Le nombre de domaines d'application de l'IoT a connu une croissance exponentielle au cours des dernières années [69].

En effet, on retrouve les équipements IoT dans des domaines divers et variés tels que les milieux industriels, les systèmes de transport routier, l'agriculture, la médecine, le sport etc. La figure 1.1 propose quelques exemples concrets d'utilisation de ces équipements :

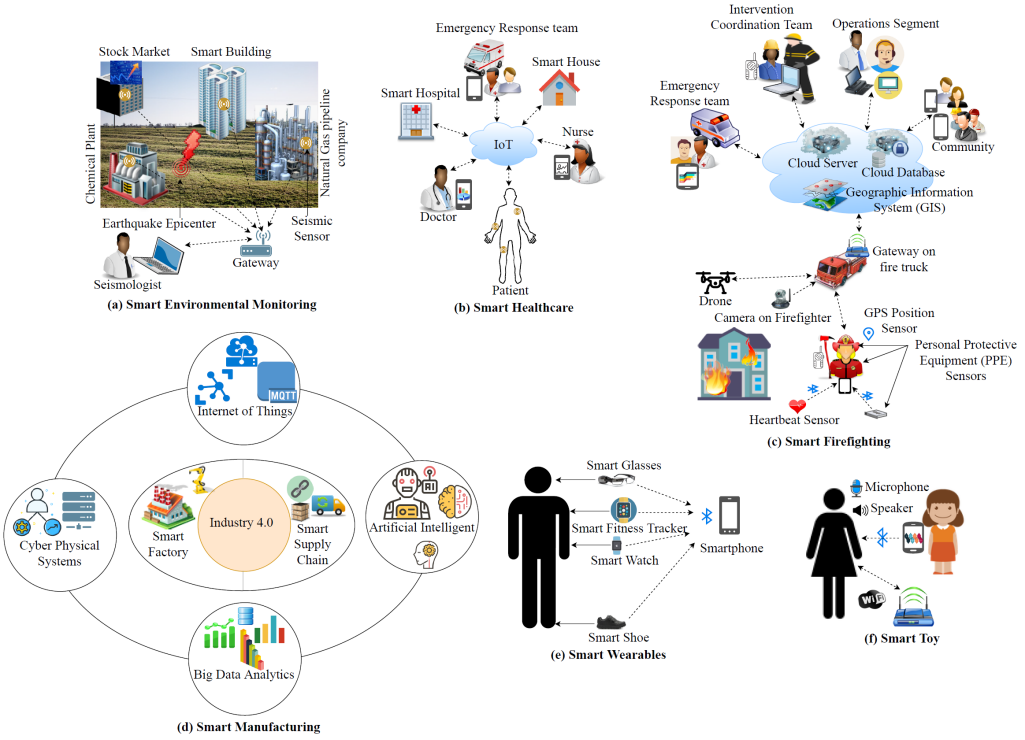


FIGURE 1.1 – Domaines d'application IoT [69]

- "Smart environmental monitoring" : dans ce domaine, la majorité des équipements IoT est utilisée pour effectuer le monitoring de la qualité de l'eau et de l'air. Certaines utilisations un peu plus atypiques existent néanmoins. On peut, par exemple, détecter certaines catastrophes naturelles telles que des tremblements de terre ou des glissements de terrain. Dans le cas des tremblements de terre, les équipements IoT peuvent, après détection, mettre à l'arrêt des installations critiques (usines chimiques, data center, etc.), prévenir les personnes éloignées de l'épicentre et obtenir une estimation détaillée des dommages ;
- "Smart healthcare" : Niewolny [61] décrit *le smart healtcare* comme suit : *"Actuellement, il existe des dispositifs médicaux implantables, minuscules et portables, qui détectent, enregistrent, analysent et transmettent en permanence des informations sur la santé des patients via Internet, auxquelles leurs médecins peuvent accéder à tout moment et en tout lieu"* ;
- "Smart firefighting" : les technologies actuelles permettent de récolter un grand nombre de données sur un incendie en cours. Ces données, une fois analysées, permettent de prédire l'évolution de l'incendie, d'aider à l'évacuation des personnes et d'assurer la sécurité des pompiers ;
- "Smart manufacturing" : l'IoT est au coeur de l'industrie 4.0. Ce nouveau paradigme, qui guide l'industrie de demain, se base sur l'analyse des données, l'intelligence artificielle, l'IoT et les systèmes cyber-physiques (CPS), dans le but de réduire les coûts et les déchets et afin de booster l'innovation ;

- "Smart wearable" : des petits objets (composés de capteurs et/ou d'actionneurs) peuvent être placés autour du cou, au poignet, sur le corps, afin d'apporter certaines fonctionnalités dans des domaines tels que le sport, la santé, l'armée, etc. ;
- "Smart toys" : certains jouets sont équipés de haut-parleurs, de microphones, d'ordinateurs intégrés et de batteries. Ils peuvent être connectés à internet via le WiFi. Ces jouets ont la capacité d'écouter et d'enregistrer les paroles des enfants et d'envoyer ces informations dans le cloud afin qu'elles soient traitées par des services de reconnaissance vocale. Le jouet peut alors répondre à l'enfant ou envoyer un email reprenant la discussion aux parents.

En plus de domaines d'application et d'usages variés, les équipements IoT diffèrent fortement les uns des autres. Ils utilisent un grand nombre de composants (actuateurs, capteurs, bibliothèques logicielles, etc.) et de médias de communication (filaire, bluetooth, WiFi, rfid, etc.) différents. C'est donc un euphémisme que de dire que les systèmes IoT et les équipements IoT eux-mêmes sont divers et variés.

1.3 Problèmes de sécurité des objets connectés

Comme nous allons le voir dans cette section, même si les systèmes IoT font désormais partie du quotidien et qu'ils sont au centre de systèmes parfois importants et critiques, de nombreux manquements au niveau sécurité ont été relevés.

On estime qu'en 2018, environ 30 milliards d'équipements IoT étaient déployés à travers le monde [13]. Même si les estimations du nombre d'équipements IoT et le taux de croissance du nombre d'IoT ne s'accordent pas toujours, une chose est certaine, ce nombre ne cesse de grandir chaque jour. De récentes études ont démontré que de nombreux équipements IoT étaient peu ou pas sécurisés et que certains avaient déjà été la cible d'attaques. On peut par exemple citer les problèmes de sécurité :

- liés aux pacemakers [70] ou aux pompes à insuline [67] ;
- dans le domaine de la mobilité avec les voitures autonomes [35] ;
- dans le domaine de la domotique avec les serrures intelligentes [30].

Ces problèmes de sécurité sont majoritairement dûs à deux facteurs : la conception des équipements IoT et la nature des infrastructures des systèmes IoT.

Comme expliqué précédemment, les équipements IoT apportent un grand nombre d'avantages, tant sur le plan humain et social, que sur le plan socio-économique. L'effervescence et l'engouement autour des équipements IoT fut tel, que les sociétés qui conçoivent ces équipements ont souvent privilégié le nombre ou le type de fonctionnalités ainsi que la rapidité de production et de mise sur le marché afin de concurrencer les sociétés rivales. L'aspect sécurité informatique fut alors mis de côté.

Un autre facteur qui complique la mise en place d'une sécurité robuste, est la particularité des systèmes IoT. Comme expliqué par Denning *et al.* : *"Un système IoT est la combinaison de 4 facteurs principaux : Un système IoT est (1) un environnement extrêmement personnel, rempli d'actifs informationnels, où (2) il n'y a pas d'administrateur professionnel et dévoué pour maintenir une (3) collection hétérogène de technologies de consommation qui (4) sont de plus*

en plus cyber-physiques et riches en capteurs" [27]. La combinaison de ces facteurs complique la conception de mécanismes de sécurité.

1.4 Problématisation

Cet engouement grandissant autour des IoT et ces problèmes de sécurité mentionnés précédemment nous amènent donc à nous questionner sur les solutions déjà existantes et sur les moyens d'attester de la sécurité d'un équipement IoT. Nous l'avons vu, les équipements IoT sont forts différents les uns des autres et font partie de systèmes complexes qui diffèrent des systèmes informatiques standards. Les méthodes connues et maîtrisées d'évaluation de la sécurité des systèmes informatiques conventionnels ne s'appliquent donc pas parfaitement aux équipements IoT.

Ce constat nous amène donc à nous poser la problématique suivante : "Comment peut-on concevoir une méthode générique d'évaluation de la sécurité d'un équipement IoT alors que ces derniers utilisent des technologies et des composants différents, le tout pour des usages et domaines d'application forts variés ?"

1.5 Motivation de la recherche

Il est intéressant de noter que nous n'avons pas trouvé, dans la littérature scientifique, de travaux proposant une méthodologie d'évaluation de la sécurité des objets connectés. Par contre, nous avons trouvé certains articles et travaux qui traitent de problèmes de sécurité liés aux équipements IoT. Certains articles recensent par exemple l'exploitation de failles et de différentes vulnérabilités connues propres aux IoT. D'autres articles proposent des méthodes pour vérifier que les équipements sont résistants à certaines attaques précises. Enfin, des groupes de chercheurs ont même proposé des bancs de test, plus ou moins aboutis, permettant d'évaluer la sécurité des équipements. Malgré tous ces travaux, personne n'a encore tenté de proposer une méthodologie générale et complète pour évaluer de façon plus globale la sécurité des IoT.

Ce travail permettrait premièrement de pallier ce manquement et deuxièmement, de fournir une base de travail pour des recherches futures.

Afin de répondre à cette question de recherche, nous allons premièrement amener certains concepts importants sur les systèmes IoT, leur architecture, les contraintes auxquelles ils sont liés, les exigences de sécurité qu'ils doivent rencontrer, etc. Nous décrirons également les méthodologies existantes permettant d'évaluer la sécurité d'un système informatique. Nous étudierons ensuite les vulnérabilités et les attaques qui ciblent les équipements IoT. Nous présenterons alors les travaux les plus récents permettant de tester, à l'aide de bancs de test, la sécurité des équipements. Sur base de ces différentes informations, nous élaborerons et détaillerons notre méthodologie d'évaluation de la sécurité IoT. Nous terminerons ensuite par une mise en pratique sur des équipements IoT.

Première partie

Notions théoriques

Chapitre 2

Systèmes IoT

2.1 Architecture des systèmes IoT

Avant de pouvoir proposer une méthodologie d'évaluation de la sécurité des objets connectés, il faut maîtriser certaines notions théoriques de base sur les équipements IoT. Nous les présentons dans cette section.

Un système IoT peut être décrit comme une architecture orientée services (*service-oriented architecture* - SOA) : *"en tant que technologie clé dans l'intégration de systèmes ou de dispositifs hétérogènes, SOA peut être appliquée pour implémenter les systèmes IoT"* [...] *"D'un point de vue technologique, un système IoT doit répondre à des besoins de modularité, d'extensibilité, d'évolutivité et d'interopérabilité entre les équipements hétérogènes"*[26]. SOA est donc une bonne approche qui permet de répondre à ces besoins [9]. Plusieurs modèles différents ont déjà été présentés pour représenter les systèmes IoT, on peut par exemple citer :

- Atzori *et al.* [9],Domingo [31], Jia *et al.* [45] et Mahmoud *et al.* [54] qui proposent de diviser l'architecture IoT en **3 couches** : la couche de perception, la couche réseau et la couche de services (ou couche d'application) ;
- Chen *et al.* [22] qui proposent un modèle d'architecture composé de **4 couches** : la couche physique, la couche de transport, la couche middleware et la couche application.
- Liu *et al.* [51] qui proposent un modèle d'architecture composé de **4 couches** : la couche physique, la couche de transport, la couche service et la couche application.
- l'ITU (International Telecommunication Union) qui propose un modèle d'architecture composé de **5 couches** : la couche de détection, la couche d'accès, la couche de mise en réseau, la couche middleware et la couche d'application ;
- Khan *et al.* [47] qui proposent un modèle d'architecture composé de **5 couches** : la couche de perception, la couche network, la couche middleware, la couche application et la couche business.

Nous avons choisi d'utiliser, dans ce travail, le modèle trois couches car premièrement, il est majoritairement utilisé dans la littérature, et deuxièmement, il correspond bien à nos besoins (besoins qui seront expliqués ultérieurement). Ce modèle nous servira, entre autres, à lister les

attaques et les vulnérabilités en fonction des différentes couches. Nous n'avons pas jugé pertinent d'utiliser les autres modèles, vu que les attaques ciblant les équipements IoT concernent majoritairement les 3 couches du modèle (perception, réseau et application).

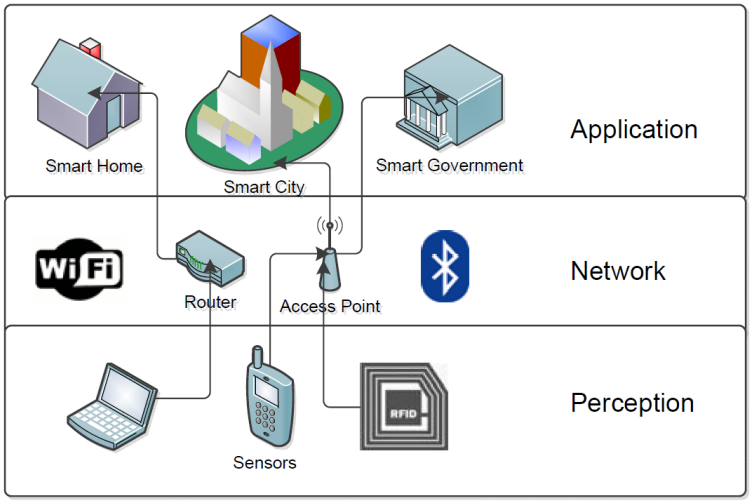


FIGURE 2.1 – Architecture IoT 3 couches [54]

La figure 2.1 présente, d'un point de vue fonctionnel, l'architecture trois couches que l'on utilisera tout au long du travail. Les différentes couches qui composent cette architecture sont décrites ci-dessous :

- 1. Perception : elle contient les équipements de terrain connectés au monde réel. C'est à ce niveau qu'à lieu la récolte des données par les capteurs et la réalisation de certaines opérations par les actuateurs. On retrouve donc dans cette couche des montres intelligentes, des capteurs GPS, des caméras, etc. ;
- 2. Network : elle permet aux différents équipements de communiquer entre eux. Les communications s'effectuent via des réseaux sans fil (WIFI, bluetooth, zigbee, Sigfox, Lora, etc.) ou via des réseaux filaires (ethernet, USB, RS-485, etc.) ;
- 3. Application : Khan *et al.* [50] décrivent la couche application comme suit : *"la couche application reçoit les données transmises par le réseau et utilise les données pour fournir les services ou les opérations nécessaires. Par exemple, la couche application peut fournir le service de stockage pour sauvegarder les données reçues dans une base de données, ou fournir le service d'analyse pour évaluer les données reçues afin de prédire l'état futur des dispositifs physiques"*. Cette couche est orientée service (SOA) [47].

2.2 Types d'équipements IoT

Afin de bien comprendre les problèmes de sécurité que peuvent rencontrer les systèmes IoT, il est nécessaire de décrire les différents types d'équipements qui font partie de ces systèmes ainsi que leurs interactions entre eux. Hossain *et al.* [42] distinguent cinq types d'équipements différents, représentés sur la figure 2.2 :

- *IoT device* : il s'agit d'un terme générique qui reprend toute une série de composants tels que des capteurs, des actionneurs, des interfaces de communication, des operating systems, des systems software, des services légers... Le rôle principal de ces objets est de collecter des informations grâce aux capteurs et d'effectuer des actions grâce aux actionneurs ;
- *Coordinator* : cet objet gère une série d'éléments. Son rôle premier est de gérer la santé des équipements ainsi que leurs interactions. Il permet également d'envoyer des rapports d'évènements au fournisseur de services IoT ;
- *Sensor bridge* : il agit comme un hub entre le réseau local et le cloud où se trouvent les services. Il agit également comme un connecteur entre des réseaux locaux différents ;
- *IoT services* : ce sont des services qui permettent, entre autres, la prise de décision, la gestion des équipements et l'automatisation de certains processus. Ils sont, la plupart du temps, hébergés dans un cloud ;
- *Controllers* : ils commandent les équipements IoT (un utilisateur qui envoie une commande via un smartphone, par exemple).

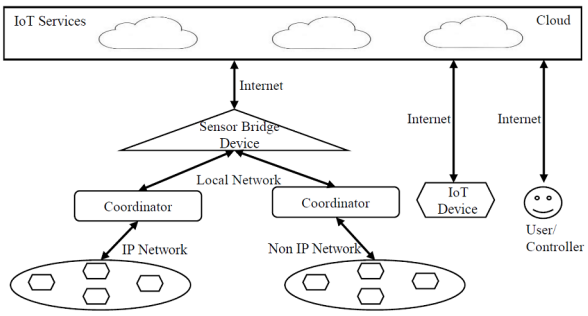


FIGURE 2.2 – Types d'équipements composant un système IoT [42]

FIGURE 2.3 – Caractéristiques des réseaux IoT [36]

2.3 Types de communication IoT

Nous pouvons différencier plusieurs types de réseaux IoT [36] :

- WAN (Wide Area Network) : c'est un réseau de plusieurs dizaines de kilomètres carrés ;
- LPWAN (Low Power Wide Area Network) : c'est un réseau de plusieurs dizaines de kilomètres carrés qui utilise peu de bande passante ;
- PAN (Personal Area Network) : c'est un réseau de quelques mètres ;
- LAN (Local Area Network) : c'est le réseau privé d'une entreprise ou d'un domicile ;
- Satellite : c'est un réseau accessible, en théorie, partout dans le monde.

La figure 2.3 compare les différents types de réseaux IoT en fonction de leur portée, leur débit et leur autonomie.

Dans les types de réseaux présentés ci-dessus, on retrouve des protocoles de communication employés par les équipements IoT qui sont assez différents les uns des autres. Certains équipements utilisent des connexions câblées afin d'échanger des données tandis que d'autres utilisent des connexions sans fil. Dans cette section, nous nous intéressons aux connexions sans fil car elles sont de loin les plus problématiques d'un point de vue de la sécurité [3]. Nous présentons ci-dessous une liste non exhaustive de ces protocoles :

- Sigfox : ce protocole de communication permet d'échanger des petites quantités de données sur des distances allant jusqu'à cinquante kilomètres. Il convient assez bien aux IoT de faible énergie ;
- cellular : ce protocole permet des communications sur de longues distances et avec un haut débit via la 3/4/5G. Elle est cependant consommatrice d'énergie ;
- 6loWPAN : c'est un des protocoles les plus répandus. Il permet à l'équipement IoT d'obtenir une adresse IP et de se connecter sur un réseau informatique standard. Il est peu gourmand en énergie et consomme peu de bande passante. Il supporte les topologies maillées et en étoile ;
- Bluetooth : la portée de ce protocole est assez faible, généralement 10 à 15 mètres. Il n'est cependant pas très consommateur d'énergie et permet des topologies en étoile ;
- Zigbee : ce protocole consomme très peu d'énergie, il convient bien à des équipements sur batterie. La portée d'un capteur ne dépasse pas plus d'une dizaine de mètres. Toutefois, vu que les équipements peuvent envoyer des données via d'autres équipements du réseau, on peut facilement atteindre une distance de communication de 100 mètres. Ce protocole permet des topologies en étoile, en arbre ou en maille ;
- RFID : on retrouve deux types d'équipements, les *transponder* et les *reader*. Les premiers contiennent les données (les données sont statiques et uniques), les deuxièmes lisent les données contenues dans les premiers. Les distances de communication varient en fonction du type de matériel (cela peut varier de quelques centimètres à plusieurs dizaines de mètres) ;
- NFC : ce protocole est assez similaire au RFID. On retrouve des *transponder* et des *reader*. La distance de communication est très faible, on parle de quelques centimètres maximum. Contrairement au RFID, les données contenues dans les *transponder* peuvent être réécrites ;
- Zwave : ce protocole se retrouve principalement dans des applications domotiques. Il permet des communications jusqu'à 30 mètres en utilisant des vitesses ne dépassant pas 100kbps. On retrouve deux types d'équipements différents, les maîtres et les esclaves. Ces derniers ne peuvent qu'exécuter et répondre aux requêtes envoyées par les maîtres.

Cet inventaire non exhaustif a pour objectif de mettre en évidence la diversité des protocoles de communication utilisés par les équipements IoT. Les communications diffèrent sur de nombreux points : les topologies, les portées, les débits, les technologies, etc.

2.4 Contraintes liées à l'IoT

Contrairement aux équipements informatiques standards (ordinateurs, serveurs, etc.), les équipements IoT se révèlent être moins performants à certains niveaux. En effet, comme l'expliquent

Sha *et al.* *"pour faire baisser les coûts de développement et de fabrication, les fournisseurs d'équipements IoT utilisent souvent des composants de faible capacité"* [75]. Les équipements IoT sont donc limités en terme de ressources et il n'est donc pas simple d'utiliser des mécanismes de sécurité conventionnels [42].

2.4.1 Contraintes hardware

Les composants matériels qui sont utilisés pour concevoir les équipements présentent certains points faibles [42] :

- capacité de calcul : la puissance et la fréquence des cpu ne permettent pas l'utilisation d'algorithmes de chiffrement qui sont coûteux en ressources ;
- source énergétique : les équipements IoT sont parfois amenés à être déplacés, on ne peut donc pas toujours leur fournir une alimentation externe. La plupart des équipements sont donc alimentés par des petites batteries. Les algorithmes trop gourmands en ressources ne peuvent donc pas être utilisés ;
- ram et espace mémoire : les algorithmes de sécurité conventionnels qui fonctionnent sur ordinateur utilisent plus d'espace mémoire et de ram que ne le permettent certains équipements IoT ;
- résistance du packaging : les équipements isolés peuvent être pris pour cible (extraction de données, corruption, etc). Concevoir des packaging résitants est une façon de se prémunir contre ce genre de dangers.

2.4.2 Contraintes Software

Les équipements présentant certains points faibles au niveau software [42] :

- software embarqué : les operating system embarqués possèdent des piles de protocoles réseau assez minces. Le module de sécurité doit donc être léger ;
- développement logiciel : si certains problèmes majeurs sont identifiés, personne n'est responsable de les corriger [40]. Contrairement aux systèmes informatiques standard, la plupart des dispositifs IoT ne sont pas maintenus ni mis à jour par les fabricants [41] ;
- patch de sécurité dynamique : l'installation et la programmation à distance ne sont pas toujours possibles.

2.4.3 Contraintes Réseau

Les équipements présentant certains points faibles au niveau réseau [42] :

- mobilité : les algorithmes de sécurité doivent gérer le fait que des équipements non configurés au préalable intègrent des systèmes déjà fonctionnels ;
- extensibilité / évolutivité : les schémas de sécurité actuels ont du mal à gérer l'extensibilité des réseaux ;
- diversité des équipements : au vu de l'hétérogénéité des équipements qui composent les systèmes IoT, il est difficile de trouver un schéma de sécurité général ;

- diversité des médias de communication : les équipements IoT communiquent via des réseaux sans fil ou filaires qui n'utilisent pas les mêmes protocoles. Il est donc difficile de trouver un schéma de sécurité qui s'appliquerait à chaque type de communication ;
- multi protocoles réseaux : les équipements peuvent utiliser des protocoles réseaux autres que IP et communiquer en même temps avec un fournisseur de services IoT via le réseau IP. Les schémas de sécurité habituels ne se prêtent pas bien à ce genre de fonctionnement ;
- topologie réseau dynamique : on peut ajouter et supprimer à tout moment des équipements. Les schémas de sécurité actuels ne gèrent pas ce mécanisme.

Lors de nos recherches dans la littérature, nous avons à plusieurs reprises lu des articles traitant des besoins d'interopérabilité des équipements IoT. Nous avons été surpris de voir que ce besoin n'était à aucun moment repris comme une contrainte. De plus, nous n'avons pas trouvé d'informations traitant du *roaming* des IoT. Ni en tant que besoin, ni en tant que contrainte.

2.5 Exigences et objectifs de sécurité

Avant de parler des vulnérabilités des équipements IoT et des attaques auxquelles ils doivent faire face, il est intéressant de rappeler les objectifs et exigences de sécurité que doivent rencontrer les dispositifs IoT.

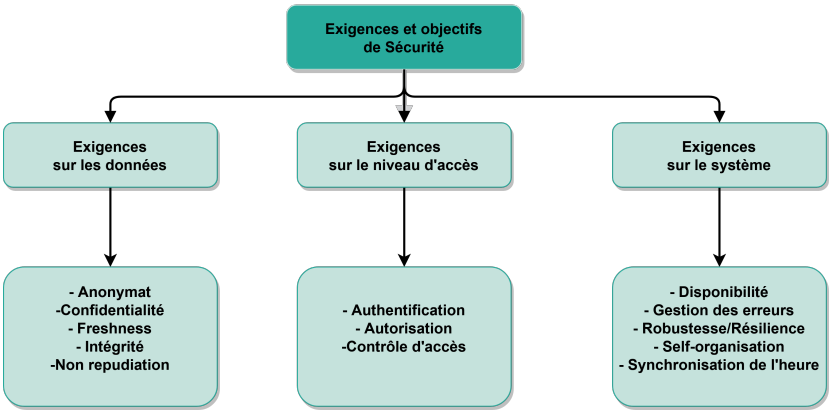


FIGURE 2.4 – Exigences de sécurité IoT

Traditionnellement, seuls la confidentialité, l'intégrité et la disponibilité étaient considérés comme des exigences de sécurité dans les réseaux IoT. [86] :

- Confidentialité : elle permet de garantir le secret des données transmises entre les nœuds du réseau en limitant l'accès aux données aux seuls utilisateurs prévus. *"Elle repose principalement sur l'utilisation de techniques cryptographiques au niveau de la couche physique, où les données sont cryptées au niveau du nœud émetteur pour empêcher la divulgation d'informations à des utilisateurs non autorisés"* [86]. La confidentialité s'applique aussi bien aux données stockées dans les noeuds, qu'aux messages qui transitent entre les noeuds ;
- Intégrité : elle garantit que les données reçues n'ont pas été altérées pendant leur transit [42]. Un adversaire peut intercepter et modifier des messages dans le but de porter atteinte au système IoT ;

- Disponibilité : elle permet d'assurer (et ce à tout moment) l'accès aux services, pour les utilisateurs autorisés.

De nos jours, les systèmes IoT sont intégrés dans des infrastructures parfois plus complexes et parfois plus critiques. Selon Tomić *et al.* [80], de nouvelles exigences de sécurité viennent donc s'ajouter aux précédentes :

- Exigences au niveau des données :
  - anonymat : dissimuler la source des données contribue à la protection et à la confidentialité des informations ;
  - "freshness" : cela permet de garantir que les messages qui transitent sont récents et qu'il n'y a pas de "rejeux" d'anciens messages ;
  - "non-répudiation" : cela permet de garantir qu'un équipement IoT ne peut pas nier avoir reçu ou émis un message.
- Exigences au niveau des accès :
  - authentification : permet à un équipement IoT de garantir l'identité du noeud avec qui il communique. Le noeud récepteur vérifie alors si les données proviennent de la bonne source. L'authentification s'applique aux équipements et aux utilisateurs ;
  - autorisation : permet de garantir que seuls les utilisateurs et équipements autorisés ont accès au réseau et aux services ;
  - contrôle d'accès : permet de garantir qu'un équipement IoT authentifié a uniquement accès aux services pour lesquels il est autorisé à avoir accès.
- Exigences au niveau du système :
  - gestion d'erreur : cela permet de garantir qu'en cas d'attaque ou de défaillance (noeud compromis, noeud détruit, problème hardware, problème software, etc.) le système sera toujours fonctionnel. Cela contribue à la robustesse du système ;
  - robustesse/résilience : cela permet de grantir que le système fonctionnera correctement en cas de compromission de plusieurs noeuds ou en cas d'ajout d'un grand nombre d'équipements sur le réseau ;
  - "self-organisation" : le système peut subir des perturbations telles qu'une attaque ou une défaillance de certains équipements. Le système doit donc être capable de réorganiser son organisation (un équipement peut par exemple choisir une autre route pour acheminer des messages, un équipement peut changer de coordinateurs si la configuration réseau le permet, etc.) ;
  - synchronisation de l'heure : cette exigence peut s'avérer utile dans certains cas, par exemple pour l'économie d'énergie des équipements, pour le calcul du temps de parcours des paquets, etc.

Chapitre 3

Evaluation de la sécurité d'un système informatique

L'évaluation de la "sécurité informatique" est un processus qui permet la découverte des vulnérabilités d'une infrastructure informatique. La découverte des vulnérabilités est également connue sous le nom de *security testing* et comprend un large éventail de techniques visant à vérifier si un système d'information atteint le niveau de sécurité prévu [19]. Parmi ces techniques on retrouve : des tests de pénétration, des *red team exercises*, des scans de vulnérabilités et du *social engineering* [49].

Bien que des outils et techniques existent, les spécialistes se sont rendu compte que ce n'était pas suffisant pour évaluer de manière complète et structurée la sécurité des systèmes informatiques. En proposant juste des outils et techniques, les personnes en charge de la sécurité vont peut-être tester partiellement le système, ou n'utiliser que des outils avec lesquels elles ont le plus d'affinités. Les tests effectués ne seront peut être pas non plus documentés. Deux personnes différentes ne vont donc pas mener les tests de la même manière. De plus, il est intéressant de voir l'évolution de la sécurité système. Il faut donc pouvoir reproduire de manière plus ou moins précise ces évaluations de la sécurité afin d'observer une évolution positive ou négative. C'est, entre autres, pour ces raisons que l'utilisation de simples outils et techniques ne garantit pas une évaluation pertinente, complète et reproductible.

Certaines entités et groupes de personnes ont donc tenté de rassembler les techniques de test et outils connus afin de proposer des méthodologies d'évaluation de la sécurité. Ces méthodologies proposent des "marches à suivre" et des manières méthodiques et structurées afin d'évaluer la sécurité.

En plus des raisons précédemment citées, le développement et l'utilisation de méthodologies de test de la sécurité sont motivés par plusieurs autres raisons [19] :

- de nombreuses vulnérabilités ne sont pas détectables via des outils de tests automatisés. L'intervention et la réflexion humaine sont souvent nécessaires afin de compromettre la sécurité d'un système ;
- une phase de test de la sécurité est indispensable pour mener à bien une analyse de risques de la sécurité. En effet, effectuer des attaques réelles et concrètes permet de se mettre à

la place de l'attaquant et permet ainsi d'obtenir des données précises pour l'analyse de risques ;

- certaines sociétés utilisent les tests de pénétration et les tests de sécurité dans l'optique de former au mieux leurs employés afin que ceux-ci détectent et réagissent efficacement contre les cyber-attaques.

3.1 Frameworks et méthodologies de test

Plusieurs méthodologies d'évaluation de la sécurité informatique ont été proposées au fil des années. Nous avons choisi de présenter ci-dessous les plus importantes et les plus souvent citées.

3.1.1 OSSTMM

L'*Open Source Security Testing Methodology Manual* (OSSTMM) est une méthodologie visant à tester la sécurité opérationnelle des lieux physiques, des interactions humaines et de toutes les formes de communication telles que les communications sans fil, câblées, analogiques et numériques. La méthodologie de test (cfr figure 3.1) comprend 4 modules principaux. Les modules totalisent ensemble 17 étapes. Chaque étape est divisée en une ou plusieurs tâches. Les 4 modules sont décrits ci-dessous :

- phase d'induction : la personne qui réalise le test de sécurité cherche à comprendre et à définir les exigences du test, les contraintes qui seront rencontrées lors du test et le *scope* du test (étapes de couleur jaune sur la figure 3.1) ;
- phase d'interaction : on cherche à comprendre les interactions entre les éléments testés du système et les actifs informationnels du système (étapes de couleur orange sur la figure 3.1) ;
- phase d'enquête : on analyse la gestion et le stockage des données et des informations du système. On analyse si cela a un impact sur les actifs informationnels (étapes de couleur bleu clair sur la figure 3.1) ;
- phase d'intervention : on vérifie que les ressources nécessaires au bon fonctionnement du système ne sont pas critiques. Le système ne doit pas être mis à mal en cas de surcharge, de privation ou de changement de ressources (étapes de couleur rouge sur la figure 3.1). La dernière étape de la méthodologie consiste à enregistrer tous les résultats des tests effectués. Ceci afin de laisser une trace des résultats intéressants (étape de couleur bleu foncé sur la figure 3.1) ;

Après avoir réalisé l'évaluation de la sécurité, un rapport détaillé est rédigé et un score est calculé. Ce score est calculé, entre autres, sur base des actifs informationnels menacés.

3.1.2 ISSAF

L'*Information Systems Security Assessment Framework* (ISSAF) est une méthodologie de test divisée en trois phases principales :

- planning et préparation : dans cette phase, un accord d'évaluation de la sécurité est signé entre les deux parties (la partie qui effectue le test et la partie qui voit son système testé). Les données utiles aux différents tests sont échangées lors de cette phase ;

- *assessment* : c'est dans cette phase qu'a lieu le test de pénétration. Ce processus de pen-testing, qui peut être itératif, comporte 9 étapes majeures qui s'effectuent dans un ordre défini (cfr figure 3.2) ;
- rapport & destruction des artefacts : dans cette étape, on fournit un rapport contenant les résultats des différents tests de pénétration. Toutes les informations recueillies et générées par les tests doivent être détruites.

On retrouve, en plus de la méthodologie principale, une série de méthodologies plus courtes permettant de tester certaines parties de l'infrastructure (un firewall, un serveur windows, un switch,un VPN, un IDS, etc.).

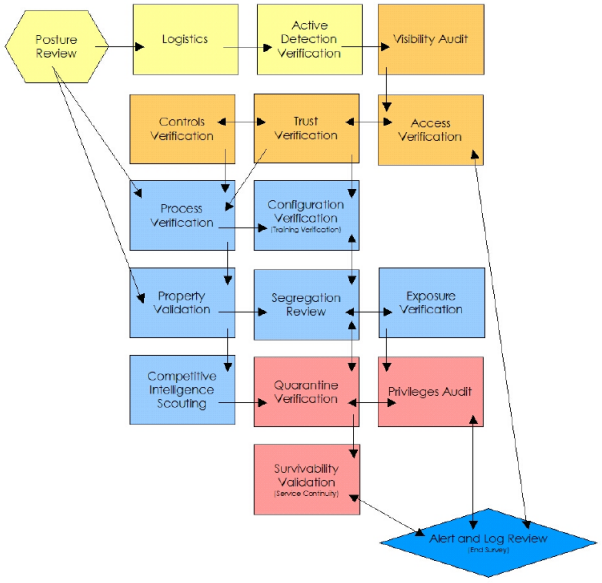


FIGURE 3.1 – Méthodologie OSSTMM [44]

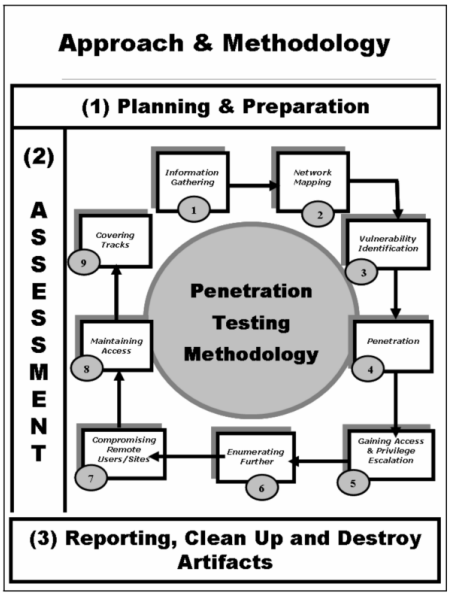


FIGURE 3.2 – Méthodologie ISSAF [62]

3.1.3 NIST SP800-115

On ne parle pas vraiment de méthodologie ici, mais plutôt d'un guide comprenant des techniques clés dans l'évaluation de la sécurité. Premièrement, ce guide explique que toute bonne méthodologie doit comprendre au minimum les trois phases suivantes :

- *planning* : cette étape permet de rassembler toutes les informations nécessaires pour la phase suivante (les actifs qui vont être évalués, les potentielles menaces qui vont viser les actifs et les contrôles de sécurité qui vont limiter les menaces) ;
- *execution* : le but principal de cette phase est d'identifier les vulnérabilités du réseau, du process et du système ;
- *post-execution* : cette phase a pour objectif l'analyse, la détection des "root causes" et la mitigation des vulnérabilités détectées. C'est également la phase durant laquelle on rédige le rapport final.

Deuxièmement, le guide décrit certaines méthodes permettant de tester certains aspects de la sécurité. On explique par exemple comment scanner un réseau sans fil, comment scanner un réseau bluetooth, etc. Ces techniques et méthodes sont réparties en 3 chapitres différents :

- 1. Les techniques "d'examen" qui analysent les actifs informationnels et les politiques de sécurité en place ;
- 2. Les techniques d'identification et d'analyse des cibles qui décrivent comment identifier les cibles potentielles ;
- 3. Les techniques de validation de la vulnérabilité des cibles qui expliquent comment tester les vulnérabilités découvertes.

Pour chacune des trois étapes, on retrouve plusieurs tests. Le guide propose une série d'outils.

3.1.4 NESCOR

Cette méthodologie a été conçue pour aider les équipes IT qui gèrent des installations industrielles/électriques à évaluer leur sécurité informatique. Elle comprend 9 étapes distinctes représentées sur la figure 3.3. On distingue 4 couleurs sur le schéma, elles correspondent au niveau de compétence que doit posséder le testeur ou l'attaquant afin de, respectivement, pouvoir mener à bien le test ou pouvoir compromettre la sécurité de la cible (vert étant le niveau le plus faible et rouge étant le niveau le plus élevé). Chacune des 9 étapes est divisée en sous-étapes. Chaque sous-étape est constituée de tâches. Les étapes, sous-étapes et tâches sont toutes étiquetées d'une couleur. Bien que la méthodologie ne propose pas beaucoup d'explications et de détails sur les étapes et sous-étapes, elle fournit cependant des descriptions précises sur la façon dont une tâche doit être exécutée et les objectifs généraux attendus pour chaque tâche.

Afin de mieux comprendre comment la méthodologie fonctionne, voici un exemple de test repris dans la méthodologie :
Si nous prenons l'étape "server OS penetration" de la figure 3.3, elle est en fait composée de 3 sous-étapes. "Information gathering", qui est une de ces 3 sous étapes, est divisée en 5 tâches. La tâche "port scanning", qui est une de ces 5 tâches, consiste à *utiliser des outils qui envoient des demandes à d'éventuels services de la couche d'application (comme le scanning des ports TCP et UDP pour découvrir des services comme HTTP et SSH)*. L'objectif de la tâche est d' *identifier tous les services d'écoute et les éventuelles règles du firewall.*

3.1.5 OWASP WSTG

Le framework WSTG (Web Security Testing Guide) est conçu pour étudier la sécurité des applications web et des services web. Il a été rédigé et testé par de nombreux experts en sécurité informatique faisant partie de l'organisation OWASP (Open Web Application Security Project). Bien que ce framework permet d'évaluer la sécurité d'une application, il propose également une méthodologie globale pour concevoir des applications sécurisées. L'OWASP explique, à juste titre, que la sécurité d'une application commence à sa création, et non pas à sa mise en production. La figure 3.4 reprend leur méthodologie. L'étape de déploiement reprend la phase d'évaluation de la sécurité. Pour cette étape de *pen testing*, on retrouve trois guides intéressants :

- OWASP Web Application Security Testing : cette méthodologie permet d'évaluer la sécurité d'un environnement web.
(https://github.com/OWASP/wstg/tree/master/document/4-Web_Application_Security_Testing)

- OWASP Mobile Security Testing Guide : cette méthodologie permet d'évaluer la sécurité d'une application mobile.
  (https://owasp.org/www-project-mobile-security-testing-guide/)
- OWASP Firmware Security Testing Methodology : cette méthodologie permet d'évaluer la sécurité d'un firmware.
  (https://github.com/scriptingxss/owasp-fstm)

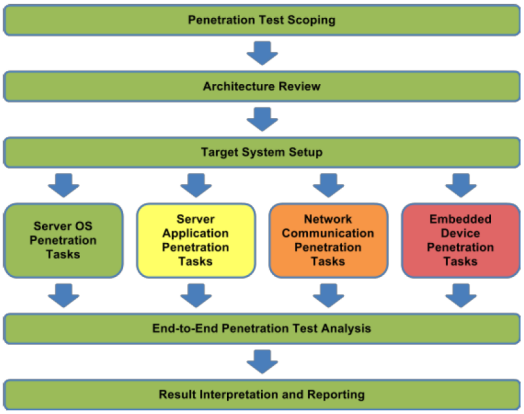


FIGURE 3.3 – Méthodologie NESCOR [74]

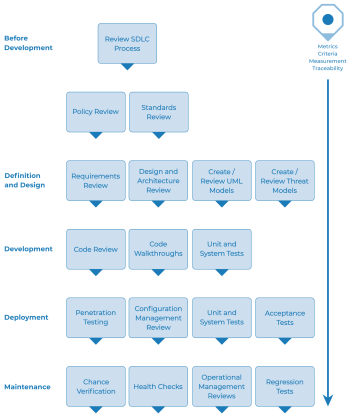


FIGURE 3.4 – Méthodologie OWASP [24]

3.2 Comparaison des frameworks et méthodologies

Après avoir expliqué et détaillé les différentes méthodologies, il nous a semblé intéressant de les comparer. Nous avons vu que certaines étaient plus générales, d'autres plus techniques, quelques unes contenaient des exemples pratiques, etc.

La méthodologie OSSTMM est générale et complète, elle peut s'appliquer à des systèmes divers et variés. Elle propose un très large éventail de tests, allant de, par exemple, la vérification des formations suivies par les utilisateurs, à la vérification du chiffrement des communications. La méthodologie énumère toutes les choses qu'il faut vérifier, elle décrit, pour chaque module, ce qui doit être testé, mais on ne dit pas comment le tester ni avec quel(s) outil(s) le tester. Elle n'est pas très orientée technique.

La méthodologie ISSAF est assez complète et générale. Elle permet de tester un bon nombre d'équipements et systèmes différents. Elle reprend les tests qu'il faut réaliser et elle propose parfois certains outils ou certaines méthodes précises pour les réaliser. Elle est cependant assez difficile à suivre et à appliquer. Même si des exemples sont bel et bien proposés, on ne voit pas clairement comment est appliquée la méthodologie. Elle est orientée technique.

Le guide NIST SP800-115 propose une série de tests clés pour effectuer une évaluation de la sécurité. Il n'est pas complet ou général mais il propose des tests intéressants et non couverts par les autres méthodologies (un scanning du réseau bluetooth par exemple). Il est orienté technique.

La méthodologie NESCOR est seulement dédiée aux systèmes industriels, on ne peut donc pas dire qu'elle est générale ou complète. Elle n'est pas très claire et détaillée dans la manière

générale d'appliquer la méthodologie. Elle est cependant assez précise et détaillée pour certaines tâches. Elle propose également certains outils. Elle est très orientée technique.

La méthodologie OWASP n'est pas très générale car elle concerne principalement les services et applications web. Elle est cependant la seule à proposer une méthodologie globale en passant par toutes les étapes de vie d'un produit (conception, déploiement, maintenance, ...). Les étapes de *pen testing* sont très précises, techniques et complètes. Elle propose de nombreux outils pour évaluer la sécurité et explique comment s'en servir.

Méthodologie	Métrique résultat	Utilisation générale	Orienté technique	Métrique complexité	Propose des outils	Exemples pratiques
OSSTMM	●	● ● ● ●	○ ○ ○ ○	○	○ ○ ○ ○	○
ISSAF	○	● ● ● ○	● ● ● ○	○	● ● ● ○	●
NIST SP800-115	○	● ● ○ ○	● ● ○ ○	○	● ● ● ○	○
NESCOR	○	● ○ ○ ○	● ● ● ○	●	● ○ ○ ○	○
OWASP	○	● ● ○ ○	● ● ● ●	○	● ● ● ●	●

TABLE 3.1 – Comparaison des méthodologies

La table 3.1 fait ressortir quelques informations intéressantes. Seul l'OSSTMM propose un score pour la partie "résultats de l'évaluation", et seul NESCOR propose de catégoriser les tests de pénétration en fonction du niveau de difficulté technique. Autre constat, la plupart des méthodologies se veulent générales et "orientées techniques". Elles proposent souvent des outils et parfois des exemples pratiques.

Deuxième partie

Recherches et développement

Chapitre 4

Revue de la littérature

4.1 Recensement des vulnérabilités IoT

Les attaques qui mettent à mal la sécurité d'un système IoT exploitent des vulnérabilités. Il est donc important de lister les différentes vulnérabilités connues, afin de maîtriser la sécurité d'un système IoT. Williams *et al.* proposent la définition suivante : *"Une vulnérabilité est définie comme une faille dans un système, une application ou un service qui permet à un attaquant de contourner les contrôles de sécurité et de manipuler les systèmes d'une manière non désirée par le développeur"* [87].

Nous confrontons ci-dessous deux travaux dans lesquels ces vulnérabilités sont mises en avant. Le premier article provient de l'OWASP (Open Web Application Security Project) [64], il fournit un classement (un top 10) des vulnérabilités les plus souvent rencontrées dans les systèmes IoT. Dans le second article, Neshenko *et al* [60] proposent 9 classes de vulnérabilités.

4.1.1 Top 10 selon l'OWASP

L'OWASP a publié un article en 2018 qui présente les 10 priorités auxquelles doivent faire face les entreprises et les consommateurs lors de la conception ou l'utilisation d'équipements IoT [64]. Nous estimons que cette étude est très pertinente car le processus qui a permis d'obtenir les résultats a été divisé en 6 phases :

1. Formation de l'équipe de recherche : sélection des personnes qui vont étudier le problème ;
2. Analyse du précédent top10 de 2014 de l'OWASP : analyse des changements qui ont eu lieu depuis 2014 (au niveau industrie), étude de la mise à jour du top 10 ;
3. Collecte des informations : recherche de vulnérabilités publiques et privées. Les vulnérabilités qui causent le plus de dégâts sont étudiées en priorité.
4. Analyse comparative avec les autres études du domaine : comparaison du top 10 avec d'autres études afin de s'assurer que rien n'a été oublié ;
5. Soumission à la communauté : soumission de l'étude à la communauté de l'OWASP. Les feedback de la communauté, les résultats de l'étude comparative avec les autres projets IoT et les données collectées ont été regroupés afin de finaliser le top10 de 2018.
6. Parution du top 10 : publication du top10.

Le top 10 des vulnérabilités est présenté ci-dessous, de la priorité la plus haute à la priorité la plus basse :

1. **Les mots de passe :** qu'ils soient composés de peu de caractères, facilement trouvables, hardcodés ou publiquement accessibles, ils constituent la plus grande source de menace ;
2. **Les services réseaux :** ces services (qui sont présents sur les équipements IoT), en particulier ceux exposés à internet, mettent en péril la confidentialité, l'authenticité ou l'intégrité et la disponibilité des informations. Ils peuvent également permettre des accès à distance non autorisés ;
3. **Interfaces :** les interfaces (webs, backend API, cloud, mobiles) présentes dans l'écosystème (on parle ici des interfaces autres que celles qui permettent de se connecter sur les équipements) non sécurisées peuvent compromettre la sécurité des équipements. Les problèmes les plus courants sont un manque d'authentification/autorisation, un manque de chiffrement et un manque de filtrage des entrées/sorties ;
4. **Mécanismes de mises à jour :** les mises à jour des équipements IoT ne sont pas sécurisées. Il n'y a pas de validation de firmware, il y a un manque de chiffrement lors des envois des mises à jour, il n'y a pas de mécanismes d'anti-rollback, il y a un manque de notifications de changement de la sécurité après une mise à jour ;
5. **Pièces et composants :** certaines librairies ou composants utilisés sont obsolètes ou peu sécurisés ;
6. **Protection des données privées :** les données des utilisateurs qui sont stockées sur les IoT ou dans l'écosystème sont stockées de façon non sécurisées ou sans la permission de l'utilisateur ;
7. **Transferts et stockage de données :** qu'elles soient au repos, en transit ou en train d'être calculées, les données ne sont pas cryptées et elles sont accessibles par des tiers car les contrôles d'accès aux données ne sont pas maîtrisés ;
8. **Manque de gestion des équipements :** gestion des mises à jour, gestion des actifs, gestion du decommissioning, gestion du monitoring ;
9. **Paramètres de configuration non sécurisés :** les équipements sont parfois fournis avec une configuration de base qui ne permet pas une sécurité optimale. En outre, les équipements n'offrent en général pas la possibilité de restreindre les droits d'édition des paramètres de configuration ;
10. **Accès physique aux équipements :** des informations sensibles peuvent être récupérées et peuvent permettre de prendre le contrôle de l'équipement localement ou à distance.

4.1.2 9 classes de Neshenko *et al*

Neshenko *et al* [60] ont parcouru l'ensemble des articles présents dans la littérature, traitant des vulnérabilités IoT et ont établi une classification. Leur travail nous semble très pertinent et nous présentons leurs résultats ci-dessous :

1. **Sécurité physique des équipements :** *"La majorité des dispositifs IoT fonctionnent de manière autonome dans des environnements non surveillés"* [54]. Les attaquants peuvent donc, moyennant peu d'effort, accéder à ces équipements et récupérer des informations

cryptographiques, insérer des équipements malveillants, récupérer des données ou corrompre certaines données.

2. **Source d'énergie des équipements :** les équipements IoT sont alimentés par batterie et ils n'ont pas la capacité de renouveler leur source d'alimentation par eux-mêmes. Un attaquant peut "vider" les batteries des équipements IoT (en les sollicitant plus que nécessaire) et donc rendre le système indisponible.
3. **Authentification inadéquate :** les contraintes (mentionnées à la section 2.4) empêchent la mise en place de mécanismes efficaces d'authentification. Un attaquant peut donc facilement insérer des noeuds malveillants dans le réseau.
4. **Chiffrement :** le chiffrement permet, dans les systèmes informatiques standards, de s'assurer que les données transitent de manière sécurisée. Les algorithmes de chiffrement sont coûteux en ressources et ils sont donc mal adaptés pour les systèmes IoT. Les attaquants peuvent donc facilement contourner les techniques de chiffrement utilisées dans les systèmes IoT.
5. **Ouverture de ports :** les ports réseaux des équipements IoT sont trop souvents ouverts et permettent donc aux attaquants d'accéder à certaines vulnérabilités liées aux services qui tournent sur les équipements.
6. **Contrôle d'accès insuffisant :** la pluplart des IoT peuvent être accessibles via des applications dans le cloud. Ces applications ne forcent pas les utilisateurs à créer des mots de passe complexes [33]. De plus, de nombreux équipements IoT ne demandent pas de modifier les mots de passe par défaut. Les attaquants ont donc la possibilité de pénétrer dans le système. Une fois dans le système, ils possèdent en général des accès élévés car les utilisateurs des systèmes IoT sont souvent détenteurs de droits élevés.
7. **Gestion des patchs :** de nombreuses études rapportent que les fabricants d'IoT ne mettent pas en place des mises à jour de sécurité (qu'elles soient automatiques ou non). Pire encore, si ces mécanismes existent, ils ne sont quand même pas jugés fiables, car ils sont sujets à des problèmes d'intégrité. Un attaquant peut modifier en sa faveur les patchs de sécurité.
8. **Pratique de programmation :** de nombreux chercheurs ont signalé que les équipements IoT comprenaient plusieurs problèmes de programmation au niveau firmware tels que des "backdoors", un manque d'utilisation de "Secure Socket Layer (SSL)", etc.
9. **Mécanismes d'audit :** la plupart des IoT ne possèdent pas d'audit trail ou de mécanismes de "logs", ce qui permet de dissimuler les activités et actions malveillantes.

4.1.3 Recoupement des vulnérabilités

Nous remarquons que, même si les deux articles ne présentent pas les informations de la même manière, certaines vulnérabilités figurent dans les deux classements. Nous avons recoupé les deux articles et nous présentons le résultat à la figure 4.1.

Sur base de la figure 4.1 nous avons établi la table 4.1 où les principaux types de vulnérabilités sont classés en fonction des couches systèmes auxquelles ils correspondent. Nous avons choisi de faire apparaître plusieurs fois les vulnérabilités si elles concernent plusieurs couches du système, comme par exemple "1. Mots de passe et contrôles d'accès". En effet, cette vulnérabilité se retrouve à la couche *Application* car on peut obtenir des mots de passe en utilisant des

techniques comme l'attaque *brute force* et elle se retrouve également à la couche *Perception* car des mots de passe peuvent être hardcodés dans le firmware de l'IoT. Il en va de même pour la vulnérabilité "6. manque de gestion du système", elle peut concerner la couche *Perception* avec par exemple une mauvaise gestion des équipements de terrain (absence de decommissioning des équipements qui ne sont plus utilisés) et elle peut concerner la couche *Application* avec par exemple les mises à jour des équipements qui doivent être réalisées par un utilisateur .

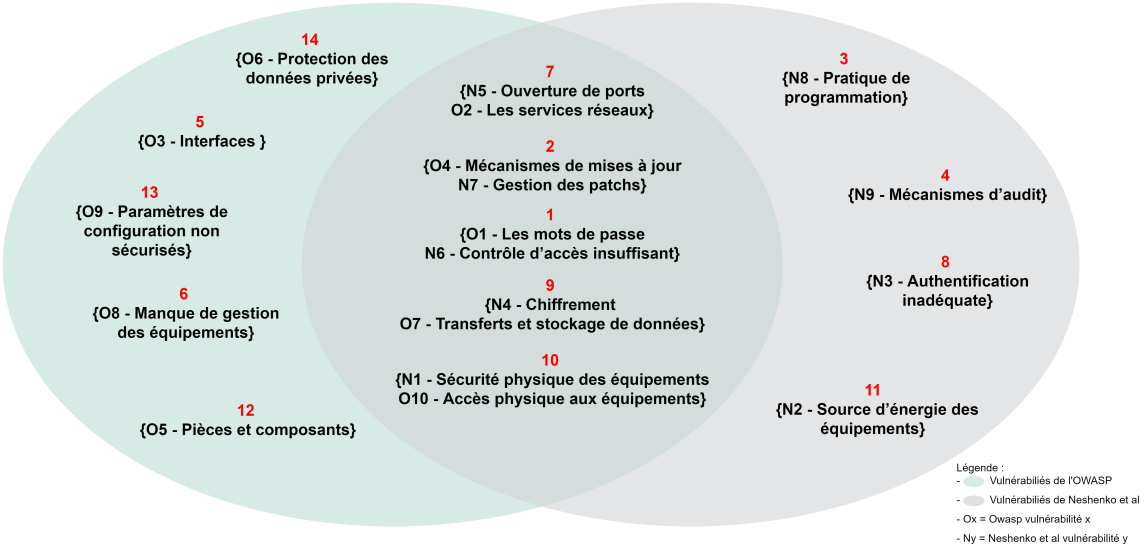


FIGURE 4.1 – Recoupement des vulnérabilités de 4.1.1 et 4.1.2

Couche système	Vulnérabilité
3. Application	1. Mots de passe et contrôles d'accès 2. Gestion des patchs 3. Pratique de programmation 4. Mécanismes d'audit trail et logging 5. Interface non sécurisée 6. Manque de gestion du système 12. Pièces et composants 13. Paramètres de configuration non sécurisés 14. Protection données personnelles
2. Network	7. Ports réseaux et services réseaux 8. Authentification inadéquate 9. Chiffrement
1. Perception	1. Mots de passe et contrôles d'accès 6. Manque de gestion des équipements 10. Sécurité physique des équipements 11. Source d'énergie des équipements

TABLE 4.1 – Vulnérabilités IoT par couche système

4.2 Attaques ciblant les IoT

Cette section inventorie les types d'attaques qui ciblent les équipements IoT. Nous avons classé les attaques en fonction des couches système auxquelles elles correspondent. Cette classification nous semble pertinente car, comme disent Andrea *et al.*, *"Comme les équipements IoT utilisent des technologies réseau différentes (RFID, sans fil, internet, etc.), il est nécessaire de bien catégoriser les attaques de manière à englober tous les différents types de menaces"* [4]. La figure 4.2 reprend les différentes couches de l'architecture ainsi que les attaques auxquelles elles sont sujettes.

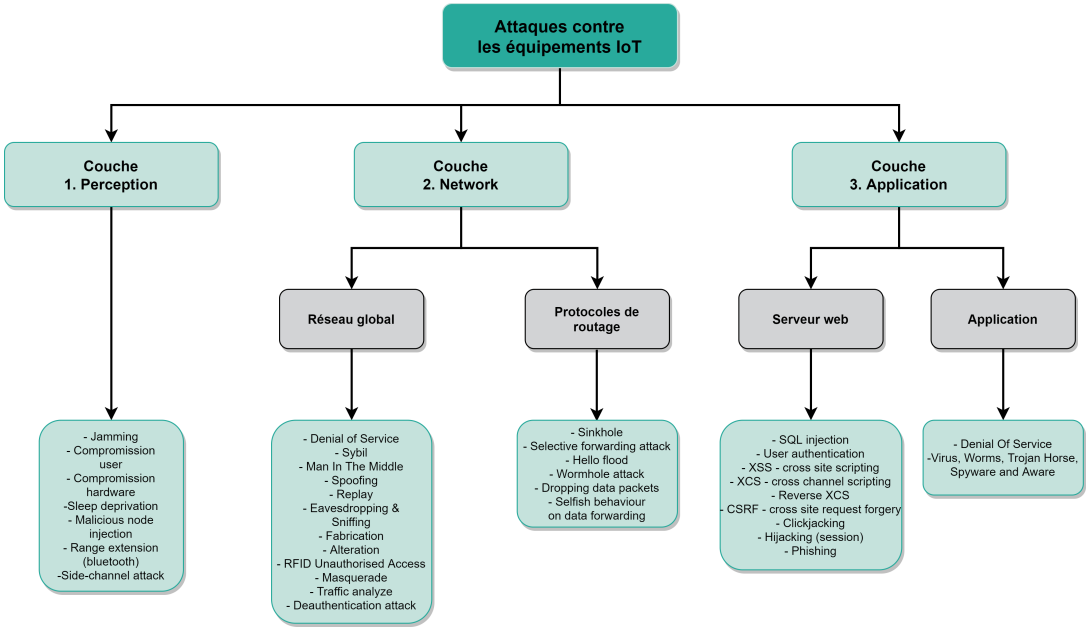


FIGURE 4.2 – Attaques ciblant les équipements IoT

4.2.1 Attaques sur la couche 1. Perception

Ces types d'attaques se concentrent sur les composants matériels du système IoT et l'attaquant doit être physiquement proche ou dans le système IoT pour que les attaques fonctionnent. De plus, les attaques qui portent atteinte à la durée de vie ou aux fonctionnalités du matériel sont également incluses dans cette catégorie.

- **"Jamming attack" :** ce type d'attaque, qui est dédié aux réseaux sans fil, consiste à rendre le réseau occupé en émettant un signal d'interférence. On distingue plusieurs types de jamming : constant jamming, deceptive jamming, reactive jamming et random jamming [88]
- **Compromission hardware :** un adversaire extrait des informations d'identification intégrées telles que des données, des clés ou des codes de programme stockés dans un dispositif IoT en altérant le matériel. Ce type d'attaque nécessite généralement un accès physique aux dispositifs [11] [2].
- **Compromission user / Social Engineering :** un adversaire piège les utilisateurs afin qu'ils dévoilent leurs informations de sécurité telles que les clés ou des mots de passe

[8]. Cette attaque est placée dans cette catégorie car l'attaquant a besoin d'interagir physiquement avec les utilisateurs du réseau IoT [4].

- **Sleep Deprivation Attack :** la plupart des équipements IoT sont alimentés par des batteries remplaçables et sont programmés pour suivre des routines d'économie d'énergie afin de prolonger la vie de la batterie. Ce type d'attaque consite à forcer l'équipement à utiliser le maximum de ressources afin de l'entraîner, de manière prématurée, à l'arrêt. [4] [15] [46].
- **Malicious Node Injection :** l'attaquant peut introduire un noeud malicieux au sein du réseau. Il contrôle alors les données qui transitent via le noeud. Cette attaque est également appelée "Man in The Middle Attack" [4].
- **Range extension :** cette attaque concerne les réseaux bluetooth. Elle consiste à étendre au maximum la portée du bluetooth afin de pouvoir effectuer des attaques à de plus grandes distances [52].
- **Side-channel attack** : cette attaque consiste à observer et à étudier des informations émises par les systèmes de cryptage. On ne se base donc pas sur le cryptage lui-même (entrées et sorties), mais plutôt sur les composants hardware et software. On peut par exemple étudier le champ électromagnétique émis par les composants hardware, les temps d'exécution, les consommations énergétiques, etc. [25] .

4.2.2 Attaques sur la couche 2. Network

Ces attaques sont centrées sur le réseau du système IoT et l'attaquant n'a pas nécessairement besoin d'être proche du réseau pour que l'attaque fonctionne.

Attaques sur le réseau global

- **(Distributed) Denial of service** : dans les réseaux, une attaque par déni de service (attaque DoS) est réalisée en inondant la victime de demandes, ce qui génère un important trafic réseau [12] [46] [72] [91].
- **"Sybil attack"** : cela consite à usurper l'identité de noeuds présents dans le réseau ou de créer des noeuds malicieux au sein du réseau. On distingue trois formes différentes de "sybil attack". La première forme d'attaque consiste à créer des équipements fictifs. Ces équipements votent alors ensemble contre des équipements réels et honnêtes du réseau. Les noeuds fictifs peuvent ensuite refuser de recevoir ou de transmettre des données ou des paquets. Les deuxièmes et troisièmes formes sont similaires et consistent à intégrer des noeuds malicieux au sein du réseau afin qu'ils disséminent des spams, des malwares, etc. [89]. Sastry *et al.* [72] expliquent également que *"dans un réseau de capteurs sans fil (WSN), comme l'attaquant a plusieurs identités, le nœud victime peut transmettre des informations via le nœud compromis, ce qui entraîne une plus grande distance de routage"*.
- **MITM (Man In The Middle)** : ce type d'attaque consiste à intercepter une communication entre deux équipements A et B. MITM se fait passer pour l'équipement B vis à vis de l'équipement A et inversement. Ce type d'attaque conduit en général à des attaques de type *eavesdropping* ou des attaques de type "altération des données". Cette attaque apparaît dans cette catégorie car il n'est pas nécessaire d'être présent physiquement sur le réseau. L'attaque repose seulement sur les protocoles de communication [4][23] [42] [84].

- **IP/RFID Spoofing** : cette attaque consiste à envoyer des paquets IP en usurpant l'adresse IP d'un autre équipement [4] ,[16].
- **Replay attack** : c'est une forme d'attaque réseau. Des données valides sont répétées ou retardées de manière malveillante. Cette attaque est menée soit par le noeud d'origine, soit par un attaquant qui aura au préalable intercepté l'échange[82].
- **Eavesdropping (ou sniffing)** : cela consiste à écouter, de manière non autorisée, les échanges qui ont lieu sur le réseau (à l'aide d'un MITM, en redirigeant le trafic, etc). De part leur nature (la grande utilisation des réseaux sans fil), les systèmes IoT sont souvent la cible de ce genre d'attaques [29][46] [92] [66] [23] [56]. Les attaquants utilisent des dispositifs et des applications de "sniffing" pour obtenir des informations sur le réseau et en extraire des données précieuses pour pourvoir perpétrer de nouvelles attaques [46] [84].
- **Fabrication** : ce type d'attaque consiste à mettre à mal l'authenticité des données. Un adversaire génère des données ou des activités supplémentaires qui n'existeraient pas normalement. Cette attaque crée une confusion parmi les parties impliquées dans la communication [93].
- **Alteration** : ce type d'attaque consiste à mettre à mal l'intégrité des données. Un adversaire obtient un accès non autorisé aux données et modifie ou falsifie les messages afin d'induire en erreur d'autres noeuds du réseau.
- **RFID Unauthorised Access** : dans les systèmes RFID, on ne retrouve pas de mécanisme d'authentification. Un attaquant peut lire, modifier ou supprimer des données dans les noeuds RFID [81].
- **Masquerade attack** : cette attaque consiste à usurper l'identité d'un noeud honnête présent sur le réseau (l'adresse MAC, par exemple). Le noeud qui usurpe l'identité peut alors lancer des attaques de l'intérieur, perturber le bon fonctionnement du réseau ou accéder à certains services. [53].
- **Traffic analyze** : *"les attaquants déduisent le modèle et la charge de communication, en analysant le nombre et la taille des paquets de données transmis"* [12] [65] [84].
- **Deauthentication attack** : cette attaque permet de déconnecter un équipement du réseau. L'attaquant usurpe l'adresse MAC de l'équipement et demande une déconnexion à *l'access point* [6].

Attaques sur les protocoles de routage

- **Sinkhole Attack** : ce type d'attaque consite à introduire un noeud malicieux au sein d'un réseau ou à corrompre un noeud honnête. Une fois le noeud malicieux présent dans le réseau, il envoie des mauvaises informations de routage aux autres noeuds, afin que le trafic soit redirigé par lui. Le noeud malicieux a ensuite la possibilité d'altérer les données [21]. Farooq *et al.* [46] expliquent que cette attaque peut également faire croire au système que certains paquets ont déjà été délivrés, alors que ce n'est pas le cas.
- **Selective Forwarding Attack** : cette attaque peut prendre deux comportements différents. Soit les noeuds malicieux dans le réseau sélectionnent les paquets de données qui pourront transiter ou non. Soit les noeuds malicieux ne laissent passer aucun paquet de données [48].

- **Hello flood attack** : dans certains protocoles de routage, les noeuds doivent envoyer un message de broadcast (contenant un message "Hello") afin de pouvoir s'intégrer à un réseau. L'attaque "hello flood" consiste à utiliser un équipement puissant afin d'envoyer un message de broadcast à tous les noeuds du réseau. Cet équipement malicieux fait croire aux autres noeuds qu'il a la possibilité de communiquer facilement avec la station de base (noeud vers lequel les messages doivent arriver). Les noeuds du réseau vont alors tenter de passer par cet équipement malicieux, mais les messages seront alors envoyés dans le vide faute de puissance d'émission. [77].
- **Wormhole attack** : cette attaque est une forme d'attaque réseau. Deux noeuds malicieux au sein d'un même réseau vont créer un tunnel fictif afin de faire croire aux autres noeuds du réseau que le tunnel est le chemin le plus court pour transporter les messages. Les autres noeuds vont alors référencer le tunnel comme étant la route la plus courte. Les noeuds malicieux vont alors envoyer les paquets de données vers des routes plus longues, ce qui causera des problèmes de délai et de latence [10].
- **Dropping data packets attack** : cela consiste à perdre volontairement des paquets de données. Certains équipements du réseau ont pour fonction de gérer le routage des messages. Si un équipement malicieux s'immisce dans les équipements responsables du routage, il peut alors supprimer des paquets au lieu de les acheminer [29]
- **Selfish behavior on data forwarding** : ce n'est pas une attaque à proprement parler, mais cela pose parfois un sérieux problème. En effet, pour certaines applications, les noeuds du réseau ne poursuivent pas un but commun et se comportent de manière "égoïste". Un noeud égoïste essaye de préserver ses ressources (batterie) et n'hésite pas à demander à des noeuds voisins d'envoyer des paquets de données à sa place ou refuse de recevoir certains paquets qui lui sont destinés afin de ne pas consommer des ressources. Ce comportement met à mal la qualité de service du système ainsi que la - Disponibilité. [29].

4.2.3 Attaques sur la couche 3. Application

Les attaques logicielles sont la principale source de vulnérabilité de la sécurité. Elles peuvent voler des informations, altérer des données, refuser l'accès aux services et même endommager les dispositifs IoT [4].

Attaques software - serveur web

- **SQL injection** : Dorai *et al.* expliquent : [...] *"en embarquant des instructions SQL dans les données d'entrée, un programme mal conçu peut être vulnérable à de telles attaques"* [32] [90].
- **User authentication** : des attaques de type "brut force" permettent de trouver les mots de passe trop simples. Certains botnets comme Mirai exploitent le fait que les mots de passe par défaut ne sont pas modifiés [83].
- **XSS - cross site scripting** : Ce type d'attaque consiste à injecter du contenu dans une interface web. Par exemple, on ajoute un script malicieux à un lien qui redirige vers une autre page.

- **XCS - cross channel scripting** : dans une attaque XCS, un canal autre que l'interface web, tel que SNMP ou FTP, est utilisé pour injecter un XSS persistant qui est activé lorsque l'utilisateur se connecte à l'interface web. Par exemple, plusieurs NAS sont sujets à ce type d'attaque. Il est possible d'uploader des fichiers avec un nom plus ou moins arbitraire via SMB. L'adversaire upload donc un fichier vide mais qui est intitulé avec un script malicieux (le nom du fichier est en fait le script). Quand on demande, via l'interface web, de lister les fichiers, le nom du fichier est exécuté comme un script malicieux. [17].
- **Reverse XCS** : un reverse XCS utilise l'interface web pour éventuellement attaquer un canal "non-web". Par exemple, on peut extraire des données protégées stockées sur un NAS par le biais d'un réseau P2P. [17].
- **CSRF - cross site request forgery** : cette attaque permet à un adversaire de compromettre un dispositif IoT en utilisant un site web externe comme tremplin pour infiltrer un intranet. Sur les dispositifs embarqués, elle peut également être utilisée comme un vecteur direct d'attaque car elle permet à l'attaquant de redémarrer des équipements réseau critiques tels que les commutateurs, les téléphones IP et les routeurs [37].
- **Clickjacking** : le "clickjacking" est une attaque qui consiste à utiliser plusieurs couches transparentes ou opaques pour inciter un utilisateur à cliquer sur un bouton ou un lien d'une autre page alors qu'il avait l'intention de cliquer sur la page de premier niveau [63]. Par exemple, le Clickjacking de base peut être utilisé pour redémarrer les appareils, effacer leur contenu et, dans le cas des routeurs, permettre l'accès au réseau. Le Clickjacking avancé permet à l'attaquant de voler la clé WPA du routeur ou le mot de passe du NAS [37].
- **Hijacking (session)** : cette attaque consiste à voler une session d'un utilisateur déjà connecté et authentifié à un *web server*. L'attaquant prend le contrôle de cette session, et continue la connexion avec le serveur tout en prétendant être l'utilisateur normal. Il y a trois types de session jijacking : active, passive et hybride [16].
- **Phishing** : cette attaque consiste à faire croire à la victime qu'elle utilise un site réel ou vrai alors qu'il s'agit en fait d'une copie malicieuse. Le design de la fausse application web ressemble fortement à la vraie application. Ces attaques visent principalement les adresses emails et les identités de haut niveau [79].

Attaques software - application

- **Denial of service** : un attaquant peut exécuter des attaques DoS ou DDoS sur le réseau IoT à travers la couche application, affectant tous les utilisateurs du réseau [55].
- **Virus, Worms, Trojan Horse, Spyware and Aware** : un adversaire peut infecter le système à l'aide d'un logiciel malveillant, ce qui peut avoir plusieurs conséquences : vol d'informations, altération de données ou même déni de service [41].

4.3 Scénarios d'attaque

Le tableau 4.2 reprend l'ensemble des scénarios d'attaque auxquels peuvent être confrontés les équipements IoT. Un scénario d'attaque met en relation une attaque (cfr section 4.2), la vulnérabilité qu'elle exploite (cfr section 4.1), l'exigence de sécurité qu'elle impacte (cfr section 2.5) et la couche système qui est concernée (cfr section 1).

Attaque	Couche IoT concernée	Exigence de sécurité impactée	Vulnérabilité exploitée
➤ **Jamming**	1. Perception	- Disponibilité	- 10. Sécurité physique des équipements - 12. Pièces et composants
➤ **Compromission hardware**	1. Perception	- Confidentialité - Intégrité	- 1. Mots de passe et contrôles d'accès - 6. Manque de gestion des équipements - 10. Sécurité physique des équipements - 13. Paramètres de configuration non sécurisés
➤ **Compromission user / Social Engineering**	1. Perception	- Confidentialité - Intégrité	- 5. Interface non sécurisée - 6. Manque de gestion des équipements - 10. Sécurité physique des équipements
➤ **Sleep deprivation**	1. Perception	- Disponibilité	- 11. Source d'énergie des équipements
➤ **Malicious Node Injection**	1. Perception	- Disponibilité - Confidentialité - Intégrité	- 6. Manque de gestion des équipements - 10. Sécurité physique des équipements
➤ **Range extension**	1. Perception	- Disponibilité - Confidentialité - Intégrité	- 6. Manque de gestion des équipements - 10. Sécurité physique des équipements
➤ **Side-channel**	1. Perception	- Confidentialité	- 10. Sécurité physique des équipements - 12. Pièces et composants
➤ **DoS / DDoS**	2. Network	- Disponibilité - Robustesse - Gestion d'erreurs	- 7. Ports réseaux et services réseaux
➤ **Sybil attack**	2. Network	- Authentification - Disponibilité - Robustesse - Gestion d'erreurs - Self organisation	- 7. Ports réseaux et services réseaux - 8. Authentification inadéquate - 9. Chiffrement
➤ **Man In The Middle**	2. Network	- Confidentialité - Intégrité - Disponibilité	4. Mécanismes d'audit trail et logging - 7. Ports réseaux et services réseaux - 8. Authentification inadéquate - 9. Chiffrement
➤ **Spoofing**	2. Network	- Authentification - Intégrité	- 8. Authentification inadéquate - 9. Chiffrement
➤ **Replay attack**	2. Network	- Freshness	- 9. Chiffrement - Synchronisation de l'heure
➤ **Eavesdropping / Sniffing**	2. Network	- Confidentialité	- 7. Ports réseaux et services réseaux - 9. Chiffrement
➤ **Fabrication**	2. Network	- Intégrité	- 4. Mécanismes d'audit trail et logging - 7. Ports réseaux et services réseaux - 9. Chiffrement
➤ **Alteration**	2. Network	- Intégrité	- 7. Ports réseaux et services réseaux - 8. Authentification inadéquate - 9. Chiffrement
➤ **Unauthorised Access**	2. Network	- Authentification	- 8. Authentification inadéquate - 9. Chiffrement
➤ **Masquerade**	2. Network	- Authentification - Confidentialité - Intégrité - Disponibilité	- 7. Ports réseaux et services réseaux - 8. Authentification inadéquate - 9. Chiffrement

Attaque	Couche IoT concernée	Exigence de sécurité impactée	Vulnérabilité exploitée
➤ **Traffic analyze**	2. Network	- Confidentialité	- 7. Ports réseaux et services réseaux - 9. Chiffrement
➤ **Deauthentication attack**	2. Network	- Disponibilité	- 7. Ports réseaux et services réseaux - 8. Authentification inadéquate
➤ **Sinkhole**	2. Network	- Disponibilité	- 4. Mécanismes d'audit trail et logging - 6. Manque de gestion des équipements - 7. Ports réseaux et services réseaux - 8. Authentification inadéquate - 9. Chiffrement
➤ **Selective Forwarding**	2. Network	- Disponibilité	- 4. Mécanismes d'audit trail et logging - 6. Manque de gestion des équipements - 7. Ports réseaux et services réseaux - 8. Authentification inadéquate - 9. Chiffrement
➤ **Hello flood**	2. Network	- Disponibilité	- 4. Mécanismes d'audit trail et logging - 6. Manque de gestion des équipements - 7. Ports réseaux et services réseaux - 8. Authentification inadéquate - 9. Chiffrement
➤ **Wormhole**	2. Network	- Disponibilité - Robustesse - Gestion d'erreurs	- 4. Mécanismes d'audit trail et logging - 6. Manque de gestion des équipements - 7. Ports réseaux et services réseaux - 8. Authentification inadéquate - 9. Chiffrement
➤ **Dropping Data Packets**	2. Network	- Disponibilité	- 4. Mécanismes d'audit trail et logging - 6. Manque de gestion des équipements - 7. Ports réseaux et services réseaux - 8. Authentification inadéquate - 9. Chiffrement
➤ **Selfish Behavior on Data Forwarding**	2. Network	- Disponibilité	- 4. Mécanismes d'audit trail et logging - 6. Manque de gestion des équipements - 7. Ports réseaux et services réseaux - 8. Authentification inadéquate - 9. Chiffrement
➤ **SQL Injection**	3. Application	- Confidentialité - Intégrité - Disponibilité	- 2. Gestion des patchs - 3. Pratique de programmation - 5. Interface non sécurisée
➤ **User authentication**	3. Application	- Confidentialité - Intégrité	- 1. Mots de passe et contrôles d'accès - 3. Pratique de programmation
➤ **XSS - cross site scripting**	3. Application	- Disponibilité - Confidentialité - Intégrité	- 3. Pratique de programmation - 5. Interface non sécurisée
➤ **XCS - cross channel scripting**	3. Application	- Disponibilité - Confidentialité - Intégrité	- 3. Pratique de programmation - 5. Interface non sécurisée - 7. Ports réseaux et services réseaux
➤ **Reverse XCS**	3. Application	- Disponibilité - Confidentialité - Intégrité	- 3. Pratique de programmation - 5. Interface non sécurisée - 7. Ports réseaux et services réseaux

Attaque	Couche IoT concernée	Exigence de sécurité impactée	Vulnérabilité exploitée
➤ **CSRF - cross site request forgery**	3. Application	- Disponibilité - Confidentialité - Intégrité	- 3. Pratique de programmation - 5. Interface non sécurisée - 7. Ports réseaux et services réseaux
➤ **Clickjacking**	3. Application	- Disponibilité - Confidentialité - Intégrité	- 3. Pratique de programmation - 5. Interface non sécurisée - 7. Ports réseaux et services réseaux
➤ **Hijacking (session)**	3. Application	- Disponibilité - Confidentialité - Intégrité	- 3. Pratique de programmation - 5. Interface non sécurisée - 7. Ports réseaux et services réseaux
➤ **Phishing**	3. Application	- Disponibilité - Confidentialité - Intégrité - Authentification	- 3. Pratique de programmation - 5. Interface non sécurisée - 7. Ports réseaux et services réseaux
➤ **DoS / DDoS**	3. Application	- Disponibilité	- 3. Pratique de programmation - 5. Interface non sécurisée - 7. Ports réseaux et services réseaux
➤ **Virus, Worms, Trojan Horse, Spyware and Aware**	2. Network	- Disponibilité - Confidentialité - Intégrité - Robustesse	- 2. Gestion des patchs - 6. Manque de gestion du système - 7. Ports réseaux et services réseaux

TABLE 4.2 – Scénarios d'attaques

Cette table 4.2 permet de récupérer facilement certaines informations. Un utilisateur pourrait, par exemple, avoir besoin de connaître toutes les attaques qui mettent à mal la disponibilité ou l'intégrité de son système. Il pourrait également connaître les attaques qui ciblent une couche spécifique du système, etc.

4.4 Bancs de test existant

Tester la sécurité des dispositifs IoT avant de les introduire sur le marché est une étape importante dans le développement des produits, et c'est un domaine dans lequel les bancs de test peuvent s'avérer extrêmement utiles [83]. Un banc de test est un environnement de test prédéfini dans lequel tous les déclencheurs, tests, attaques et dispositifs sont contrôlés [18]. Ils réalisent des évaluations complètes de la vulnérabilité des dispositifs en utilisant des outils de diagnostics et des outils de test de pénétration. Les bancs de test évaluent les vulnérabilités de l'appareil dans des conditions réelles. Une fois les tests effectués, on peut tirer des conclusions sur les faiblesses et les vulnérabilités de l'appareil. En général, les bancs de test se composent d'un ensemble d'outils logiciels et matériels qui peuvent simuler ou modifier certains paramètres environnementaux tels que la lumière, l'heure, la localisation GPS, etc.

Même si certains bancs de test existent depuis une dizaine d'année, nous avons choisi d'étudier ceux qui étaient postérieurs à 2015. En effet, les bancs de test antérieurs à 2015, même s'ils permettaient de mettre en avant certaines vulnérabilités, ne couvraient pas un large éventail de tests, n'étaient pas automatisés et étaient peu modulables. Parmi ces bancs de tests, on peut

citer : Kansei [7], MoteLab [85], CitySense [58], Senselab [28], FIT IoT-LAB [43], FIESTA-IOT [34], SmartCampus [59], IoTSAT [57], Indriya2 [5], Wisebed [20], Berhanu *et al.* [14], SmartSander [71]. Depuis 2015, les bancs de test ont bien évolué et ils proposent une série de nouvelles fonctionnalités intéressantes, comme le caractère automatique des tests de vulnérabilités, des rapports détaillés des vulnérabilités détectées, la disponibilité d'un "Management System" (MS) qui contrôle le banc de test, etc. Parmi les quelques bancs de test du genre qui existent, on peut citer : année 2016 - Tekeoglu *et al.* [78], année 2017 - Sachidananda *et al.* [68], année 2019 Siboni *et al.* [76] et - Hale *et al.* [39], année 2020 - Waraga *et al.* [83]. Ces bancs de test seront détaillés et analysés aux sections 4.4.1 et 4.4.2. Nous pouvons remarquer qu'on ne dénombre pas beaucoup de bancs de test IoT "récents". On en recense 4 différents (en effet celui de Siboni *et al.* [76] est une amélioration de celui de Sachidananda *et al.* [68]).

4.4.1 Liste des logiciels et outils utilisés

Nous présentons ci-dessous les outils et tests repris dans les différents bancs de test.

	Tekeoglu *et al.* [78]	Sachidananda *et al.* [68]	Hale *et al.* [39]	Siboni *et al.* [76]	Waraga *et al.* [83]
Aircrack	-	X	-	X	-
Binwalk	X	-	-	-	-
Cain & Abel	-	X	-	X	-
Dhcpdump	-	-	-	X	-
Dirb	-	-	-	-	X
Ebtables	X	-	-	-	-
Iptables	X	-	-	-	-
Kismet	X	-	-	-	-
Metasploit	-	-	-	X	X
Nessus	-	X	-	X	-
Nikto	-	-	-	X	X
Nmap	-	X	-	X	X
OpenVAS	X	X	-	X	-
OpenWrt	X	-	-	-	-
OSSEC	-	X	-	X	-
scapy PyPI	-	-	-	X	-
SL Scan	-	-	-	-	X
SQLmap	-	-	-	-	X
SSLStrip	-	-	-	-	X
Tenable	-	X	-	-	-
TLS proper	-	-	-	-	X
Tshark	-	-	-	-	X
Ubertooth	-	-	X	-	-
WAFW00F	-	-	-	-	X
Wireshark	X	X	X	X	-

TABLE 4.3 – Logiciels utilisés dans les bancs de test

	Tekeoglu *et al.* [78]	Sachidananda *et al.* [68]	Hale *et al.* [39]	Siboni *et al.* [76]	Waraga *et al.* [83]
Port scanning	-	X	-	X	X
Fingerprinting	-	X	-	X	
Vulnerability scans	-	X	-	X	X
Downgrading attack	-	-	-	-	X
Brute force (passwords)	X	-	-	X	X
Brute force (directories)	X	-	-	X	X
Brute force (ports service)	X	-	-	X	X
Testing SSL configuration	-	-	-	X	X
EavesDropping	-	-	X	-	-
DoS attack	-	-	X	-	-
Process enumeration	-	X	-	X	-
Firmware update encryption	X	-	-	-	-
Communication tampering	-	-	-	X	-
Spoofing/masquerade attack	-	-	-	X	-
Communication delay attack	-	-	-	X	-
List known vulnerabilities	-	-	-	X	-
Data collection	-	-	-	X	-
Data leakage	-	-	-	X	-
Cipher suits checks	X	-	-	-	-
Connection time server	-	-	-	-	X
SQl injection	-	-	-	-	X
Firewall	-	-	-	-	X
Firmware check	-	-	-	-	X
Automated tests	-	X	-	X	X
Bluetooth options	X	X	X	X	X
Wifi options	X	X	X	X	X
Management system	-	X	-	X	X
Open source manag. system	-	-	-	-	X

TABLE 4.4 – Liste des tests réalisés par les bancs de test

Les bancs de test utilisent des outils open source afin de détecter et de mettre en évidence certaines vulnérabilités. Ils sont repris dans la table 4.3. Les scripts et outils créés spécialement pour le banc de test ne sont pas repris dans cette table. Les 5 bancs de test utilisent tous, sans exception, une machine sur laquelle tourne l'operating system Kali Linux [1]. Une partie des logiciels listés dans la table 4.3 provient d'ailleurs de cet operating system.

4.4.2 Liste des attaques et expérimentations couvertes

La table 4.4 décrit les fonctionnalités propres à chaque banc de test. La première partie de la table liste les tests de vulnérabilités qui sont effectués. La deuxième partie de la table reprend certaines fonctionnalités générales du banc de test. Nous pouvons constater que plus le banc de test est récent, plus il couvre un nombre important de tests et plus il comprend de fonctionnalités. On remarque également que certains tests ne sont effectués que par un seul banc de test et que la plupart des tests effectués diffèrent entre les bancs de test. Nous pouvons donc dire que les bancs de test sont relativement complémentaires. On pourrait envisager de tous les utiliser afin d'obtenir une meilleure évaluation de la sécurité. Nous n'avons pas testé personnellement les bancs de test. La table 4.4 a été complétée en lisant les articles dédiés aux bancs de test.

4.4.3 Classement en fonction des vulnérabilités

La table 4.5 catégorise les tests effectués par les bancs de test en fonction des vulnérabilités listées à la section 4.1.3. Cette table permet de voir que certaines vulnérabilités ne sont pas testées dans les bancs de test, elles sont surlignées en couleur verte.

	Test effectué	Tekeoglu *et al.* [78]	Sachidananda *et al.* [68]	Hale *et al.* [39]	Siboni *et al.* [76]	Waraga *et al.* [83]
1. Mots de passe et contrôles d'accès	Brute force (directories)	X	-	-	X	X
	Brute force (passwords)	X	-	-	X	X
	Brute force (ports service)	X	-	-	X	X
	Disassemble mobile application	-	-	-	-	X
2 Gestion des patchs	Vérifier firmaware à jour	-	-	-	-	X
3. Pratique de programmation		-	-	-	-	-
4. Audit trail et logging	Check async. connection NTP	-	-	-	-	X
5. Interface non sécurisée	Scan vulnérabilités	-	-	-	-	X
	SQl & XSS injection	-	-	-	-	X
	Check Firewall web	-	-	-	-	X
6. Gestion système & équipements		-	-	-	-	-
7. Ports réseaux et services réseaux	Port scanning	-	X	-	X	X
	Vulnerability scanner	-	X	-	X	-
	Process enumeration	-	X	-	X	-
	DoS attack	-	-	X	-	-
8. Authentification inadéquate	Replay attack	-	-	-	X	-
	Spoofing attack	-	-	-	X	-
	Bypass basic HTTP authentication	-	-	-	-	X
9. Chiffrement	Analyse communication	-	-	-	-	X
	Downgrade Attack	-	-	-	-	X
	Data leakage	-	-	-	X	-
	Eavesdropping	-	-	X	-	-
	Cipher suits checks	X	-	-	-	-
	Secure Sockets Layer certificate	-	-	-	-	X
	Updates sent with encryption	X	-	-	-	-
10. Sécurité physique équipements	Prise UART	-	-	-	-	X
11. Source d'énergie des équipements		-	-	-	-	-
12. Pièces et composants		-	-	-	-	-
13. Paramètres de configuration		-	-	-	-	-

TABLE 4.5 – Recoupement des tests réalisés par les bancs de test et des vulnérabilités

Chapitre 5

Elaboration de la méthodologie

Nous avons vu, à la section 3, qu'il existe des méthodologies d'évaluation de la sécurité informatique. Nous avons également vu, à la section 4.4, qu'il existe des bancs de test permettant de tester la sécurité d'un équipement IoT. Il n'existe cependant pas, à notre connaissance, de méthodologie complète et générique permettant d'évaluer la sécurité d'un équipement IoT. L'objectif de ce travail est donc de proposer une méthodologie qui permettrait de pallier ce manquement.

Les méthodologies d'évaluation de la sécurité informatique ne s'appliquent pas facilement aux systèmes IoT et ce, pour plusieurs raisons. Premièrement, les problèmes de sécurité ne sont pas les mêmes, certains problèmes sont spécifiques aux équipements IoT (par exemple *sleep deprivation attack*). Deuxièmement, certaines méthodes de sécurité conventionnelles ne sont pas utilisables dans les systèmes IoT à cause des limitations software, hardware et réseau (cfr section 2.4). Troisièmement, elles sont parfois trop généralistes et complètes. Elles visent à tester chaque équipement du système, un firewall, un routeur, etc. Un équipement ou un système IoT ne possède pas toujours cet ensemble de composants.

Les bancs de test sont certes utiles pour évaluer la sécurité d'un équipement IoT, mais ils restent néanmoins problématiques pour certains aspects. Le premier problème que l'on rencontre avec les bancs de test est leur caractère "propriétaire". Les concepteurs ne mettent pas toujours à disposition toutes les informations permettant de reproduire les bancs de test. Ils décrivent les différents composants utilisés, ils expliquent les tests qui sont réalisés, ils donnent les résultats obtenus avec les différents équipements testés, mais cela ne permet pas pour autant de reproduire les bancs de test. Certaines informations essentielles manquent parfois : le nom des outils/logiciels utilisés, la manière dont sont utilisés les outils/logiciels, etc. Le second problème concerne les tests effectués. Certains bancs de test n'effectuent que quelques tests, ils sont donc loin de permettre une évaluation complète de la sécurité. D'autres bancs de test proposent quant à eux un plus grand nombre de tests, mais ils ne permettent de tester qu'une partie des vulnérabilités relevées à la section 4.1.3 et qu'une partie des attaques reprises à la section 4.2. Le troisième problème concerne un certain manque de cohérence parfois relevé dans les bancs de test. On passe d'un test de *port scanning* à un test de *time synchronisation verification* sans trop comprendre s'il y a un lien logique entre ces deux tests, ou pourquoi ils sont effectués dans cet ordre là. Pour toutes ces raisons, nous pensons qu'on ne peut pas effectuer une évaluation complète et cohérente à l'aide des bancs de test actuels.

5.1 Conception de la méthodologie

L'idée n'est pas de créer une toute nouvelle méthodologie d'évaluation qui viendrait révolutionner le domaine de la sécurité informatique mais de nous baser sur les travaux existants (méthodologies IT et les bancs de test IoT), de reprendre les différentes parties qui sont pertinentes et qui s'appliquent aux IoT, et d'ajouter des nouveaux tests et des nouvelles données afin de créer une méthodologie complète et appropriée aux équipements IoT.

Afin de concevoir cette méthodologie, nous avons premièrement étudié les différentes méthodologies IT présentées à la section 3. Pour chaque méthodologie IT nous avons analysé les principales étapes qui les constituaient. Soit 9 pour NESCOR, 17 pour OSSTMM, 9 pour ISSAF, 3 pour NIST SP800-115. Concernant la méthodologie de l'OWASP, nous avons étudié les trois guides de *pen testing*. Certaines étapes intéressantes étaient présentes dans plusieurs méthodologies, quelques unes étaient propres à une seule méthodologie et d'autres étaient inutiles pour nos besoins. Cette démarche a permis de faire ressortir une première ébauche de notre méthodologie IoT (voir figure 5.1) contenant plusieurs étapes, soit *1. Information gathering*, *2. Architecture review*, *3. Host network analysis*, *4. Vulnerability scan* et *5. reporting*. Les étapes 1 à 4 nous semblaient correspondre à une phase d'analyse ou d'étude du sujet testé. L'étape 5 nous semblait logiquement être l'étape dans laquelle les résultats de l'évaluation seraient regroupés. D'autres étapes présentes dans les méthodologies IT nous semblaient intéressantes et pertinentes. Elles constituaient les étapes de *pen testing*. Nous ne voulions pas les présenter telles qu'elles l'étaient dans les méthodologies IT car elles étaient trop générales et mal adaptées aux IoT. Il nous fallait donc les présenter d'une autre manière, afin qu'elles correspondent mieux au domaine de l'IoT.

Afin de compléter notre méthodologie et afin d'y ajouter de manière adéquate les étapes pertinentes de *pen testing*, nous nous sommes tournés vers les bancs de test. Cela ne nous a malheureusement pas permis de trouver ce que nous cherchions. Les tests de pénétration n'étaient pas soumis à un ordre logique ou à un regroupement spécifique. Après quelques interrogations et réflexions et après avoir réétudié la partie théorique de ce travail, nous avons décidé de répartir les étapes de *pen testing* en 3 nouvelles étapes (voir figure 5.2) qui faisaient référence aux différentes couches d'un système IoT (voir section 2.1), soit les étapes *5. Pen testing - couche perception*, *6. Pen testing - couche réseau* et *7. Pen testing - couche application*. La couche perception reprend les tests qui nécessitent soit un accès direct à l'équipement (ou une certaine proximité) ou soit qui concernent des composants physiques. La couche réseau reprend les tests qui peuvent s'effectuer à travers le réseau (mais qui ne concernent pas la partie applicative). La couche application reprend les tests qui concernent la partie applicative. Cette découpe n'est pas parfaite car il n'est pas facile de classer les attaques ou vulnérabilités. Dans la littérature on retrouve dans certains articles des attaques qui sont assimilées à la partie réseau et ces mêmes attaques sont parfois placées dans la partie applicative dans d'autres articles.

Malgré cela, nous avons gardé la découpe car elle amène certains avantages. Premièrement, cela permet de mieux structurer les tests et de ne pas proposer une grosse étape lourde et complexe qui pourrait effrayer l'utilisateur. Deuxièmement, l'utilisateur qui souhaite appliquer notre méthodologie peut choisir de n'effectuer qu'une partie des tests de pénétration (s'il désire ne tester qu'une couche spécifique de l'architecture).

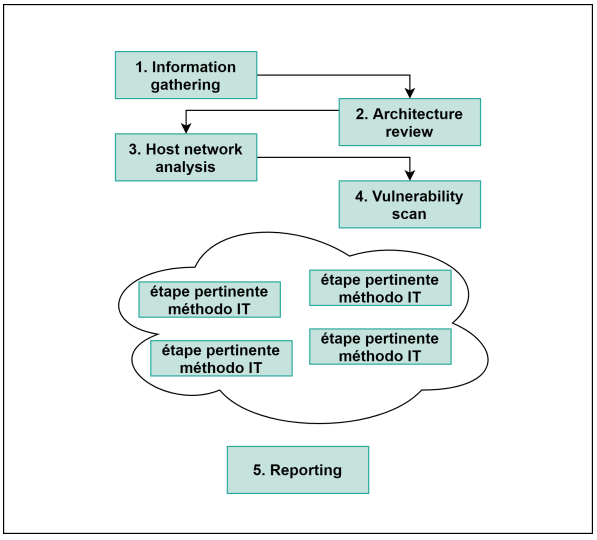


FIGURE 5.1 – Conception méthodologie IoT - étape 1

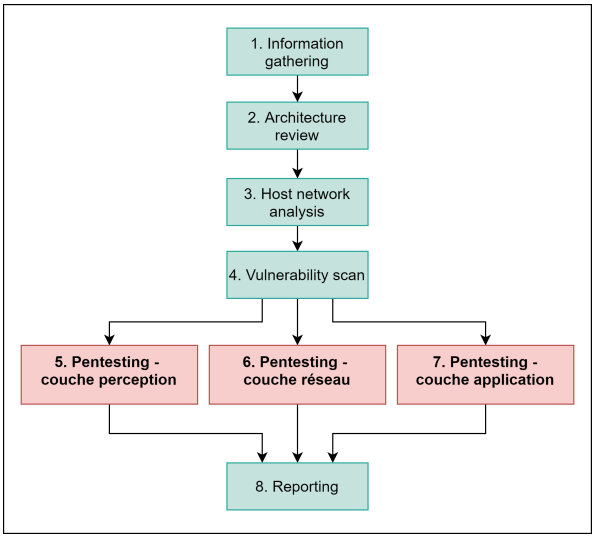


FIGURE 5.2 – Conception méthodologie IoT - étape 2

Nous avons ensuite essayé d'intégrer des étapes qui permettraient de tester toutes les vulnérabilités présentes à la section 4.1.3. Il nous a semblé important de travailler de cette manière car la méthodologie doit évidemment permettre de tester les vulnérabilités qui reviennent le plus souvent dans les problèmes sécuritaires IoT. Nous avons donc obtenu le résultat présent à la figure 5.3. La vulnérabilité n°6 (V6. gestion système) a été placée dans la partie analyse et préparation car ce n'est pas à proprement parler du *pen testing* que de vérifier que l'utilisateur applique les bonnes pratiques de gestion des équipements. La vulnérabilité n°1 n'a pas été insérée car elle touche plusieurs couches du système IoT (on peut récupérer des mots de passe en désassemblant le firmware, on peut faire du *brute force* sur les directories, etc). Des tests relatifs à cette vulnérabilité seront intégrés dans les autres étapes de la méthodologie. Les vulnérabilités 2, 10 et 13 seront incorporées dans l'étape Gestion du système, car elles ne sont pas réellement des étapes de *pen testing*. Nous avons également ajouté une étape "d'exploitation des vulnérabilités". En effet, les vulnérabilités qui ont été mises en avant dans l'étape de scan des vulnérabilités doivent être testées spécifiquement.

Une fois les étapes permettant de tester les vulnérabilités ajoutées, nous avons essayé d'incorporer toutes les attaques reprises à la section 4.2. Cette démarche nous a amené à la figure 5.4. Nous avons ajouté quatre nouvelles sous-étapes car nous n'arrivions pas à classer toutes les attaques recensées. Nous avons ajouté la sous-étape "firmware" dans l'étape 6 *pen testing* couche perception et nous avons ajouté la sous-étape "serveur web" dans l'étape 8 *pen testing* couche application. Nous avons également supprimé l'étape "71. Pratiques de programmation" car cette vulnérabilité est en partie reprise dans l'étape firmware (les mauvaises pratiques de programmation telles que les mots de passe hardcodés ou autres se détectent en analysant le firmware).

Les attaques concernant les protocoles de routage ne sont pas reprises dans la méthodologie car premièrement elles concernent les systèmes IoT et non pas les IoT eux-mêmes et deuxièmement car elles sont déjà partiellement couvertes par les autres tests (en effet les attaques sur les protocoles de routage nécessitent un accès physique au réseau, ou nécessitent un contrôle ou

des actions à distance, via des ports ouverts par exemple).

La dernière étape du processus a été l'analyse des bancs de test. Nous avons parcouru l'ensemble des tests réalisés par les bancs de test et nous avons essayé de les placer dans la méthodologie de la figure 5.4. Nous nous sommes aperçus que cela était possible sans devoir modifier la méthodologie. Nous avons donc décidé de nous arrêter là, il n'était pas nécessaire d'apporter de nouvelles modifications au squelette de la méthodologie.

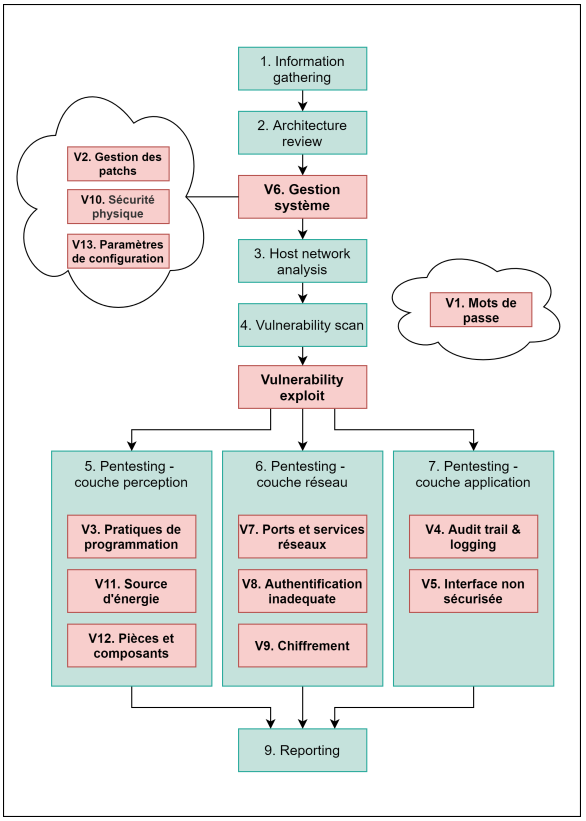


FIGURE 5.3 – Conception méthodologie IoT - étape 3

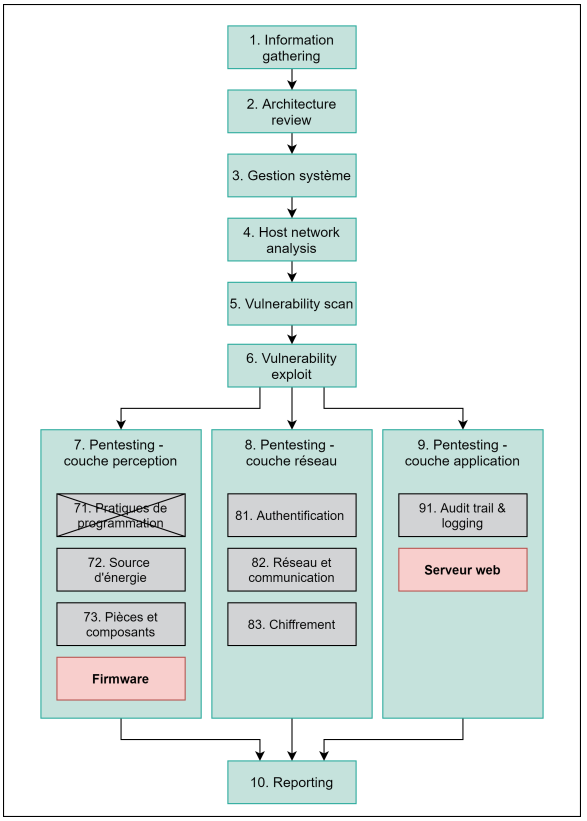


FIGURE 5.4 – Conception méthodologie IoT - étape 4

5.2 Détails de la méthodologie

Nous avons donc établi une méthodologie divisée en trois parties principales. La première partie est la phase **d'analyse** dans laquelle on essaye de récupérer le maximum d'informations sur le système étudié. La deuxième partie est la phase de **pen testing**. C'est à ce moment qu'on effectue les principaux tests de pénétration, qu'on détecte les vulnérabilités, les faiblesses et les problèmes de sécurité. La troisième partie est la phase de **reporting**. On rédige un rapport contenant les résultats de l'évaluation de sécurité.

Chacune des trois phases est divisée en une ou plusieurs étapes. Chaque étape contient une série d'actions à réaliser ou de tests à effectuer. Nous avons choisi d'expliquer et de présenter la méthodologie à l'aide de tableaux. Cela améliore la lisibilité et cela permet de visualiser directement les différentes étapes, les outils proposés, les objectifs, etc. Les différentes étapes sont détaillées ci-dessous.

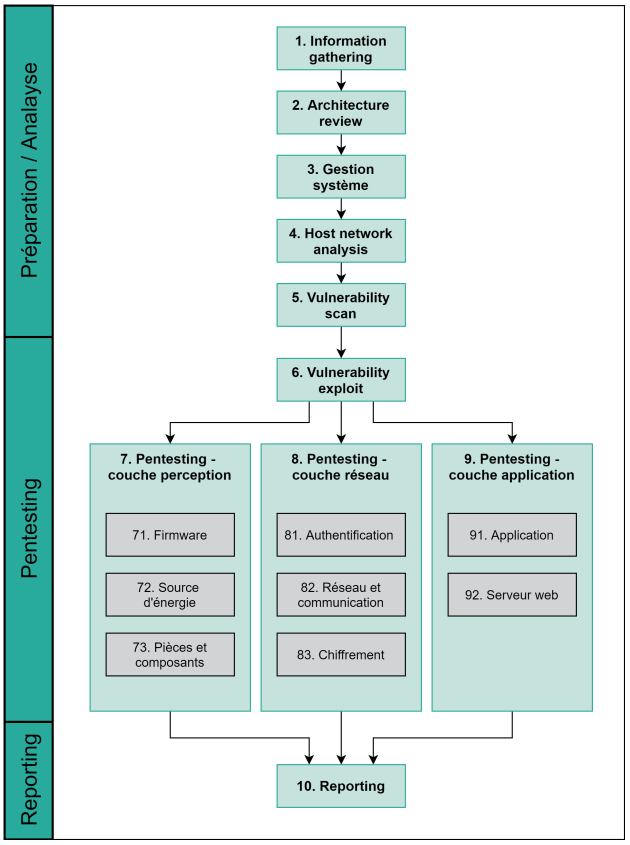


FIGURE 5.5 – Méthodologie de test IoT

1. Information Gathering :
Cette étape consiste à récupérer le maximum d'informations par le biais d'internet et des moteurs de recherche. On parcourt les forums traitant de sécurité, on navigue sur le site web du fabricant, on essaye de trouver des informations sur des problèmes de sécurité qui seraient rendus publics etc. Nous proposons ci-dessous une check-list d'informations pouvant être récupérées. Le test est PASS s'il est effectué, il est FAIL sinon. Il n'y a pas de résultat minimum attendu. Certaines informations ne sont parfois pas disponibles.

1. Information Gathering	
Moteurs de recherche	**Informations à récupérer**
➢ Google ➢ Shodan ➢ Baidu ➢ binsearch.info ➢ Bing ➢ Duck Duck Go ➢ ixquick/Startpage ➢ PunkSpider	✓ Datasheet de l'équipement ✓ Schémas réseau ✓ Configuration de base de l'équipement ✓ Procédure d'identification et format des identifiants ✓ Identifiants par défaut ✓ Messages sur les forums contenant des informations utiles (failles de sécurité détectées, services et ports utilisés, etc.).

2. Architecture review :
Cette étape va de pair avec l'étape 1, mais nous souhaitons insister sur l'importance de l'architecture review. En effet, comme le dit l'OWASP [24], comprendre l'architecture et la configuration utilisée est aussi important que le test de sécurité en lui-même. Le test est PASS s'il est effectué, il est FAIL sinon. Il n'y a pas de résultat minimum attendu. Certaines informations ne sont parfois pas disponibles.

2. Architecture review
Description et objectif(s)
Dans cette étape on étudie l'architecture du système dans sa globalité. ➢ **Objectif** : répondre aux questions ci-dessous. ✓ Quel est le rôle du composant et quelle est sa fonction ? ✓ Quelle est sa place dans l'architecture IoT ? (la figure 2.2 peut s'avérer utile) ✓ Quel type de communication utilise-t-il (câblé, bluetooth, etc.) ? ✓ Avec quels autres équipements communique-t-il ?

3. Gestion du système :
Cette étape permet de vérifier que les bonnes pratiques de gestion des équipements sont respectées. Certains tests peuvent sembler basiques, mais ils permettent néanmoins de sécuriser l'équipement.

3. Gestion système	
Tests	Description, objectif(s) et résultats
31. Updates	Les équipements non mis à jour peuvent présenter des failles de sécurité. ➢ **Objectif** : vérifier que les mises à jour sont effectuées sur l'équipement IoT. On vérifie via l'application smartphone ou directement sur l'équipement. ➢ **Résultat attendu** : le test est PASS si les mises à jours sont effectuées, il est FAIL sinon.
32. Monitoring	Analyser les logs permet de détecter des équipements, des adresses IP et des comportements malicieux et inattendus. ➢ **Objectif** : vérifier qu'il existe un système de logs ou d'évènements. Vérifier que le système est bien monitoré par l'utilisateur. ➢ **Résultat attendu** : le test est PASS si un système de logs est présent, il est FAIL sinon.
33. Sécurité physique	Un attaquant peut récupérer des informations sensibles s'il parvient à se connecter physiquement sur l'IoT. ➢ **Objectif** : vérifier que les IoT ne sont pas directement/physiquement accessibles par un "attaquant". ➢ **Résultat attendu** : le test est PASS si l'équipement n'est pas physiquement accessible, il est FAIL sinon.

34. Paramètres et configuration	Les paramètres et la configuration d'origine ne sont pas souvent modifiés lors de la mise en service de l'équipement. Les attaquants peuvent donc prendre partie de ce manque de gestion du système. ➢ **Objectif** : vérifier que les paramètres ne sont accessibles qu'avec un certain privilège.Vérifier que les paramètres par défaut ont été modifiés. ➢ **Résultat attendu** : le test est PASS si la configuration d'origine a été modifiée et si les paramètres sont protégés par un mot de passe, il est FAIL sinon.
35. Decommissioning	Un équipement non utilisé et toujours présent sur le réseau peut être une brèche dans le système (manque de mises à jour, comportements non surveillés, etc.) ➢ **Objectif** : Vérifier que les équipements non utilisés sont décommissionés. ➢ **Résultat attendu** : le test est PASS s'il n'y a pas d'équipements non utilisés sur le réseau, il est FAIL sinon.

4. Network Port and Service Identification :

Cette étape permet d'identifier les ports et services réseaux utilisés sur l'équipement IoT. Elle permet également de récupérer des informations relatives à l'équipement (Mac adresse, OS, etc.). Ces informations sont essentielles pour la suite car elles influencent les tests qui seront réalisés. Par exemple, si on détecte les ports réseau 80 ou 443, on effectuera des tests en lien avec un serveur web.

4. Network Port and Service Identification		
Tests	Description, objectif(s) et résultats	Outil(s)
41. Port scanning	Les ports réseaux permettent de déterminer les channels de communication qui sont utilisés. On peut également savoir quels sont les points d'accès potentiels à l'équipement. ➢ **Objectif :** tester les 65535 ports UDP et TCP. Récupérer la liste des ports utilisés et leur statut (ouvert, fermé, filtré, ...). ➢ **Résultat attendu** : le test est PASS s'il est effectué, il est FAIL sinon. Il n'y a pas de résultat minimum attendu, il arrive parfois que tous les ports soient fermés.	- netcat - nmap - unicornscan
42. Services scanning	Connaître le service qui tourne sur un port ouvert est intéressant, mais connaître la version de ce service l'est encore plus. Certaines vulnérabilités dépendent de la version des services. ➢ **Objectif :** pour chaque port ouvert ou filtré, déterminer le service qui tourne ainsi que sa version. ➢ **Résultat attendu** : le test est PASS s'il est effectué, il est FAIL sinon. Il n'y a pas de résultat minimum attendu, il arrive parfois que tous les ports soient fermés (il n'y a donc pas de services à découvrir).	- amap - netdiscover - nmap
43. Fingerprinting	En plus des informations précédemment récoltées, on peut aussi récupérer d'autres informations relatives à l'équipement. ➢ **Objectif :** on souhaite récupérer ✓ l'adresse mac. ✓ l'adresse IP. ✓ le nom du fabricant. ✓ l'operating system et sa version. ➢ **Résultat attendu** : le test est PASS s'il est effectué, il est FAIL sinon. Il n'y a pas de résultat minimum attendu, certaines informations ne sont parfois pas accessibles (l'outil peut par exemple renvoyer ce genre de message : *Too many fingerprints match this host to give specific OS details.*).	- amap - nmap - scapy - p0f

5. Vulnerability scan :
Cette étape consiste à répertorier les vulnérabilités de l'équipement IoT qui sont déjà connues et rendues publiques. Plusieurs outils et bases de données existent pour trouver ces vulnérabilités. Si ces dernières sont trouvées sur un appareil du même concepteur, ou sur un appareil du même type, on les teste sur l'équipement IoT cible.

5. Vulnerability scan		
Tests	Description, objectif(s) et résultats	Outil(s)
51. Scan de vulnérabilité	Certains scanners permettent de trouver des vulnérabilités telles que des OS non mis à jours, des problèmes de configuration, des patchs manquants, etc. *Pour ce faire, on identifie les systèmes d'exploitation et les principales applications logicielles exécutées sur les hôtes et on les compare aux informations sur les vulnérabilités connues stockées dans les bases de données de vulnérabilités des scanners* [73]. ➢ **Objectif :** on utilise les scanners (outils) proposés afin de découvrir d'éventuelles vulnérabilités. ➢ **Résultat attendu** : le test est PASS si aucune vulnérabilité n'est présente, il est FAIL sinon.	WiFi : - metasploit - nmap - owasp zap - skipfish - snitch - wascan Bluetooth : - bluescan
52. Bases de données	Des bases de données répertorient les vulnérabilités recensées par les utilisateurs, testeurs, etc. On peut chercher si des vulnérabilités, de l'IoT qu'on souhaite tester, sont connues et renseignées : ➢ **Objectif :** on recherche, dans les bases de données reprises ci-dessous, l'équipement qu'on teste ou des modèles équivalent du même fournisseur afin de voir si des vulnérabilités sont connues et répertoriées (à noter que pour chaque vulnérabilité trouvée dans les bases de données, on récupère les scripts qui permettent d'exploiter la vulnérabilité , s'ils existent). Si des vulnérabilités sont trouvées sur d'autres équipements construits par le même fournisseur, on les teste quand même.) Pour chaque services identifiés à l'étape 42, on cherche des vulnérabilités qui leurs sont associées dans les bases de données. ➢ **Résultat attendu** : le test est PASS si aucune vulnérabilité n'est présente, il est FAIL sinon.	- *Common Vulnerabilities and Exposures database* (CVE). - *the National Vulnerabilities Database (NVD).* - *the Rapid7 Exploits Database.* - *Exploit database.*

6. Vulnerability exploit :
C'est ici que commence la partie *pen testing*. Cette étape a pour but d'essayer d'exploiter les vulnérabilités recensées à l'étape 5.

6. Vulnerability exploit		
Tests	Description, objectif(s) et résultats	Outil(s)
61. Exploitation des vulnérabilités recensées	Des outils et méthodes existent pour tester des vulnérabilités déjà rendues publiques. Le fait d'essayer d'exploiter les vulnérabilités permet de dire si oui ou non la vulnérabilité est bien présente et peut être exploitée. S'il n'y a pas de vulnérabilités détectées aux étapes 51 et 52, ce test n'est pas applicable. ➢ **Objectif :** on utilise les outils proposés afin d'exploiter les vulnérabilités détectées. On exécute les scripts que l'on a éventuellement récupérés dans les bases de données du test 52. ➢ **Résultat attendu** : le test est PASS si aucune vulnérabilité n'a pu être exploitée (il se peut que le fournisseur ait récemment corrigé le problème, via une mise à jour par exemple) ou si le test n'était pas applicable, il est FAIL sinon.	- armitage tools - metasploit

7. Pentesting - couche perception :
Dans cette section, on regroupe les tests qui nécessitent soit un accès physique à l'équipement, soit une certaine proximité ou soit qui concernent les composants physiques. Le workflow de la figure 5.6 permet de déterminer les tests qu'il faut exécuter sur l'équipement IoT.

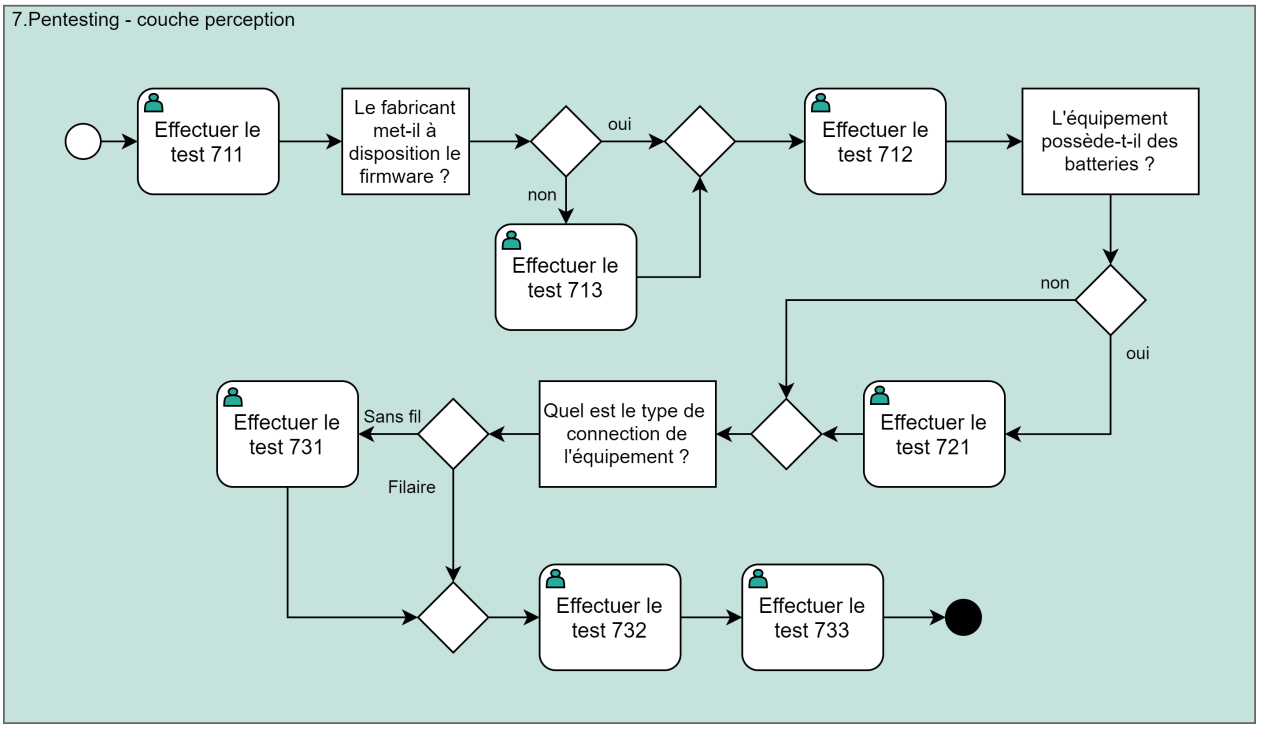


FIGURE 5.6 – Workfolow de l'étape 7 de la méthodologie

L'étape 71 Firmware s'inspire fortement de la méthodologie proposée par l'OWASP [24]. De plus amples informations peuvent être trouvées sur le repository github (voir lien à la section 3.4). Le firmware est une des pièces majeures des équipements IoT. Ces tests permettent de s'assurer que les firmware ne peuvent pas être manipulés afin d'exécuter des fonctions non autorisées et donc potentiellement compromettre la sécurité.

71. Firmware		
Tests	Description, objectif(s) et résultats	Outil(s)
711. Information gathering	Le but de ce test est de récupérer le maximum d'informations concernant le firmware. L'OWASP propose une liste d'informations à récupérer [24]. ➢ **Objectif :** récupérer les informations suivantes ✓ Is firwmare available ? ✓ Supported CPU architecture(s) ✓ Operating system platform ✓ Bootloader configurations ✓ Hardware schematics ✓ Datasheets ✓ Lines-of-code (LoC) estimates ✓ Source code repository location ✓ Third-party components ✓ Open source licenses (e.g. GPL) ✓ Changelogs ✓ FCC IDs ✓ Design and data flow diagrams ✓ Threat models ✓ Previous penetration testing reports ✓ Bug tracking tickets ➢ **Résultat attendu** : le test est PASS s'il est effectué, il est FAIL sinon. Il n'y a pas de résultat minimum attendu.	- moteurs de recherche (voir test 1)
712. Analyse du firmware	Certaines informations sensibles sont présentes dans le firmware. On peut, entre autres, récupérer des login et mots de passe hardcodés (dans des *répertories* tels que "/etc/passwd" or "/etc/shadow"), détecter d'éventuelles backdoors, détecter des vulnérabilités liées à des injections de commandes, etc. ➢ **Objectif :** exécuter les différentes étapes ci-dessous. ✓ *Analyzing firmware* ✓ *Extracting the filesystem* ✓ *Analyzing filesystem contents* ✓ *Emulating firmware* ✓ *Dynamic analysis* ✓ *Runtime analysis* ✓ *Binary Exploitation* ✓ *Modified firmware injection* ➢ **Résultat attendu** : le test est PASS si aucune information sensible n'a été récupérée et si aucun dysfonctionnement n'est apparu, il est FAIL sinon. Il n'y a pas de résultat minimum attendu.	- binwalk
713. Extraction du firmware	Le firmware est parfois disponible sur le site du fournisseur ou sur les moteurs de recherche. Certains fabricants ne donnent cependant pas un libre accès au firmware de leurs IoT. ➢ **Objectif :** récupérer le firmware via une des méthodes proposées ci-dessous (liste non exhaustive). ✓ *from the development team, manufacturer/vendor or client* ✓ *From the vendor's support site* ✓ *Man-in-the-middle (MITM) device communication during updates* ✓ *Extract directly from hardware via UART, JTAG, PICit, etc* ✓ *Dumping firmware from the bootloader (e.g. U-boot) to flash storage or over the network via tftp* ➢ **Résultat attendu** : le test est PASS si le firmware a pu être récupéré, il est FAIL sinon.	- /

72. Source d'énergie		
Tests	Description, objectif(s) et résultats	Outil(s)
721. Sleep deprivation attack	Certaines attaques consistent à solliciter plus que nécessaire un équipement sur batterie afin que celle-ci se décharge plus vite et que l'équipement soit donc rendu inutilisable. ➢ **Objectif :** on envoie un grand nombre de requêtes valides à l'équipement testé afin de solliciter la batterie et afin qu'elle se vide plus vite. Cette attaque est une forme d'attaque DoS (SYN flood). ➢ **Résultat attendu** : le test est PASS si la batterie ne se vide pas plus vite que d'habitude. Le test est FAIL sinon.	- metasploit

73. Pièces et composants		
Tests	Description, objectif(s) et résultats	Outil(s)
731. Jamming attacks	Certains attaquants se munissent d'appareils de *jamming* afin de brouiller le signal sans fil et d'empêcher les équipements de communiquer. ➢ **Objectif :** à l'aide d'un appareil de *jamming*, on envoie un signal d'interférence afin d'empêcher l'équipement de communiquer. On observe alors si la communication est interrompue ou non entre l'IoT et son interlocuteur (*Access point*, serveur, etc.). ➢ **Résultat attendu** : le test est PASS si la communication de l'équipement n'est pas perturbée par le signal d'interférence. Le test est FAIL sinon.	- bands jammer EO-08-007 8
732. Composants utilisés	Certains composants sont utilisés alors qu'ils présentent des failles de sécurité. Cela s'explique principalement pour des raisons financières. Le fournisseur tente d'écouler un stock de pièces dépréciées afin de ne pas perdre d'argent, il achète des composants de moins bonne qualité, etc. ➢ **Objectif :** vérifier que les composants utilisés ne sont pas obsolètes. ➢ **Résultat attendu** : le test est PASS si les composants de l'équipement ne présentent pas de défauts ou vulnérabilités connus et s'ils ne font pas partie des composants décrétés obsolètes. Le test est FAIL sinon.	- /
733. Side channel attack	Certaines informations peuvent être obtenues en analysant les différents composants de l'IoT afin de déterminer le schéma et la clé de chiffrement. Les composants qui sont analysés sont appelés *side channel*. ➢ **Objectif :** Bart Coppens *et al.* proposent d'analyser les *side channel* suivants [25] : ✓ *power consumption behavior* ✓ *instruction or data cache behavior* ✓ *branch predictor behavior* ✓ *pipeline instruction* ✓ *execution behavior* ✓ *pipeline speculation behavior* ➢ **Résultat attendu** : le test est PASS si, après avoir essayé les différentes méthodes reprises ci-dessus, la clé de chiffrement ne peut être obtenue. Il est FAIL sinon.	- /

8.Pentesting - couche réseau :
Dans cette section, on regroupe les tests qui concernent la partie réseau (mais qui ne sont pas en lien avec la partie applicative). Le workflow de la figure 5.7 permet de déterminer les tests qu'il faut exécuter sur l'équipement IoT.

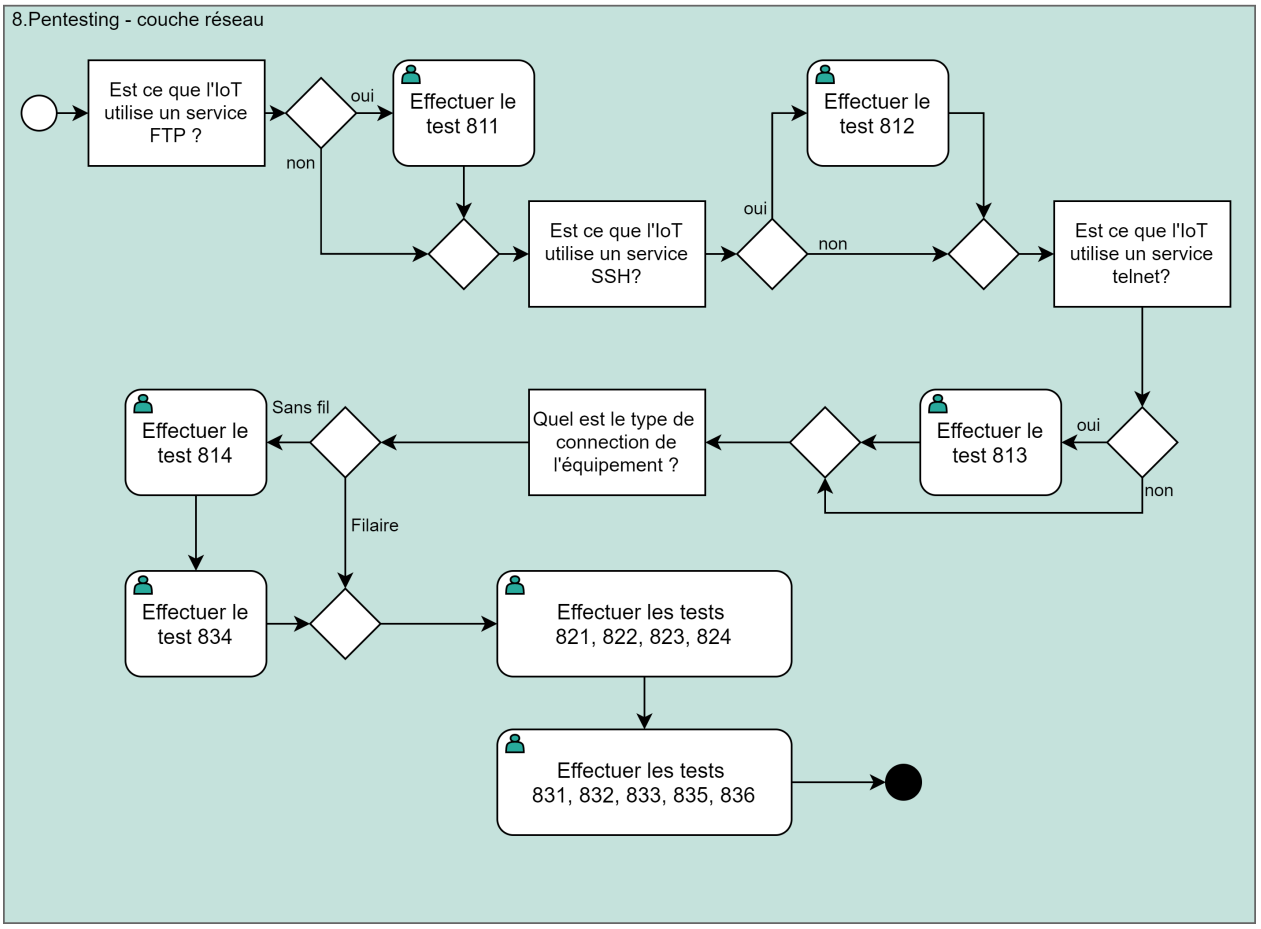


FIGURE 5.7 – Workfolow de l'étape 8 de la méthodologie

81. Authentification		
Tests	Description, objectif(s) et résultats	Outil(s)
811. Brute force FTP	Ce test s'applique si un service FTP a été détecté sur l'équipement (généralement sur les ports TCP 20 et ports TCP/UDP 21). ➢ **Objectif :** on utilise un dictionnaire contenant un ensemble de noms d'utilisateur et de mots de passe potentiels afin de trouver les identifiants de connexion. ➢ **Résultat attendu** : le test est PASS si les identifiants ne sont pas découverts. Le test est FAIL sinon.	- metasploit - hydra - medusa
812. Brute force SSH	Ce test s'applique si un service SSH a été détecté sur l'équipement (généralement sur le port TCP/UDP 22). ➢ **Objectif :** on utilise un dictionnaire contenant un ensemble de noms d'utilisateur et de mots de passe potentiels afin de trouver les identifiants de connexion. ➢ **Résultat attendu** : le test est PASS si les identifiants ne sont pas découverts. Le test est FAIL sinon.	- metasploit - hydra - medusa

813. Brute force telnet	Ce test s'applique si un service telnet a été détecté sur l'équipement (généralement sur le port TCP 23). ➢ **Objectif :** on utilise un dictionnaire contenant un ensemble de noms d'utilisateur et de mots de passe potentiels afin de trouver les identifiants de connexion. ➢ **Résultat attendu** : le test est PASS si les identifiants ne sont pas découverts. Le test est FAIL sinon.	- metasploit - hydra - medusa
814. Deauthentication attack	Si un équipement est connecté à un *access point* (AP), un attaquant peu envoyer une requête de désauthentification au nom de l'équipement en usurpant son adresse MAC. Ce test s'applique aux équipements communicant en WiFi et bluetooth. ➢ **Objectif :** on essaye d'éjecter l'équipement présent sur le réseau (on essaye qu'il se désauthentifie du réseau et qu'il ne soit donc plus connecté). On utilise pour ce faire les outils proposés en pièce jointe et l'adresse MAC de l'équipement (récupérée au test 43). ➢ **Résultat attendu** : le test est PASS si l'équipement reste connecté à l'AP. Le test est FAIL sinon.	Wifi : - airCrack-ng Bluetooth : - script python 2

82. Réseau et communication		
Tests	Description, objectif(s) et résultats	Outil(s)
821. DoS attack	Cette attaque consiste à empêcher l'utilisateur d'accéder à une ressource ou un service. Pour ce faire, on envoie un grand nombre de requêtes. ➢ **Objectif :** on effectue les attaques DoS suivantes ✓ UDP flood ✓ ICMP (Ping) flood ✓ SYN flood ✓ Ping of Death ✓ Slowloris ✓ NTP Amplification ✓ HTTP flood ➢ **Résultat attendu** : le test est PASS si les services ne sont pas interrompus ou rendus inaccessibles. Le test est FAIL sinon.	Wifi : - LOIC - hping3 Bluetooth : - script python 1
822. Replay attack	Cette attaque consiste à intercepter des trames échangées entre l'IoT et son interlocuteur. Une fois les trames interceptées, on les rejoue afin d'essayer de reproduire certains comportements. ➢ **Objectif :** intercepter des trames envoyées et reçues par l'IoT et les rejouer ultérieurement. ➢ **Résultat attendu** : le test est PASS si l'IoT n'accepte pas ou n'exécute pas les requêtes. Le test est FAIL sinon.	WiFi : - / Bluetooth : - spooftooph
823. Communication tampering	Cette attaque consiste à envoyer des trames modifiées (non valides) afin de mettre à mal le fonctionnement de l'équipement. ➢ **Objectif :** intercepter des trames destinées à l'IoT, les modifier afin de les rendre non valides, les envoyer à l'IoT et analyser son comportement. ➢ **Résultat attendu** : le test est PASS si l'IoT n'est pas perturbé et qu'il ne montre pas de signes de dysfonctionnement. Le test est FAIL sinon.	- /
824. Communication avec adresse IP externe	Il arrive parfois qu'un IoT envoie des informations à une adresse IP ou un serveur malveillant. ➢ **Objectif :** on analyse les adresses avec lesquelles communique l'équipement via l'outil proposé. ➢ **Résultat attendu** : le test est PASS si l'IoT ne communique pas avec des adresses malicieuses. Le test est FAIL sinon.	- wireshark

83. Chiffrement		
Tests	Description, objectif(s) et résultats	Outil(s)
831. Analyse de la communication	Pour des raisons de sécurité évidentes, on désire que les informations qui transitent sur le réseau (données, mots de passe, clés, etc.) soient chiffrées. ➢ **Objectif :** à l'aide d'un outil permettant de lire tous les paquets qui transitent sur le réseau, on vérifie que les données qui transitent sont chiffrées. ➢ **Résultat attendu** : le test est PASS si on ne sait pas lire les données (si elles sont chiffrées). Le test est FAIL sinon.	Wifi : - wireshark Bluetooth : - kismet
832. Updates chiffrées	Les mises à jour peuvent contenir des informations sensibles (des corrections de problèmes existants, etc.) qu'il ne faut pas divulguer. On s'assure donc qu'elles sont chiffrées. ➢ **Objectif :** à l'aide d'un outil permettant de lire tous les paquets qui transitent sur le réseau, on vérifie que les mises à jour qui s'effectuent sont chiffrées. ➢ **Résultat attendu** : le test est PASS si les mises à jour sont chiffrées. Le test est FAIL sinon.	- wireshark
833. Downgrading attack	Cette attaque vise à réduire le niveau de chiffrement. On essaye de passer de l'utilisation de HTTPS vers HTTP. ➢ **Objectif :** à l'aide des outils proposés, on vérifie que l'équipement n'accepte pas les requêtes HTTP. ➢ **Résultat attendu** : le test est PASS si l'équipement refuse d'utiliser les requêtes HTTP. Le test est FAIL sinon.	- SSLStrip - ettercap - better-cap
834. Key reinstallation attack (KRACK)	Cette attaque utilise une faille dans le standard WiFi. La réinstallation de la clé permet de lire des informations qui étaient auparavant supposées être chiffrées. ➢ **Objectif :** on utilise le script proposé afin de déterminer si l'équipement est sensible à cette faille. ➢ **Résultat attendu** : le test est PASS si les informations restent chiffrées. Le test est FAIL sinon.	- script python 3
835. MITM	L'attaque *man in the middle* (MITM) consiste à se positionner entre deux équipements qui communiquent et échangent des informations. On se fait passer pour l'équipement 2 aux yeux de l'équipement 1 et inversement. ➢ **Objectif :** on réalise une attaque MITM à l'aide des outils proposés. ➢ **Résultat attendu** : le test est PASS si on ne récupère aucune information sensible à l'aide de l'attaque. Le test est FAIL sinon.	- MITMf - MITM proxy
836. Cryptanalysis Attacks	Cette attaque repose sur l'analyse de l'algorithme de chiffrement, ou l'analyse des textes chiffrés. Le but de cette attaque est de déduire le schéma de cryptage du système. ➢ **Objectif :** on réalise, en fonction des informations possédées (texte en clair, clé, texte chiffré) une des attaques suivantes : ✓ Known-Plaintext Analysis (KPA) ✓ Chosen-Plaintext Analysis (CPA) : ✓ Ciphertext-Only Analysis (COA) : ✓ Man-In-The-Middle (MITM) attack : ✓ Adaptive Chosen-Plaintext Analysis (ACPA) : ➢ **Résultat attendu** : le test est PASS si on ne sait pas déduire le schéma de cryptage du système. Le test est FAIL sinon.	- /

9. Pentesting - couche application :

Dans cette section, on regroupe les tests qui concernent la partie application et serveur web. Le workflow de la figure 5.8 permet de déterminer les tests qu'il faut exécuter sur l'équipement IoT.

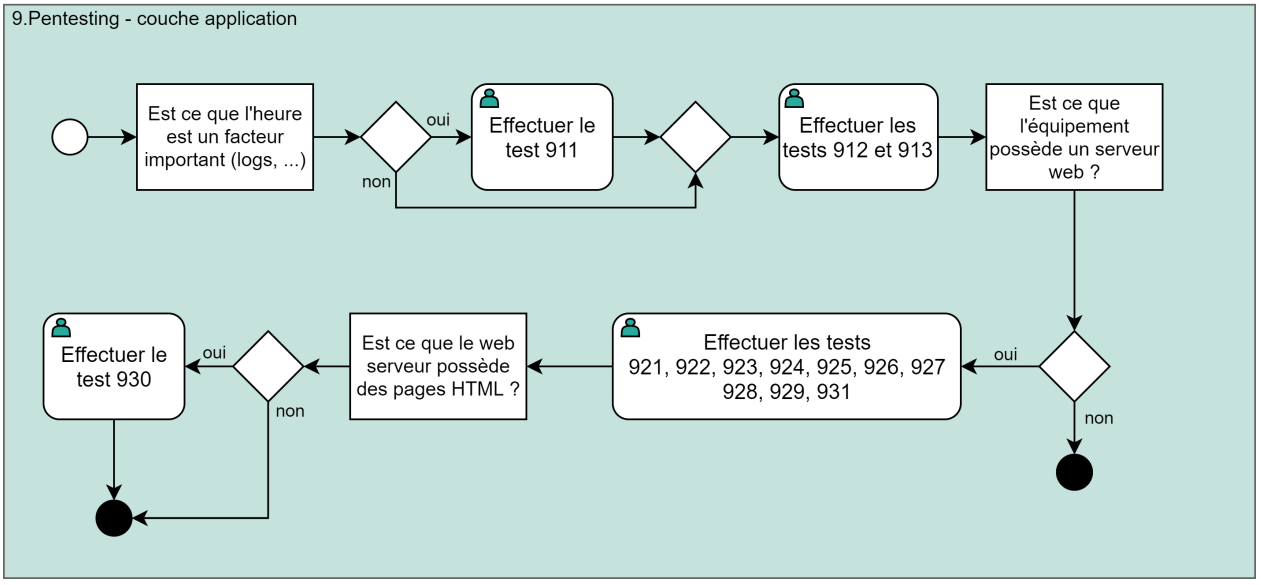


FIGURE 5.8 – Workfolow de l'étape 9 de la méthodologie

91. Application		
Tests	Description, objectif(s) et résultats	Outil(s)
911. Vérifier synchronisation NTP	Il est parfois important qu'un équipement soit parfaitement à l'heure. Soit parce que c'est important pour son fonctionnement (une pompe à insuline par exemple) ou soit parce que l'équipement possède un système de logs (ce qui permet d'enregistrer les évènements à la bonne heure). ➢ **Objectif :** on vérifie que l'IoT est synchronisé avec un serveur de temps. A l'aide d'un outil permettant de lire tous les paquets qui transitent sur le réseau, on vérifie que des paquets NTP sont échangés. ➢ **Résultat attendu** : le test est PASS si des paquets NTP sont observés. Le test est FAIL sinon.	- wireshark
912. Désassemblage de l'application mobile	Certains outils permettent de désassembler les applications mobiles afin d'obtenir du code java. ➢ **Objectif :** On utilise les outils proposés afin de désassembler l'application mobile. Une fois le code java obtenu, on essaye de trouver des informations sensibles. ➢ **Résultat attendu** : le test est PASS si on ne trouve pas de données sensibles (clés, mots de passe, etc.). Le test est FAIL sinon.	- dex2jar - JD-GUI

913. Source Code Analyzers	L'analyse statique de code, souvent réalisée via des scanners, permet de mettre en évidence dans le code d'une application des erreurs de programmation, des mauvaises pratiques, des standards non respectés, etc. ➢ **Objectif :** on utilise les outils afin de trouver des bugs ou des mauvaises pratiques de programmation dans le code source. ➢ **Résultat attendu** : le test est PASS si on ne trouve pas de bugs ou de problèmes de sécurité. Le test est FAIL sinon.	- spotBugs - find security bugs - PMD - sonarQube community edition
914. Bibliothèques logicielles	Certaines bibliothèques sont utilisées alors qu'elles présentent des failles de sécurité. Cela peut par exemple s'expliquer car certains développeurs utilisent par habitude les mêmes bibliothèques au fil des projets, sans s'inquiéter des éventuels problèmes de sécurité qui auraient été relevés par la communauté. ➢ **Objectif :** vérifier que les bibliothèques utilisées ne sont pas obsolètes. ➢ **Résultat attendu** : le test est PASS si les versions et les bibliothèques en elles-mêmes ne sont pas obsolètes. Le test est FAIL sinon.	- /

Pour la partie serveur web, l'OWASP Web Security Testing Guide [24] propose une méthode de test très complète sur leur repository github (voir 3.4). Nous proposons ci-dessous une version simplifiée de test d'un serveur web IoT qui s'inspire, comme expliqué précédemment, de la littérature, des bancs des test existants, de l'OWASP WSTG, etc.

92. Serveur web		
Tests	Description, objectif(s) et résultats	Outil(s)
921. Scan vulnérabilités	Certains outils permettent de faire un scan du serveur web afin de récupérer des informations telles que le plan du site, l'ensemble des répertoires, les vulnérabilités, etc. ➢ **Objectif :** à l'aide des outils proposés, on effectue un scan du serveur web. ➢ **Résultat attendu** : le test est PASS si l'outil n'a pas détecté de problèmes de sécurité. Le test est FAIL sinon.	- grabber - skipfish
922. Vérifier le firewall	Certains serveurs web possèdent leur propre firewall afin de se protéger des attaques. ➢ **Objectif :** on vérifie si l'IoT utilise un firewall ou non. ➢ **Résultat attendu** : le test est PASS si l'outil a détecté un firewall. Le test est FAIL sinon.	- WAFWoof
923. Vérifier certificat SSL	Afin de pouvoir naviguer en toute sécurité avec des pages HTTPS, le serveur web doit être reconnu par une autorité de certification et posséder un certificat SSL valide. ➢ **Objectif :** on vérifie que l'IoT possède un certificat SSL valide. ➢ **Résultat attendu** : le test est PASS si l'outil a détecté un certificat valide. Le test est FAIL sinon.	- SSLScan - nikto tools
924. Vérification des identifiants	Ce test vise à vérifier les règles en vigueur autour du mécanisme de connexion et de déconnexion des utilisateurs. ➢ **Objectif :** on vérifie les points suivants : ✓ Complexité minimum requise pour le mot de passe ✓ Mécanisme d'*auto logout* ✓ Nombre maximum d'essais ratés consécutifs ✓ On teste un dictionnaire de mots de passe par défaut ➢ **Résultat attendu** : le test est PASS en fonction des critères d'acceptation de l'utilisateur. Le test est FAIL sinon.	- /

925. Bypass basic HTTP authentication	*Certains serveurs web utilisent l'authentification de base HTTP pour obtenir les informations d'identification des utilisateurs. Les requêtes HTTP peuvent utiliser les méthodes POST et GET. Si les serveurs sont faiblement configurés, ils peuvent contourner les demandes d'authentification HTTP qui ont des méthodes HTTP autres que GET/-POST* [83]. ➢ **Objectif :** on essaye d'utiliser des requêtes http afin de voir si on arrive à avoir accès à des contenus protégés. ➢ **Résultat attendu** : le test est PASS si l'accès n'est pas autorisé. Le test est FAIL sinon.	- hydra
926. Brute force directories	Certains outils sont dits des "scanners de contenus". Ces outils permettent de trouver des *directories* ou des objets cachés sur le serveur web. ➢ **Objectif :** on vérifie que tous les *directories* sont protégés par un mot de passe. On effectue également un test de brute force. ➢ **Résultat attendu** : le test est PASS si l'accès aux *directories* nécessite une identification et si les outils n'ont pas trouvé de mots de passe. Le test est FAIL sinon.	- dir - dirBuster
927. Injection SQL	Cette attaque consiste à modifier les requêtes SQL que l'application envoie à la base de données, dans le but d'obtenir des informations qui sont de base non accessibles. ➢ **Objectif :** on vérifie que le *webserver* n'est pas sensible à des injections SQL. ➢ **Résultat attendu** : le test est PASS si l'outil ne permet pas d'obtenir des informations sensibles. Le test est FAIL sinon.	- SQLmap
928. Injection XSS	Cette attaque consiste à "injecter" du code dans une page web afin de réaliser certaines actions malicieuses lorsque l'utilisateur utilise ces pages modifiées. ➢ **Objectif :** on vérifie que le *webserver* n'est pas sensible à des injections XSS. ➢ **Résultat attendu** : le test est PASS si l'outil ne permet pas d'obtenir des informations sensibles. Le test est FAIL sinon.	- XSSer
929. HTML analysis	Certains serveurs web peuvent contenir des pages HTML. Celles-ci peuvent contenir des bugs ou des vulnérabilités. ➢ **Objectif :** scanner les pages HTML afin de détecter des problèmes de sécurité. ➢ **Résultat attendu** : le test est PASS si l'outil ne détecte pas de problèmes. Le test est FAIL sinon.	- /
930. Specific attacks	Les pages webs sont la cible de nombreuses attaques spécifiques. ➢ **Objectif :** vérifier, pour chacune des attaques reprises ci-dessous, s'il est possible d'obtenir des informations sensibles ou effectuer des actions malicieuses. ✓ CSRF ✓ Virus/trojan/... ✓ Clickjacking ✓ Hijacking ✓ Phishing ✓ XCS ✓ Reverse XCS ➢ **Résultat attendu** : le test est PASS si les attaques reprises ci-dessus n'ont pas permis d'obtenir des informations sensibles, il est FAIL sinon.	- /

10. Reporting :

L'étape de reporting synthétise et compile toutes les informations obtenues dans les tests précédents. Nous présentons ci-dessous un exemple de tableau qui pourrait être utilisé à cet effet. Nous présentons également un exemple de test réalisé qui serait synthétisé dans le tableau.

Module testé				
Test	Intitulé du test	Statut du test	Commentaire	Résultat
911.	Synchronisation NTP	Effectué	A chaque mise sous tension de l'équipement, on observe des paquets NTP qui transitent.	PASS

La première colonne contient le numéro du test. La deuxième colonne reprend l'intitulé du test. La troisième colonne contient le statut du test (effectué, non effectué, non applicable). La quatrième colonne donne une information sur le test (une justification si le test n'a pas été effectué, un problème de sécurité détecté, etc.). La dernière colonne donne un résultat pour le test (PASS si l'objectif attendu est atteint et qu'il n'y a pas de problème de sécurité repéré, TBT (To be Tested) si le test n'a pu être réalisé, FAIL dans les autres cas).

Une fois que le tableau a été rempli, si tous les résultats des tests sont PASS, l'équipement est jugé sécurisé. Si au moins un des tests est au statut TBT ou FAIL, c'est à l'utilisateur à juger du risque encouru en fonction de la vulnérabilité, de la fonction de l'équipement, etc.

5.3 Finalité de la méthodologie

Nous avons pensé la méthodologie pour qu'elle s'utilise comme un guide et un aide-mémoire. Nous ne souhaitions pas concevoir quelque chose trop lourd et trop détaillé. L'objectif premier de la méthodologie est de pouvoir réaliser un test de sécurité complet tout en étant facilement reproductible. Nous estimons avoir renseigné les informations minimales permettant de comprendre et de réaliser chaque test.

Il est important de signaler que cette méthodologie est destinée à des personnes maîtrisant au minimum les concepts de base de la sécurité informatique et familières des outils de *pen testing* et d'analyse. Certains tests requièrent des connaissances précises et pointues dans le domaine de l'informatique et de l'électronique.

Certains tests contiennent peu d'informations ou ne possèdent pas d'outils renseignés. Certains tests sont également assez théoriques et complexes. Nous les avons tout de même repris dans la méthodologie. Même si certains tests ne sont pas exécutés (faute de moyens, de connaissances techniques, de temps, etc.), le rapport de sécurité obtenu à l'étape 10 de la méthodologie reprendra ces tests et les utilisateurs et les testeurs garderont à l'esprit que des potentielles vulnérabilités n'ont pas été testées.

Chapitre 6

Application et évaluation de la méthodologie

6.1 Objets IoT étudiés

Nous avons décidé de tester et d'appliquer la méthodologie sur deux équipements IoT. La mise en pratique de concepts théoriques permet souvent de repérer des problèmes ou des exceptions qui s'adaptent mal à la théorie. Cela permet donc d'améliorer le modèle théorique (ici la méthodologie) de base. Nous présentons ci-dessous les équipements analysés :

- Caméra Imilab home - Xiaomi : cette caméra est présente dans de nombreux commerces. Il est donc raisonnable de penser qu'elle est assez présente dans le paysage IoT et c'est donc une bonne raison que de l'étudier. Cette caméra, surtout utilisée pour un usage domestique, est alimentée sur le secteur et communique via le réseau sans fil **WiFi** (voir figure 6.1a) ;
- Doorbell video v7 - EKEN : cette sonnette intelligente permet d'envoyer des notifications (message et vidéo) lorsqu'un mouvement est détecté. Elle est alimentée par des piles. Elle communique avec l'application via le **WiFi**. (voir figure 6.1b) ;
- Printer ENVY 7830 - HP : cette imprimante multi-fonction permet d'imprimer et de numériser des photos et des documents, soit depuis le réseau, soit depuis un support USB ou une carte SD. Dans le cadre de nos tests, elle communique avec le réseau via le **WiFi** (elle pourrait également être connectée via un câble réseau). (voir figure 6.1c) ;

(a) Imilab home (b) Doorbell v7 (c) Envy 7830

FIGURE 6.1 – IoT étudiés

Les deux premiers équipements étudiés (6.1a et 6.1b) sont assez similaires. Ils envoient des images, sont utilisés pour un usage domestique et sont commandés via une application smartphone. Les tests de sécurité effectués seront donc relativement similaires et nous pourrons facilement comparer les résultats. Le troisième équipement est assez différent des deux premiers, mais il est également utilisé pour un usage domestique. Les trois appareils communiquent via le WiFi.

6.2 Environnement de test

Afin de pouvoir tester la méthodologie, nous avons mis en place un "environnement de test". Nous utilisons premièrement un *access point* (AP) Ubiquiti Networks UAP-AC-PRO. Celui-ci est connecté au port numéro 1 d'un *switch manageable* hp-1810-24g (modèle J9803A). Le PC, sur lequel tourne une machine virtuelle Kali Linux version 2021.1, est connecté sur le port numéro 3 du switch. Les ports 1 et 3 sont en *mirroring*, les paquets qui transitent sur le port 1 sont recopiés sur le port 3. Le port 24 est connecté au modem internet. La figure 6.2 schématise le premier environnement. Nous utilisons deuxièmement une carte WiFi (Alfa AWUS06ACM), capable de passer en mode monitoring, couplée avec la machine virtuelle kali Linux. La figure 6.3 schématise le deuxième environnement. Ces deux environnements de test nous permettent de tester et d'appliquer la méthodologie.

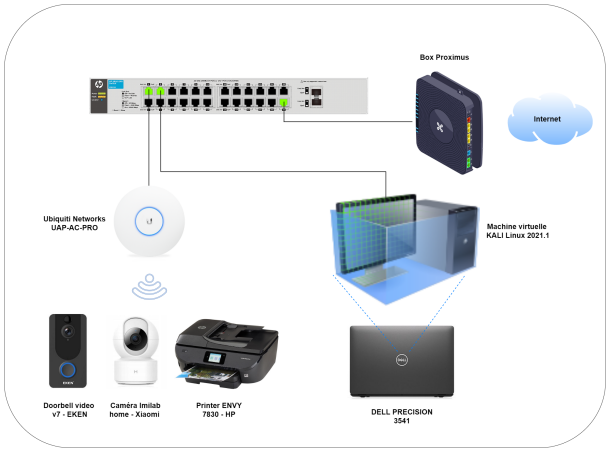


FIGURE 6.2 – Environnement de test 1

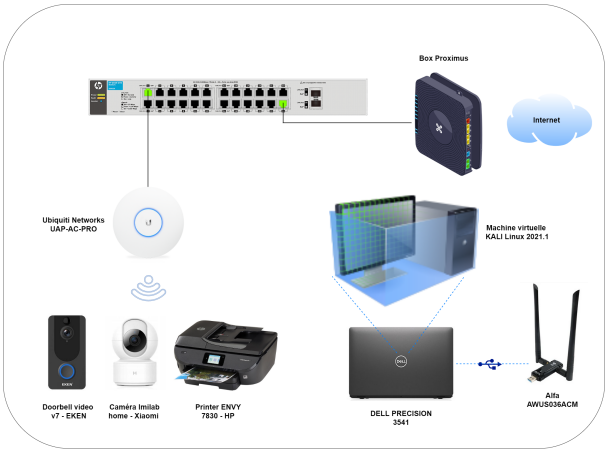


FIGURE 6.3 – Environnement de test 2

6.3 Application de la méthodologie

A l'aide des environnements de test présentés à la section précédente et en nous aidant de la méthodologie précédemment élaborée, nous allons tester la sécurité des trois équipements IoT. Nous présentons et détaillons dans cette section les tests réalisés. Cette section comprendra les étapes 1 à 9 de la méthodologie, l'étape 10 sera présentée à la section 6.4. Afin de ne pas alourdir inutilement le présent travail, les résultats des tests jugés peu intéressants ne seront pas détaillés ci-dessous (ils seront néanmoins repris dans l'étape 10). Les commandes et outils utilisés ne seront détaillés qu'une seule fois, et non pas à chaque utilisation.

6.3.1 Analyse Caméra Imilab Home

➥ Test 41. *Port scanning* :

Nous avons utilisé l'outil nmap afin d'effectuer ce test. Les 65535 ports TCP sont fermés. Les principaux ports UDP sont fermés, mais nmap a détecté que certains ports UDP étaient ouverts-filtrés. Il n'y a pas d'accès SSH, FTP ou telnet.

```
# permet de scanner les ports TCP
$ sudo nmap -sS -vv -p1-65535 -oN nmapResultsImilabTCP.txt 192.168.1.17
# permet de scanner les ports UDP
$ sudo nmap -sU -vv -p1-65535 -oN nmapResultsImilabUDP.txt 192.168.1.17
```

➥ Test 43. *Fingerprinting* :

Nous avons utilisé l'outil nmap afin d'effectuer ce test. Cela nous a permis de récupérer les informations suivantes :
Adresse MAC = 78 :8b :2a :a3 :4a :be
Vendeur = Zhen Shi Information Technology (Shanghai)
OS = /.

```
# permet de scanner les ports TCP
$ sudo nmap -A -oN nmapResultsImilabFingerPrint.txt 192.168.1.17
```

➥ Test 51. *Vulnerability scan* :

Nous avons utilisé l'outil nmap (vulscan) afin d'effectuer ce test. Ce scan teste les vulnérabilités présentes dans les bases de données CVE. Nous n'avons pas trouvé de vulnérabilités.

```
# permet d'importer les scripts pour vulscan
$ sudo git clone https://github.com/scipag/vulscan scipag_vulscan
$ sudo ln -s `pwd`/scipag_vulscan /usr/share/nmap/scripts/vulscan
# permet de lancer le scan vulscan
$ sudo nmap -sV --script=vulscan/vulscan.nse 192.168.1.17
```

➥ Test 713. *Firmware extraction* :

Après avoir démonté l'équipement, nous avons observé deux connexions, RX0 et TX0 (voir figure 6.4). Nous avons essayé de connecter un adaptateur série TTL (voir figure 6.5) sur ces pins, mais nous n'y sommes pas parvenus. Les soudures ne tenaient pas sur les connections RX0 et TX0 car ces dernières étaient trop "lisses". Ne disposant pas d'autre matériel, il n'a pas été possible d'extraire le firmware. Le test 711 (*firmware information gathering* n'a pas permis de récupérer le firmware non plus. Il n'est donc pas possible d'effectuer le test 712 *firmware analyse*.

➥ Test 816. *Déauthentifaction attack* :

Nous avons utilisé les outils repris dans le pack [Aircrack-ng]. Après avoir lancé l'attaque, la caméra se déconnecte du réseau, une fois l'attaque arrêtée, elle se reconnecte par elle-même. Le test est FAIL, car la caméra se déconnecte du WiFi.

```
# permet de recuperer l'adresse MAC de l'access point
$ sudo airodump-ng wlan0
# permet de demander une deauth pour la camera
$ sudo aireplay-ng --deauth 0 -c 78:8b:2a:a3:4a:be -a 68:d7:9a:81:37:0e -D wlan0
```

➥ Test 821. *DoS attack* :

Nous avons utilisé les outils LOIC et hping3 pour réaliser ce test. Nous avons essayé les attaques

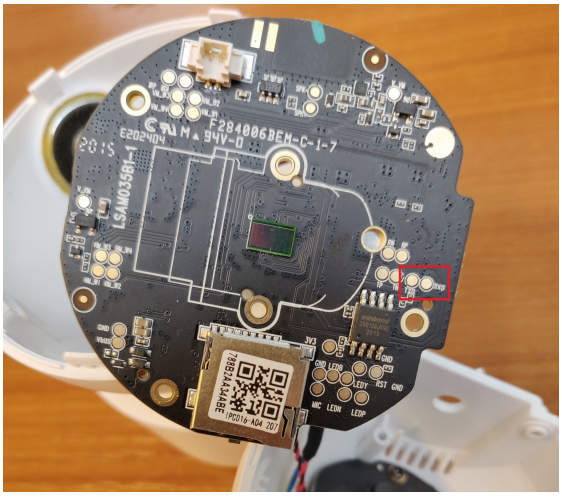


FIGURE 6.4 – Circuit imprimé caméra

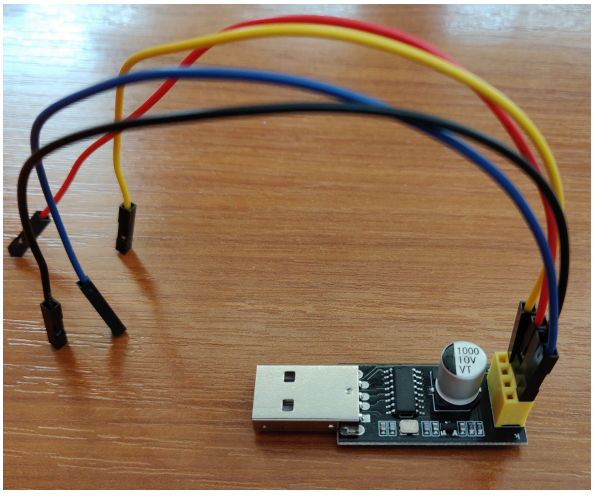
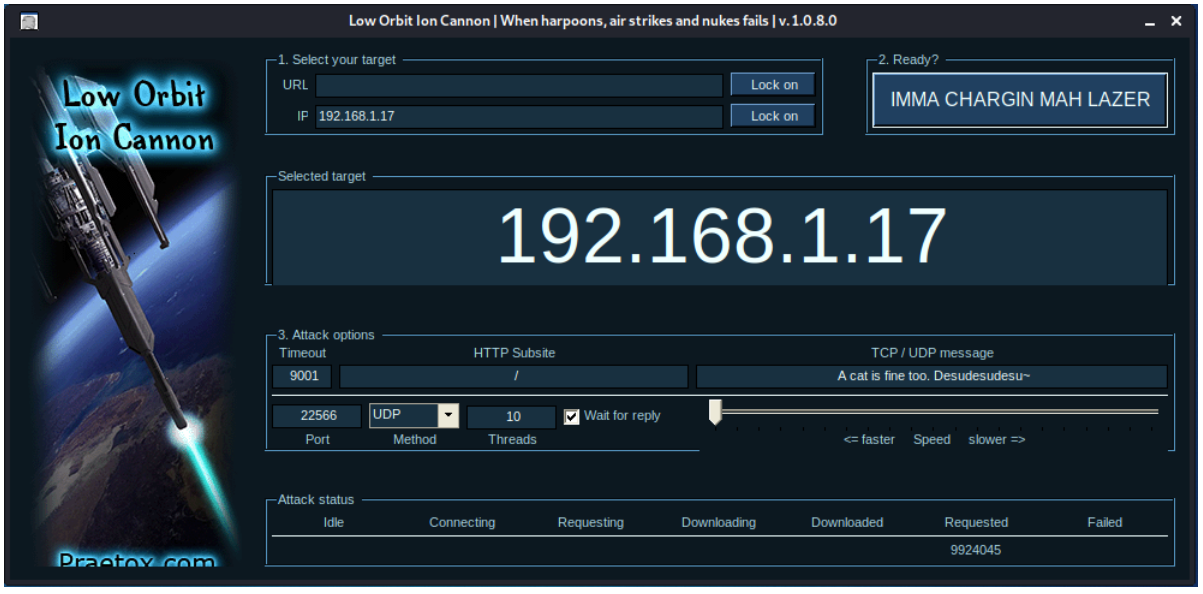


FIGURE 6.5 – Adaptateur série TTL

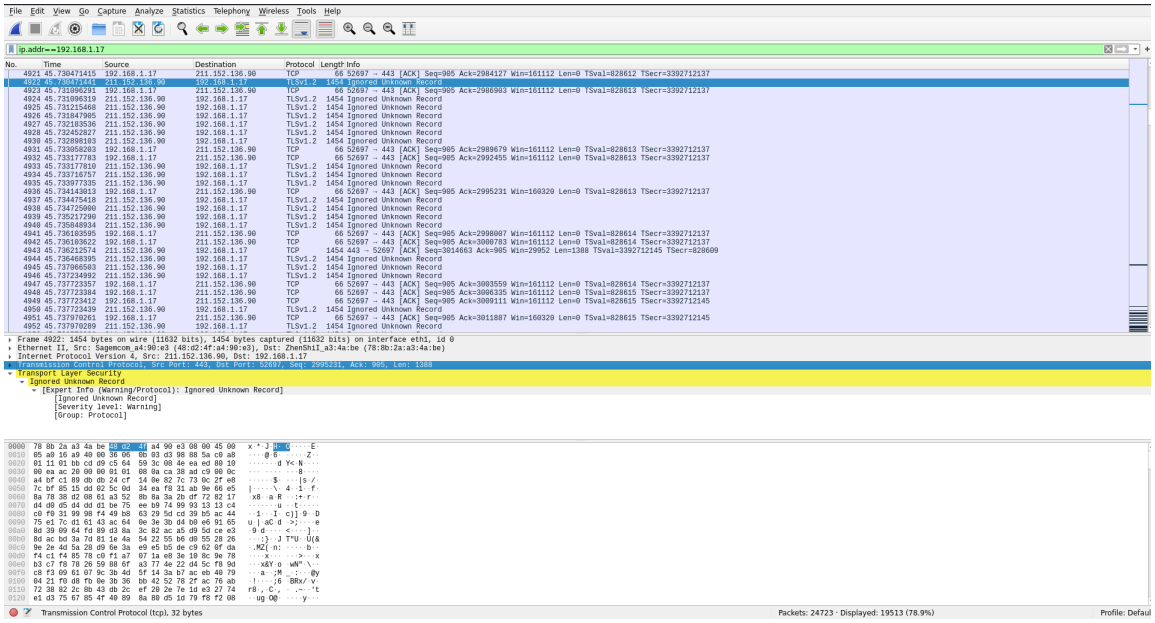
reprises dans la méthodologie. Après avoir lancé les attaques, la caméra réagit plus lentement mais elle est toujours fonctionnelle. Le test est PASS.




```
# attaque udp flood
$ sudo hping3 -2 --flood 192.168.1.17
# attaque icmp flood
$ sudo hping3 -1 --flood 192.168.1.17
```

➥ Test 832. *Updates chiffrées* :

Nous avons enregistré le trafic avec Wireshark pendant la mise à jour de la caméra. Une fois le trafic analysé, nous avons vu que la mise à jour s'effectuait grâce à TLS V1.2. Le test est PASS.



➥ Test 912. Désassemblage application mobile :

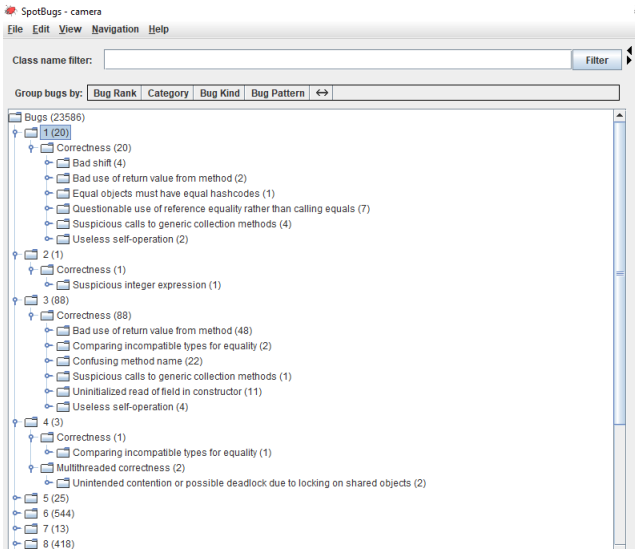
Afin de désassembler l'application mobile nous avons :

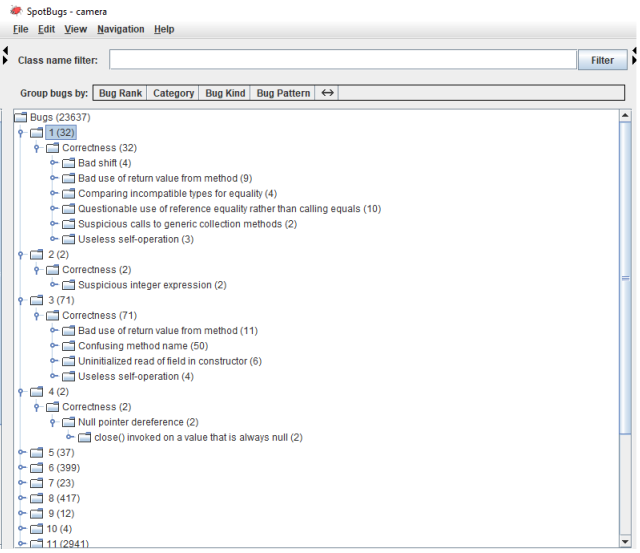
- installé dex2jar sous windows
- récupéré le fichier apk de l'application android qui contrôle la caméra (la version du fichier est v6.5.700).
- renommé le fichier .apk en .zip et désarchivé le fichier .zip
- récupéré les fichiers classes.dex contenus dans le dossier désarchivé (il y en avait 8 au total).
- lancé la commande ci-dessous 8 fois (en modifiant le nom du fichier à chaque fois) afin de récupérer 8 fichiers .jar (il faut se trouver dans le dossier dex2jar).

```
# obtenir un fichier .jar
$ d2j-dex2jar.bat classes.dex
```

➥ Test 913. *Source code analyzer* :

Nous avons utilisé l'outil Spotbugs afin d'analyser le fichier .jar précédemment obtenu. Nous avons obtenu beaucoup d'erreurs, mais elles n'ont pas l'air de mettre en péril la sécurité de la caméra. Nous n'avons pas trouvé de mots de passe hardcodés ou autres. Il y avait 8 fichiers .dex, nous avons donc lancé deux scans distincts afin d'alléger le processus d'analyse.


SpotBugs - camera
File Edit View Navigation Help
Class name filter:
Filter
Group bugs by:
Bug Rank
Category
Bug Kind
Bug Pattern
Bugs (23586)
1 (20)
Correctness (20)
Bad shift (4)
Bad use of return value from method (2)
Equal objects must have equal hashcodes (1)
Questionable use of reference equality rather than calling equals (7)
Suspicious calls to generic collection methods (4)
Useless self-operation (2)
2 (1)
Correctness (1)
Suspicious integer expression (1)
3 (88)
Correctness (88)
Bad use of return value from method (48)
Comparing incompatible types for equality (2)
Confusing method name (22)
Suspicious calls to generic collection methods (1)
Uninitialized read of field in constructor (11)
Useless self-operation (4)
4 (3)
Correctness (1)
Comparing incompatible types for equality (1)
Multithreaded correctness (2)
Unintended contention or possible deadlock due to locking on shared objects (2)
5 (25)
6 (544)
7 (13)
8 (418)


SpotBugs - camera
File Edit View Navigation Help
Class name filter:
Filter
Group bugs by:
Bug Rank
Category
Bug Kind
Bug Pattern
Bugs (23637)
1 (32)
Correctness (32)
Bad shift (4)
Bad use of return value from method (9)
Comparing incompatible types for equality (4)
Questionable use of reference equality rather than calling equals (10)
Suspicious calls to generic collection methods (2)
Useless self-operation (3)
2 (2)
Correctness (2)
Suspicious integer expression (2)
3 (71)
Correctness (71)
Bad use of return value from method (11)
Confusing method name (50)
Uninitialized read of field in constructor (6)
Useless self-operation (4)
4 (2)
Correctness (2)
Null pointer dereference (2)
close() invoked on a value that is always null (2)
5 (37)
6 (399)
7 (23)
8 (417)
9 (12)
10 (4)
11 (2941)

6.3.2 Analyse video Doorbell V7

➥ Test 41. *Port scanning* :

Nous avons utilisé l'outil nmap afin d'effectuer ce test. Les 65535 ports TCP sont fermés. Les principaux ports UDP sont fermés, mais nmap a détecté que certains ports UDP étaient ouverts-filtrés. Il n'y a pas d'accès SSH, FTP ou telnet.

➥ Test 43. *Fingerprinting* :

Nous avons utilisé l'outil nmap afin d'effectuer ce test. Cela nous a permis de récupérer les informations suivantes :
Adresse MAC = 68 :b9 :d3 :0f :af :a8.
Vendeur = Shenzhen Trolink Technology.
OS = / .

➥ Test 713. *Firmware extraction* :

Après avoir démonté l'équipement, nous n'avons pas trouvé de prise UART ou JTAG (voir figures 6.6 et 6.7. Il n'est donc pas possible de récupérer le firwmare de cette façon. Le test 711 (*firmware information gathering* n'a pas permis de récupérer le firmware non plus. Il n'est donc pas possible d'effectuer le test 712 *firmware analyse*.

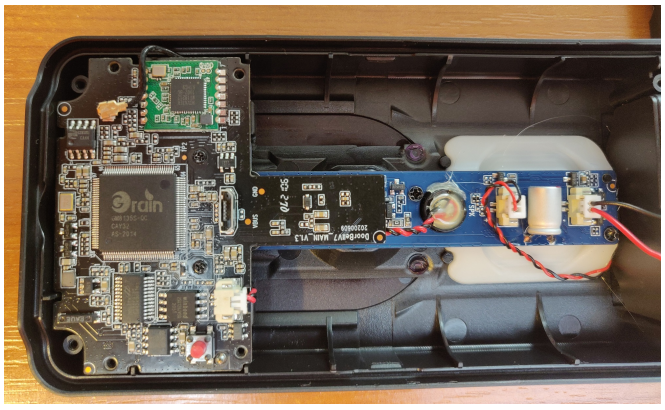


FIGURE 6.6 – Circuit imprimé Doorbell 1

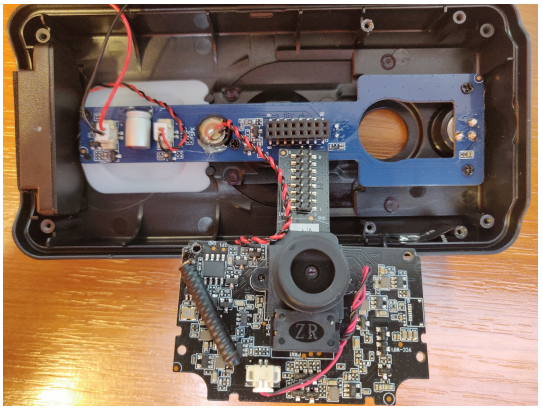


FIGURE 6.7 – Circuit imprimé Doorbell 2

➥ Test 816. *Déauthentifaction attack* :

Nous avons utilisé les outils repris dans le pack [Aircrack-ng]. Après avoir lancé l'attaque, la *video doorbell* se déconnecte du réseau, une fois l'attaque arrêtée, elle se reconnecte par elle-même. Le test est FAIL, car la *video doorbell* se déconnecte du WiFi.

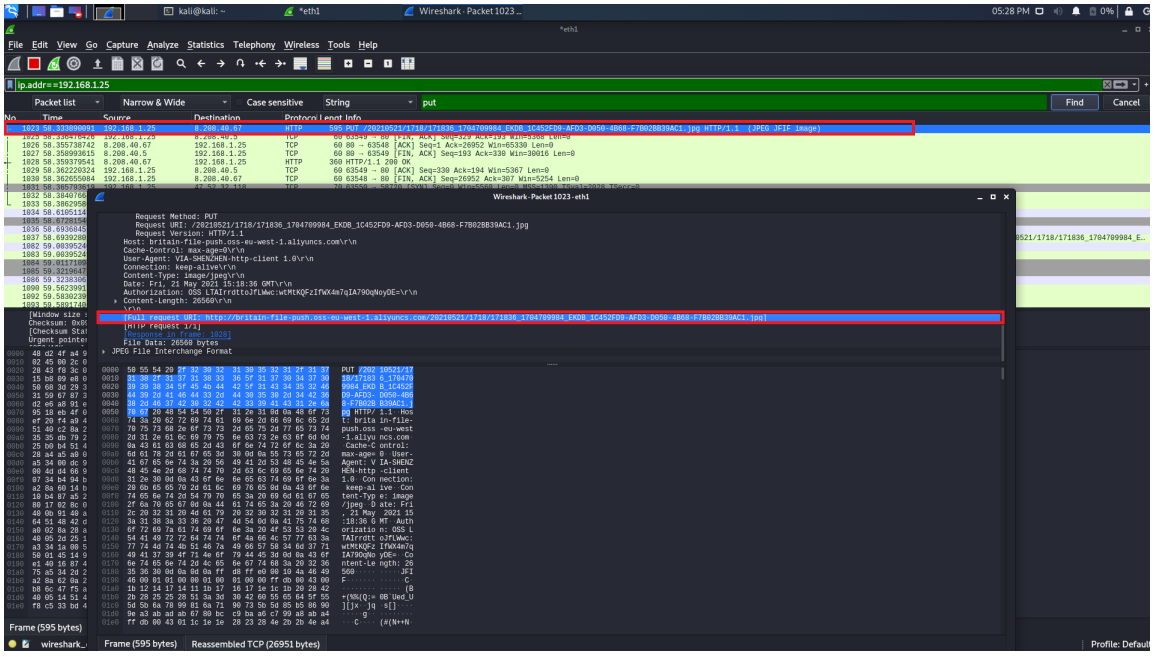
➥ Test 821. DoS attack :

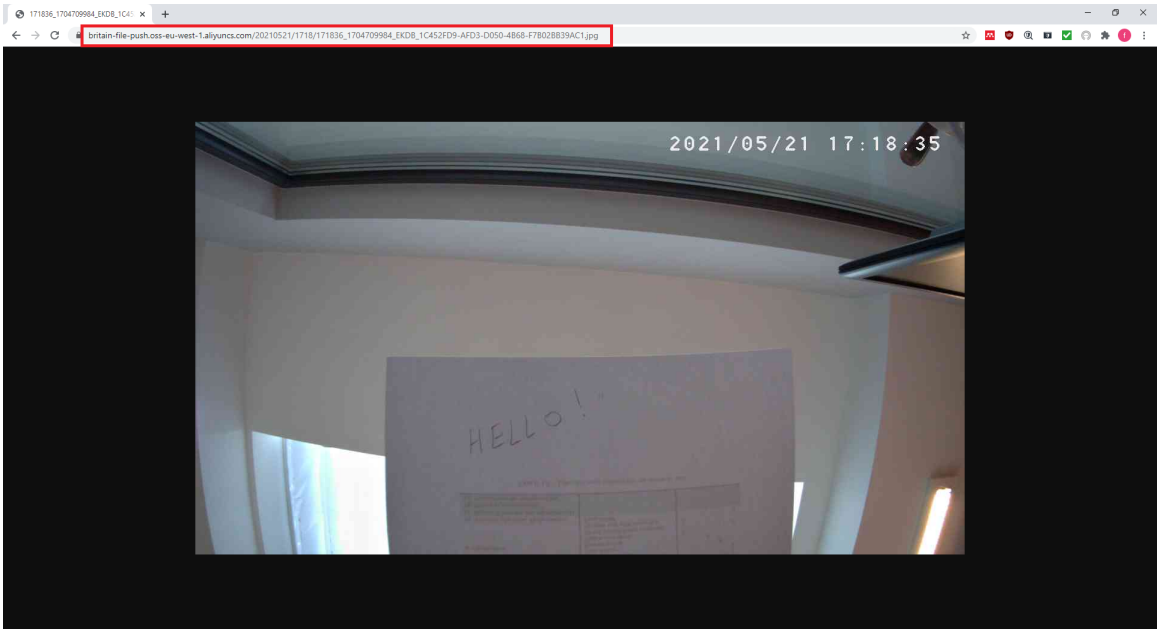
Nous avons utilisé l'outil hping3 pour réaliser ce test. Après avoir lancé l'attaque, la doorbell se met en "défaut" et il n'est plus possible de la commander depuis l'application, les pings ne fonctionnent plus non plus. Quelques minutes après avoir arrêté l'attaque, la doorbell est redevenue accessible.

```
# attaque udp flood
$ sudo hping3 -2 --flood 192.168.1.25
```

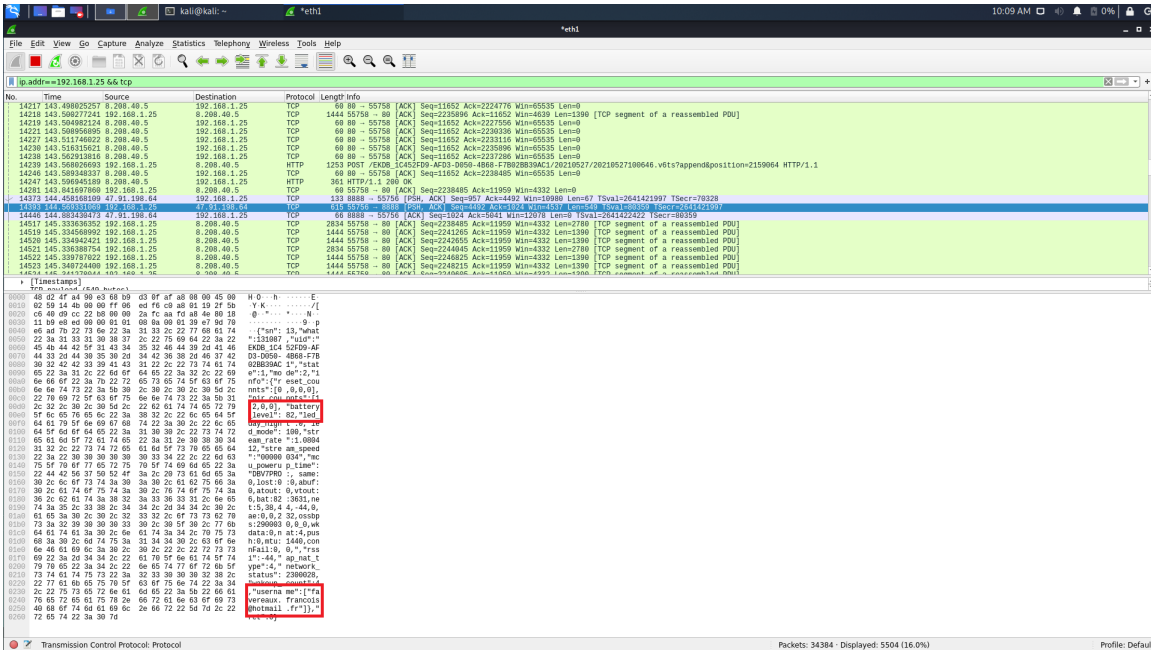
➥ Test 831. Analyse communication :

Quand la caméra détecte un mouvement, elle envoie une image au serveur. Les données envoyées ne sont pas chiffrées.



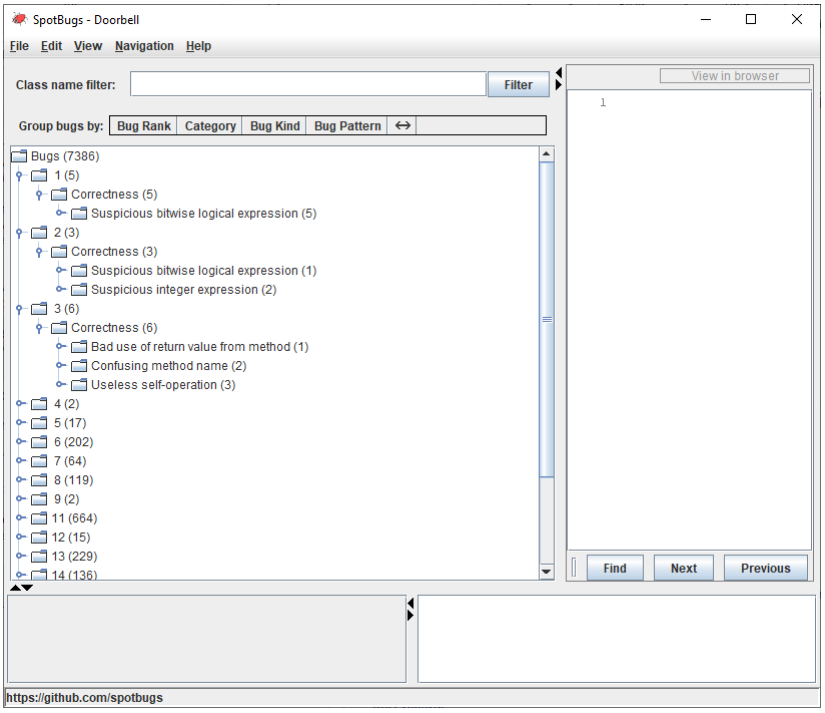


Quand on se connecte sur la caméra via l'application, on peut voir certaines informations telles que l'utilisateur, l'état de la batterie, etc.



➥ Test 913. Source code analyzer :

Nous avons utilisé l'outil Spotbugs afin d'analyser le fichier .jar précédemment obtenu. Nous avons obtenu beaucoup d'erreurs, mais elles n'ont pas l'air de mettre en péril la sécurité de la *doorbell*. Nous n'avons pas trouvé de mots de passe hardcodés ou autres.



6.3.3 Analyse printer HP Envy 7830

➥ Test 41. *Port scanning* et test 42. *Service scanning* :

Nous avons utilisé l'outil nmap afin d'effectuer ce test. Plusieurs ports TCP sont ouverts sur l'équipement. On remarque la présence d'un serveur web. Il n'y a pas d'accès SSH, FTP ou telnet.

```
kali@kali:~$ sudo nmap -sS -sV -p1-65535 192.168.1.26
Starting Nmap 7.91 ( https://nmap.org ) at 2021-05-30 11:43 CEST
Nmap scan report for 192.168.1.26
Host is up (0.034s latency).
Not shown: 65526 closed ports
PORT      STATE SERVICE    VERSION
80/tcp    open  http       nginx
443/tcp   open  ssl/http   nginx
631/tcp   open  http       nginx
3910/tcp  open  soap       gSOAP 2.7
3911/tcp  open  soap       gSOAP 2.7
5355/tcp  open  llmnr?
8080/tcp  open  http       nginx
9100/tcp  open  jetdirect?
53048/tcp open  soap       gSOAP 2.7
MAC Address: AC:E2:D3:82:00:F1 (Hewlett Packard)
```

FIGURE 6.8 – HP ENVY - port scanning TCP

```
kali@kali:~$ sudo nmap -sU 192.168.1.26
Starting Nmap 7.91 ( https://nmap.org ) at 2021-05-30 11:42 CEST
Nmap scan report for 192.168.1.26
Host is up (0.066s latency).
Not shown: 993 closed ports
PORT     STATE         SERVICE
68/udp   open|filtered dhcpc
137/udp  open|filtered netbios-ns
138/udp  open|filtered netbios-dgm
161/udp  open|filtered snmp
3702/udp open|filtered ws-discovery
5353/udp open|filtered zeroconf
5355/udp open|filtered llmnr
MAC Address: AC:E2:D3:82:00:F1 (Hewlett Packard)
```

FIGURE 6.9 – HP ENVY - port scanning UDP

➥ Test 43. *Fingerprinting* :

Nous avons utilisé l'outil nmap afin d'effectuer ce test. Cela nous a permis de récupérer les informations suivantes :
Adresse MAC = AC :E2 :D3 :82 :00 :F1
Vendeur = Hewlett Packard
Running : Linux 3.X|4.X
OS CPE : cpe :/o :linux :linux_kernel :3 cpe :/o :linux :linux_kernel :4
OS details : Linux 3.2 - 4.9

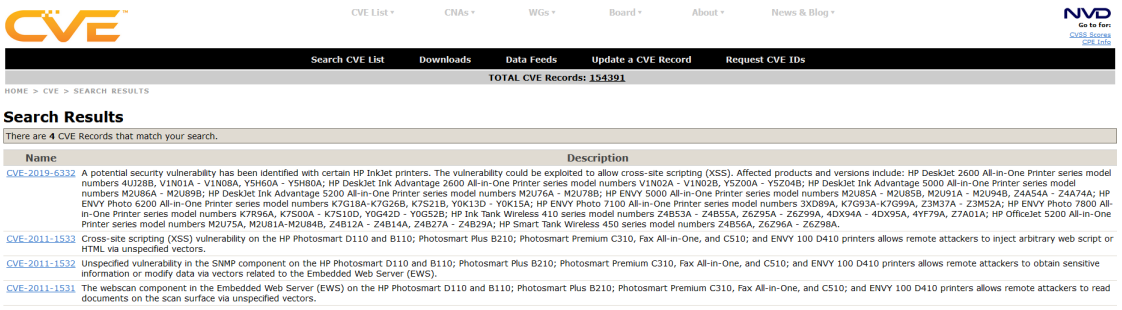
➥ Test 51. *Vulnerability scan* :

Nous avons utilisé l'outil nmap (vulscan) afin d'effectuer ce test. Ces scans testent les vulnérabilités présentes dans les bases de données CVE. Les scans ont relevé 756 vulnérabilités. Nous en présentons ci-dessous quelques unes.

```
kali@kali:~$ sudo nmap -sV --script=vulscan/vulscan.nse -oN nmapVulnENVY.txt 192.168.1.26
[sudo] password for kali:
Starting Nmap 7.91 ( https://nmap.org ) at 2021-05-30 14:23 CEST
Nmap scan report for 192.168.1.26
Host is up (0.010s latency).
Not shown: 995 closed ports
PORT    STATE SERVICE    VERSION
80/tcp  open  http       nginx
| vulscan: VulDB - https://vuldb.com:
| [155282] nginx up to 1.18.0 HTTP Request Request Smuggling privilege escalation
| [154857] Nginx Controller up to 3.3.0 Web Server Session Token Logout weak authentication
| [154326] Nginx Controller up to 3.2.x Agent Installer Script install.sh weak encryption
| [154324] Nginx Controller up to 3.2.x Postgres Database Server Man-in-the-Middle weak encryption
| [154323] Nginx Controller up to 3.1.x TLS weak encryption
| [152728] strong-nginx-controller up to 1.0.2 _nginxCmd() command injection
| [152416] Nginx Controller up to 3.1.x Controller API privilege escalation
| [148519] nginx up to 1.17.6 Error Page HTTP Request Request Smuggling privilege escalation
| [145942] nginx 0.8.40 HTTP Proxy Module Man-in-the-Middle weak authentication
| [144114] Xiaomi Mi WiFi R3G up to 2.28.22 Nginx Alias account directory traversal
| [133852] Sangfor Sundray WLAN Controller up to 3.7.4.2 Cookie Header nginx_webconsole.php Code Execution
| [132132] SoftNAS Cloud 4.2.0/4.2.1 Nginx privilege escalation
| [131858] Puppet Discovery up to 1.3.x Nginx Container weak authentication
| [130644] Nginx Unit up to 1.7.0 Router Process Request Heap-based memory corruption
| [127759] VeryNginx 0.3.3 Web Application Firewall privilege escalation
| [126525] nginx up to 1.14.0/1.15.5 ngx_http_mp4_module Loop denial of service
| [126524] nginx up to 1.14.0/1.15.5 HTTP2 CPU Exhaustion denial of service
```

➥ Test 52. Bases de données :

Nous avons trouvé 4 CVE dans les bases de données. Nous n'avons pas trouvé de scripts associés à ces CVE qui permettraient de tester cette vulnérabilité.



6.4 Evaluation des résultats obtenus

Nous présentons dans cette section l'étape 10 de la méthodologie, l'étape de *reporting*. Nous avons compilé et regroupé les résultats obtenus lors des phases d'analyse et de *pen testing* dans les tableaux 6.1, 6.2 et 6.3.

Nous avons obtenu des résultats assez semblables pour les deux premiers équipements (la caméra et la *doorbell*). Nous n'avons pas trouvé beaucoup d'informations techniques relatives à ces deux équipements, cela est probablement dû au fait qu'ils sont principalement destinés à un usage domestique et que peu d'utilisateurs ont besoin d'avoir accès au firmware, aux datasheets techniques, etc. Il se peut également que le fabricant ne souhaite tout simplement pas rendre public ce genre d'informations. Les ports TCP et UDP étaient fermés ou ouverts-filtrés, il n'y avait donc pas beaucoup d'informations à obtenir au niveau des ports et des services. La caméra n'a pas montré beaucoup de faiblesses, parmi tous les tests effectués, seuls quelques uns ont échoué. La *doorbell* a, quant à elle, montré plus de vulnérabilités : peu ou pas de chiffrement dans les communications, elle s'est montrée sensible aux attaques DoS et à la déauthentification, etc. Elle est donc moins "robuste" que la caméra. Dans le rapport on peut voir que certains tests n'ont pas été effectués (en orange dans le tableau) car nous avons manqué de temps, de moyens et de connaissances. Malgré cela, les résultats obtenus permettent de présenter des informations précieuses relatives à sécurité de l'équipement. Au premier coup d'oeil, on peut avoir un aperçu assez précis du niveau de sécurité de l'équipement.

Comme les deux premiers équipements testés n'ont pas permis de mettre en avant l'importance de la phase d'analyse, que la plupart des ports réseaux étaient fermés et qu'il n'y avait pas de *webserver*, nous avons décidé d'appliquer la méthodologie sur un troisième équipement, l'imprimante HP ENVY 7830. Nous avons trouvé beaucoup d'informations lors de la phase d'analyse. Comme attendu, nous avons trouvé de la documentation sur le site du fournisseur et les étapes 41, 42 et 43 ont également permis de récupérer des données intéressantes, les ports ouverts, les services actifs, etc. Lors des étapes 51 et 52 nous avons trouvé un grand nombre de vulnérabilités potentielles. Dans les bases de données nous en avons trouvé 4 et à l'aide des scanners, nous en avons trouvé un peu moins de 800 ! Nous n'avons évidemment pas testé chacune des vulnérabilités à l'étape 61 (exploitation des vulnérabilités). L'imprimante ne

réussit donc pas l'évaluation de sécurité, de trop nombreuses vulnérabilités sont présentes. Par manque de temps, nous nous sommes arrêtés à l'étape 6 de la méthodologie.

Caméra Imilab Home Xiaomi				
Test	Intitulé du test	Statut du test	Commentaire	Résultat
1.	Information gathering	Effectué	Très peu d'informations trouvées. Nous avons récupéré un *user manual* ne contenant pas beaucoup d'informations techniques.	PASS
2.	Architecture review	Effectué	Très peu d'informations trouvées. Après analyse du fonctionnement de la caméra et de l'application smartphone, il s'avère que la caméra communique directement avec les serveurs Xiaomi à travers le modem WiFi. Il n'est pas nécessaire d'avoir un bridge ou autre. La caméra est accessible depuis n'importe quelle connexion internet.	PASS
31.	Updates	Effectué	Pas de mises à jour en attente.	PASS
32.	Monitoring	Effectué	Pas de système de monitoring permettant de voir les différentes actions, mises sous tension etc.	FAIL
33.	Sécurité physique	Effectué	Pas d'accès physique à la caméra.	PASS
34.	Paramètres et configuration	Effectué	Il faut utiliser un smartphone avec lequel on a couplé la caméra pour pouvoir visualiser les paramètres et données. Une fois le smartphone déverrouillé, un mot de passe (4 chiffres) peut être configuré afin de modifier les paramètres de la caméra.	PASS
35.	Decommissioning	Non applicable	Pas d'autres équipements présents.	PASS
41.	Port scanning	Effectué	Les 65535 ports TCP sont fermés. Certains ports UDP sont au statut "ouvert-filtré".	PASS
42.	Services scanning	Effectué	Il n'y a pas de ports ouverts, donc pas de services à scanner.	PASS
43.	Fingerprinting	Effectué	Adresse MAC :78 :8b :2a :a3 :4a :be Vendeur : Zhen Shi Information Technology (Shanghai) OS : /	PASS
51.	Scan vulnérabilités	Effectué	Pas de vulnérabilités détectées. Outils utilisés : nmap vulscan.	PASS
52.	Base de données	Effectué	Pas de vulnérabilités répertoriées dans les bases de données.	PASS
61.	Exploitation vulnérabilités	Non applicable	Pas de vulnérabilités détectées aux étapes 51 et 52.	PASS
711.	Information gathering	Effectué	Nous n'avons trouvé aucune information intéressante pour ce test.	PASS
712.	Analyse du firmware	Non effectué	Le firmware n'a pas pu être récupéré, ni via l'étape 713, ni via internet ou un autre moyen.	TBT
713.	Extraction du firmware	Effectué	Nous pensons qu'il est possible de récupérer le firmware via les pins RX0 et TX0 (voir 6.4). Nous n'avons pas réussi à souder des fils sur ces connexions. Nous ne possédons pas d'autre matériel pour effectuer le test.	TBT
721.	Sleep deprivation attack	Non applicable	La caméra ne possède pas de batteries	PASS
731.	Jamming attack	Non effectué	Nous ne possédons pas de *jamming device*.	TBT

732.	Composants utilisés	Non effectué	Nos connaissances en électronique ne permettent pas de savoir si les composants utilisés sont obsolètes ou non.	TBT
733.	Side channel attack	Non effectué	Nous n'avons pas les compétences techniques nécessaires pour réaliser le test.	TBT
811.	Brute force FTP	Non applicable	Pas de service FTP sur la caméra	PASS
812.	Brute force SSH	Non applicable	Pas de service SSH sur la caméra	PASS
813.	Brute force Telnet	Non applicable	Pas de service telnet sur la caméra	PASS
814.	Deauthentication attack	Effecuté	L'équipement se déconnecte du réseau pendant l'attaque	FAIL
821.	DoS attack	Effectué	Tests effectués avec LOIC et hping3 (UDP, TCP, ICMP flood). La caméra réagit plus lentement, mais elle est toujours opérationnelle.	PASS
822.	Replay attack	Effectué	Nous avons enregistré grâce à *wireshark* des paquets dans un fichier .pcap. Nous avons essayé de les renvoyer à l'aide de *tcpreplay* et *scapy*, mais nous n'y sommes pas arrivés. Nous avons eu le même message d'erreur à chaque fois : "Errno 90".	TBT
823.	Communication tampering	Non effectué	Manque de temps pour effectuer le test	TBT
824.	Communication IP externe	Effectué	La caméra communique avec plusieurs adresses IP (nous ne savons pas dire si ces adresses sont malicieuses ou non, nous mettons le test en FAIL pour attirer l'attention). Ces adresses sont probablement les services dédiés à la caméra et situés dans le cloud.	FAIL
831.	Analyse communication	Effectué	Nous n'avons pas vu d'informations sensibles via *wireshark*.	PASS
832.	Updates chiffrées	Effectué	On utilise TLS 1.2 pour mettre à jour la caméra. Les updates sont chiffrées.	PASS
833.	Downgrading attack	Non applicable	Pas d'utilisation de HTTPS.	PASS
834.	Key reinstallation attack	Non effectué	Manque de temps pour effectuer le test	TBT
835.	MITM	Non effectué	Manque de temps pour effectuer le test	TBT
836.	Cryptanalysis Attacks	Non effectué	Nous n'avons pas les compétences techniques nécessaires pour réaliser le test.	TBT
911.	Synchronisation NTP	Effectué	Nous n'avons pas vu d'échanges de paquets NTP. Ni au démarrage de la caméra, ni après un certain temps d'utilisation. L'heure est disponible sur les images vidéo, elle provient probablement du cloud.	FAIL
912.	Désassemblage application	Effectué	Nous avons désassemblé la version 6.5.700 de l'application *Xiaomi Home*. Nous avons obtenu un fichier .jar.	PASS
913.	Source code analyzer	Effectué	Nous avons analysé le fichier .jar de l'application à l'aide de Spotbugs. Beaucoup d'erreurs sont présentes mais elles n'ont pas l'air de mettre à mal la sécurité de l'équipement.	PASS
914.	Bibliothèques logicielles	Non effectué	Manque de temps pour effectuer le test	TBT
921.	Scan vulnérabilités	Non applicable	Pas de serveur web sur la caméra	PASS
922.	Firewall	Non applicable	Pas de serveur web sur la caméra	PASS

923.	Certificat SSL	Non applicable	Pas de serveur web sur la caméra	PASS
924.	Vérification identifiants	Non applicable	Pas de serveur web sur la caméra	PASS
925.	Bypass basic HTTP auth.	Non applicable	Pas de serveur web sur la caméra	PASS
926.	Brute force directories	Non applicable	Pas de serveur web sur la caméra	PASS
927.	Injection SQL	Non applicable	Pas de serveur web sur la caméra	PASS
928.	Injection XSS	Non applicable	Pas de serveur web sur la caméra	PASS
929.	HMTL analysis	Non applicable	Pas de serveur web sur la caméra	PASS
930.	Specific attack	Non applicable	Pas de serveur web sur la caméra	PASS

TABLE 6.1 – Résultats de l'évaluation de sécurité de la Caméra Imilab

Doorbell V7				
Test	Intitulé du test	Statut du test	Commentaire	Résultat
1.	Information gathering	Effectué	Très peu d'informations trouvées. Nous avons récupéré un *user manual* ne contenant pas beaucoup d'informations techniques.	PASS
2.	Architecture review	Effectué	Très peu d'informations trouvées. Après analyse du fonctionnement de la caméra et de l'application smartphone, il s'avère que la *doorbell* communique directement avec les serveurs EKEN à travers le modem WiFi. Il n'est pas nécessaire d'avoir un bridge ou autre. La *doorbell* est accessible depuis n'importe quelle connexion internet.	PASS
31.	Updates	Effectué	Pas de mise à jour en attente.	PASS
32.	Monitoring	Effectué	Pas de système de monitoring permettant de voir les différentes actions, mises sous tension etc.	FAIL
33.	Sécurité physique	Effectué	La *doorbell* est placée en extérieur, elle est accessible de tous.	FAIL
34.	Paramètres et configuration	Effectué	Il faut utiliser un smartphone avec lequel on a couplé la doorbell pour pouvoir visualiser les paramètres et données. Il n'y a pas de mot de passe pour accéder à la doorbell une fois que le smartphone a été déverrouillé.	PASS
35.	Decommissioning	Non applicable	Pas d'autres équipements présents.	PASS
41.	Port scanning	Effectué	Les 65535 ports TCP sont fermés. Certains ports UDP sont au statut "ouvert-filtré".	PASS
42.	Services scanning	Effectué	Il n'y a pas de ports ouverts, donc pas de services à scanner	PASS
43.	Fingerprinting	Effectué	Adresse MAC = 68 :b9 :d3 :0f :af :a8. Vendeur = Shenzhen Trolink Technology. OS = / .	PASS
51.	Scan vulnérabilités	Effectué	Pas de vulnérabilités détectées. Outils utilisés : nmap vulscan.	PASS
52.	Base de données	Effectué	Pas de vulnérabilités répertoriées dans les bases de données	PASS
61.	Exploitation vulnérabilités	Non applicable	Pas de vulnérabilités détectées aux étapes 51 et 52	PASS

711.	Information gathering	Effectué	Nous n'avons trouvé aucune information intéressante pour ce test.	PASS
712.	Analyse du firmware	Non effectué	Le firmware n'a pas pu être récupéré, ni via l'étape 713, ni via internet ou un autre moyen.	TBT
713.	Extraction du firmware	Effectué	Nous n'avons pas pu récupérer le firmware. Nous n'avons pas trouvé de prises UART ou JTAG sur l'équipement.	FAIL
721.	Sleep deprivation attack	Non effectué	Manque de temps pour effectuer le test	TBT
731.	Jamming attack	Non effectué	Nous ne possédons pas de *jamming device*.	TBT
732.	Composants utilisés	Non effectué	Nos connaissances en électronique ne permettent pas de savoir si les composants utilisés sont obsolètes ou non.	TBT
733.	Side channel attack	Non effectué	Nous n'avons pas les compétences techniques nécessaires pour réaliser le test.	TBT
811.	Brute force FTP	Non applicable	Pas de service FTP sur la caméra	PASS
812.	Brute force SSH	Non applicable	Pas de service SSH sur la caméra	PASS
813.	Brute force Telnet	Non applicable	Pas de service telnet sur la caméra	PASS
814.	Deauthentication attack	Effecuté	L'équipement se déconnecte du réseau pendant l'attaque	FAIL
821.	DoS attack	Effectué	Nous avons utilisé l'attaque UDP flood. La *doorbell* ne répond plus aux pings, on ne sait plus avoir accès à la doorbell via l'application.	FAIL
822.	Replay attack	Non effectué	Manque de temps pour effectuer le test	TBT
823.	Communication tampering	Non effectué	Manque de temps pour effectuer le test	TBT
824.	Communication IP externe	Effectué	La *doorbell* communique avec plusieurs adresses IP (nous ne savons pas dire si ces adresses sont malicieuses ou non, nous mettons le test en FAIL pour attirer l'attention). Ces adresses sont probablement les services dédiés à la *doorbell* et situés dans le cloud.	FAIL
831.	Analyse communication	Effectué	Beaucoup d'informations sont visibles via wireshark (nom utilisateur, statut batterie, les images envoyées vers le cloud, etc.). L'équipement utilise le protocole HTTP pour échanger des informations.	FAIL
832.	Updates chiffrées	Non effectué	Nous n'avons jamais dû faire de mises à jour durant la phase de test. Il n'est donc pas possible de vérifier le chiffrement.	TBT
833.	Downgrading attack	Non applicable	Pas d'utilisation de HTTPS.	PASS
834.	Key reinstallation attack	Non effectué	Manque de temps pour effectuer le test	TBT
835.	MITM	Non effectué	Manque de temps pour effectuer le test	TBT
836.	Cryptanalysis Attacks	Non effectué	Nous n'avons pas les compétences techniques nécessaires pour réaliser le test.	TBT
911.	Synchronisation NTP	Effectué	Nous n'avons pas vu d'échanges de paquets NTP. Ni au démarrage de la caméra, ni après un certain temps d'utilisation. L'heure est disponible sur les images vidéo, elle provient probablement du cloud.	FAIL

912.	Désassemblage application	Effectué	Nous avons désassemblé la version 2.2.2 de l'application *Aiwit*. Nous avons obtenu un fichier .jar.	PASS
913.	Source code analyzer	Effectué	Nous avons analysé le fichier .jar de l'application à l'aide de Spotbugs. Beaucoup d'erreurs sont présentes mais elles n'ont pas l'air de mettre à mal la sécurité de l'équipement.	PASS
914.	Bibliothèques logicielles	Non effectué	Manque de temps pour effectuer le test	TBT
921.	Scan vulnérabilités	Non applicable	Pas de serveur web sur la caméra	PASS
922.	Firewall	Non applicable	Pas de serveur web sur la caméra	PASS
923.	Certificat SSL	Non applicable	Pas de serveur web sur la caméra	PASS
924.	Vérification identifiants	Non applicable	Pas de serveur web sur la caméra	PASS
925.	Bypass basic HTTP auth.	Non applicable	Pas de serveur web sur la caméra	PASS
926.	Brute force directories	Non applicable	Pas de serveur web sur la caméra	PASS
927.	Injection SQL	Non applicable	Pas de serveur web sur la caméra	PASS
928.	Injection XSS	Non applicable	Pas de serveur web sur la caméra	PASS
929.	HMTL analysis	Non applicable	Pas de serveur web sur la caméra	PASS
930.	Specific attack	Non applicable	Pas de serveur web sur la caméra	PASS

TABLE 6.2 – Résultats de l'évaluation de sécurité de la Doorbell V7

Imprimante HP ENVY 7830				
Test	Intitulé du test	Statut du test	Commentaire	Résultat
1.	Information gathering	Effectué	Nous avons trouvé plusieurs informations intéressantes sur le site du fabricant (HP).	PASS
2.	Architecture review	Effectué	Pas d'architecture spécifique. L'imprimante se connecte directement au réseau.	PASS
31.	Updates	Effectué	Pas de mises à jour en attente.	PASS
32.	Monitoring	Effectué	L'imprimante possède un journal d'évènements (accessible via l'adresse ip et un browser). Cependant, ce service est au statut "erreur système".	FAIL
33.	Sécurité physique	Effectué	Pas d'accès physique à l'imprimante.	PASS
34.	Paramètres et configuration	Effectué	Pas de restriction de modification des paramètres. On peut modifier l'adresse IP, par exemple.	FAIL
35.	Decommissioning	Non applicable	Pas d'autres équipements présents.	PASS
41.	Port scanning	Effectué	Plusieurs ports TCP et UDP ouverts. On a détecté un serveur web.	PASS
42.	Services scanning	Effectué	Pour chaque ports TCP et UDP ouverts, les services ont été récupérés.	PASS
43.	Fingerprinting	Effectué	Les quatre informations reprises dans la méthodologie ont été trouvées (OS, vendeur, MAC adresse, adresse ip).	PASS
51.	Scan vulnérabilités	Effectué	Nous avons trouvé 756 vulnérabilités.	FAIL
52.	Base de données	Effectué	Nous avons trouvé 4 CVE.	FAIL
61.	Exploitation vulnérabilités	Non effectué	De trop nombreuses vulnérabilités sont présentes, il n'est pas possible de tout tester.	TBT

711.	Information gathering	Non effectué	Manque de temps pour effectuer le test.	TBT
712.	Analyse du firmware	Non effectué	Manque de temps pour effectuer le test.	TBT
713.	Extraction du firmware	Non effectué	Manque de temps pour effectuer le test.	TBT
721.	Sleep deprivation attack	Non effectué	Manque de temps pour effectuer le test.	TBT
731.	Jamming attack	Non effectué	Manque de temps pour effectuer le test.	TBT
732.	Composants utilisés	Non effectué	Manque de temps pour effectuer le test.	TBT
733.	Side channel attack	Non effectué	Manque de temps pour effectuer le test.	TBT
811.	Brute force FTP	Non effectué	Manque de temps pour effectuer le test.	TBT
812.	Brute force SSH	Non effectué	Manque de temps pour effectuer le test.	TBT
813.	Brute force Telnet	Non effectué	Manque de temps pour effectuer le test.	TBT
814.	Deauthentication attack	Non effectué	Manque de temps pour effectuer le test.	TBT
821.	DoS attack	Non effectué	Manque de temps pour effectuer le test.	TBT
822.	Replay attack	Non effectué	Manque de temps pour effectuer le test.	TBT
823.	Communication tampering	Non effectué	Manque de temps pour effectuer le test.	TBT
824.	Communication IP externe	Non effectué	Manque de temps pour effectuer le test.	TBT
831.	Analyse communication	Non effectué	Manque de temps pour effectuer le test.	TBT
832.	Updates chiffrées	Non effectué	Manque de temps pour effectuer le test.	TBT
833.	Downgrading attack	Non effectué	Manque de temps pour effectuer le test.	TBT
834.	Key reinstallation attack	Non effectué	Manque de temps pour effectuer le test.	TBT
835.	MITM	Non effectué	Manque de temps pour effectuer le test.	TBT
836.	Cryptanalysis Attacks	Non effectué	Manque de temps pour effectuer le test.	TBT
911.	Synchronisation NTP	Non effectué	Manque de temps pour effectuer le test.	TBT
912.	Désassemblage application	Non effectué	Manque de temps pour effectuer le test.	TBT
913.	Source code analyzer	Non effectué	Manque de temps pour effectuer le test.	TBT
914.	Bibliothèques logicielles	Non effectué	Manque de temps pour effectuer le test.	TBT
921.	Scan vulnérabilités	Non effectué	Manque de temps pour effectuer le test.	TBT
922.	Firewall	Non effectué	Manque de temps pour effectuer le test.	TBT
923.	Certificat SSL	Non effectué	Manque de temps pour effectuer le test.	TBT
924.	Vérification identifiants	Non effectué	Manque de temps pour effectuer le test.	TBT
925.	Bypass basic HTTP auth.	Non effectué	Manque de temps pour effectuer le test.	TBT
926.	Brute force directories	Non effectué	Manque de temps pour effectuer le test.	TBT
927.	Injection SQL	Non effectué	Manque de temps pour effectuer le test.	TBT
928.	Injection XSS	Non effectué	Manque de temps pour effectuer le test.	TBT
929.	HMTL analysis	Non effectué	Manque de temps pour effectuer le test.	TBT

930.	Specific attack	Non effectué	Manque de temps pour effectuer le test.	TBT

TABLE 6.3 – Résultats de l'évaluation de sécurité de l'imprimante HP ENVY 7830

6.5 Evaluation de la méthodologie

L'utilisation et la mise en pratique de la méthodologie ont permis de mettre en avant plusieurs choses. Le premier constat positif que nous avons observé concerne l'utilisation de la méthodologie, elle est assez simple, logique et intuitive. En effet, les objectifs de chaque test sont clairement définis et les tests sont exécutés dans un ordre logique et intuitif. Un deuxième constat positif concerne les résultats obtenus. Même si tous les tests n'ont pas été exécutés, on voit clairement les vulnérabilités qui sont présentes (en rouge dans le tableau), celles qui ne le sont pas (en vert dans le tableau) et celles qui le sont potentiellement (en orange dans le tableau).

Cependant, même si la méthodologie est facile à utiliser, nous avons rencontré certaines difficultés d'un point de vue connaissances théoriques et techniques. Certains tests sont assez complexes et ne se réalisent pas facilement. A moins d'être un expert, la méthodologie seule ne suffit pas pour pouvoir réaliser tous les tests sans aide extérieure. Nous avons dû chercher après des outils sur internet, lire leur documentation et se familiariser avec. De plus, entreprendre une évaluation complète d'un équipement prend beaucoup de temps. Soit parce que le test en lui-même prend du temps (un scan UDP des 65535 ports par exemple) soit parce qu'il faut chercher des informations supplémentaires pour réaliser le test.

Chapitre 7

Conclusion

7.1 Retour sur la problématisation

L'enjeu de ce mémoire était de concevoir et de proposer une méthodologie d'évaluation de la sécurité dédiée aux objets connectés. A la lecture de ce travail, nous pouvons facilement conclure que cet objectif est rempli. Néanmoins, même s'il est aisé de voir que l'objectif a été atteint, il est beaucoup plus compliqué de savoir si la solution proposée permet de réellement satisfaire la problématique de départ.

Suite à la problématique de départ, le premier critère que la méthodologie doit remplir est le côté générique de la méthode. Cela revient à se poser la question suivante : est-ce que la méthodologie s'applique de la même manière peu importe le type d'équipement ? Le deuxième critère que doit rencontrer la méthodologie concerne les différents moyens de communication et les différents types d'IoT. Donc, est-ce que la méthodologie fonctionne sur tous les équipements, et ce, peu importe le moyen de communication utilisé ? A ces deux questions, nous sommes tentés de répondre oui. Lors de la phase d'évaluation de la méthodologie, nous sommes parvenus à l'appliquer sur trois équipements distincts sans rencontrer de difficultés et nous avons mis en évidence plusieurs problèmes de sécurité. La méthodologie permet donc de détecter des problèmes de sécurité et peut être appliquée à différents IoT.

Même si, comme nous venons de le voir, le travail proposé semble répondre à la problématique de départ, nous tenons toutefois à mettre en évidence certains manquements ou problèmes que nous avons constatés pendant l'élaboration et l'utilisation de la méthodologie. Premièrement, même si la méthodologie a permis de détecter des vulnérabilités lors de la phase d'évaluation, nous ne sommes pas certains que la liste des tests repris soit assez exhaustive. Certaines attaques et vulnérabilités étant spécifiques à certains protocoles de communication, il est tout à fait possible que quelques tests nous aient échappés. Deuxièmement, la méthodologie a été testée sur seulement trois équipements IoT (qui utilisent tous les trois le WiFi comme protocole de communication). Le nombre d'équipements testés et leur "homogénéité" ne permet donc pas d'attester complètement que la méthodologie peut s'adapter à n'importe quel type d'IoT et à tous les protocoles de communication. Troisièmement, nous tenons à rappeler que nous n'avions pas spécialement de connaissances en sécurité informatique. Ce travail a été le fruit de recherches minutieuses dans la littérature, d'analyses de travaux pratiques, etc. Certaines erreurs peuvent

s'être glissées dans le travail. Enfin, une chose essentielle manque à ce travail, l'implication des *stakeholders*. C'est de loin le point le plus important des quatre et le plus gros manquement de ce travail. Nous n'avons impliqué aucun *stakeholder* lors de la conception de méthodologie. Nous avons soumis notre méthodologie à deux sociétés actives dans le domaine afin qu'elles commentent ce travail, mais nous n'avons pas reçu de retour. Ce genre de méthodologie doit évidemment passer entre les mains de gens de terrain, de fabricants, d'utilisateurs, d'experts en sécurité etc. C'est de cette façon qu'on pourra modifier, adapter, corriger et améliorer la méthodologie.

Pour conclure, nous pouvons légitimement nous demander si cette méthodologie innovante est réellement utile et efficace. Nous avons vu que les bancs de test étaient incomplets, qu'ils ne permettaient de tester qu'une partie des attaques et vulnérabilités, qu'ils étaient difficilement reproductibles et que les rapports de résultats ne mentionnaient pas les vulnérabilités non testées. Nous avons également vu que les frameworks existants d'évaluation de la sécurité s'adaptaient mal aux équipements IoT. Lors de l'utilisation de notre méthodologie nous nous sommes rendus compte que, bien qu'utile et facile à suivre, bien que le rapport d'évaluation soit complet et exhaustif, la méthodologie seule ne suffisait pas. Les tests techniques n'étaient pas assez détaillés et certains outils n'étaient pas renseignés. Il ressort donc qu'aucune solution existante ne fonctionne parfaitement. Nous pensons que les recherches futures devraient s'orienter vers une solution plus pratique, à la frontière entre les bancs de test et notre méthodologie.

7.2 Améliorations et recherches futures

Cette dernière section s'adresse plus particulièrement aux personnes qui voudraient poursuivre les recherches que nous avons commencées. Nous avons vu, dans la deuxième partie de ce travail (II - partie recherches et développement), que nous proposons ici la première méthodologie de test de la sécurité IoT. Comme pour toutes les premières versions d'un projet, on dénombre plusieurs points d'amélioration.

Comme nous l'avons déjà suggéré à la section précédente, une première amélioration possible concerne les exemples pratiques et outils. Nous avons proposé une méthodologie assez courte et assez simple afin de ne pas effrayer les utilisateurs futurs (à titre de comparaison, le framework de l'ISSAF [62] dédié à la sécurité informatique totalise 845 pages). On pourrait cependant apporter de plus amples informations pour chaque test avec des exemples pratiques, des printscreens, des résultats obtenus avec certains outils, etc. On pourrait également proposer des outils qui permettraient de tester les protocoles de communication peu ou pas abordés dans ce travail tels que le Zigbee, le bluetooth, etc.

Une deuxième amélioration possible concerne l'étape 10 de la méthodologie - l'étape de *reporting*. Presque tous les frameworks d'évaluation de la sécurité informatique s'accordent sur ce point, la phase de reporting doit comprendre un "score de sécurité" et posséder des métriques. Même si intuitivement on comprend l'utilité de cette fonctionnalité (pouvoir comparer facilement deux analyses de sécurité), la mise en place de cette métrique de résultat est compliquée. Preuve en est, sur les cinq méthodologies de sécurité informatique étudiées dans ce travail, seul le framework OSSTM en propose une ! Nous laissons cette tâche pour de futures recherches.

La troisième et dernière amélioration proposée concerne la structure de la méthodologie et

la liste des tests. Nous avons choisi de diviser la phase de *pen testing* en trois parties afin d'améliorer la lisibilité et l'utilisation de la méthodologie. Nous ne sommes cependant pas certains d'avoir choisi la meilleure classification et nous ne sommes pas certains d'avoir rangé tous les tests dans les bonnes catégories. Nous avons par exemple classé l'attaque *DoS* dans la partie couche réseau, mais nous avons déjà observé dans la littérature que certains chercheurs la classent parfois dans la couche applicative. Une meilleure structure et une classification plus judicieuse pourraient donc être obtenues si un chercheur venait reprendre le travail sous un regard nouveau.

Comme nous pouvons le constater, plusieurs améliorations intéressantes peuvent être apportées à notre travail. Cette méthodologie n'est qu'une première ébauche et nous espérons que d'autres poursuivront ce que nous avons commencé et que des modèles, frameworks et méthodologies différents verront le jour dans le domaine de l'IoT.

Troisième partie

Annexes

Annexe A

Liste des outils

La table A.1 a pour but d'inventorier les outils mentionnés et utilisés lors de ce travail. Une brève description de l'outil sera proposée ainsi qu'un lien permettant d'avoir accès à plus d'informations.

N°	Outil	Description	Documentation
1	**Aireplay-ng**	Cet outil propose de nombreuses attaques permettant de désauthentifier les clients sans fil dans le but de capturer des informations.	https://tools.kali.org/wireless-attacks/aireplay-ng
2	**Amap**	Permet de trouver des services et applications sur un hôte (même si on utilise un port inhabituel).	https://gitlab.com/kalilinux/packages/amap
3	**Armitage tools**	TBD	https://tools.kali.org/exploitation-tools/armitage
4	**Bettercap**	outil multi-usage permettant entre autres d'effectuer : attaques de désauthentification, network sniffer, credentials harvesting, ...	https://www.bettercap.org/intro/
5	**Binwalk**	Outil conçu pour identifier les fichiers et le code intégrés dans les images firmware.	https://tools.kali.org/forensics/binwalk
6	**Bluescan**	c'est un scanner bluetooth, il permet de récupérer une série d'informations telles que les vulnérabilités, les services SDP, etc.	https://github.com/f0-000/bluescan
7	**Dex2jar**	Cet outil permet de transposer l'archive d'une application android en du code java.	https://tools.kali.org/reverse-engineering/dex2jar
8	**Dirbuster**	DirBuster est un outil java multi-thread conçu pour forcer les noms de répertoires et de fichiers sur les serveurs web/applications.	https://tools.kali.org/web-applications/dirbuster
9	**Ettercap**	permet d'effectuer des attaques MITM, sniffer des connexions live, etc.	https://www.ettercap-project.org/
1	**Grabber**	scanner d'application web	https://tools.kali.org/web-applications/grabber
10	**Hydra**	permet de récupérer des mots de passe et ce, sur de nombreux protocoles (CVS, FTP, HTTP(S)-FORM-GET, HTTP(S)-FORM-POST, etc.)	https://tools.kali.org/password-attacks/hydra
11	**Kismet**	C'est un détecteur de réseaux et de périphériques sans fil, un sniffer, un outil de surveillance, etc.	https://www.kismetwireless.net/ https://tools.kali.org/wireless-attacks/kismet
12	**Masscan**	TBD	https://github.com/robertdavidgraham/masscan
13	**Metasploit**	TBD	https://www.metasploit.com/ https://docs.rapid7.com/metasploit/
14	**Netdiscover**	TBD	https://github.com/netdiscover-scanner/netdiscover

15	**Nikto**	C'est un scanner de serveur web	https://tools.kali.org/information-gathering/nikto
16	**P0f**	TBD	https://tools.kali.org/information-gathering/p0f
17	**PMD**	Code source analyzer	https://pmd.github.io/
18	**Scapy**	TBD	https://scapy.net/ https://github.com/secdev/scapy/
19	**Script python 1**	Script pyhton permettant de réaliser une attaque DoS sur un équipement bluetooth	https://github.com/crypt0b0y/BLUETOOTH-DOS-ATTACK-SCRIPT
20	**Script python 2**	Script pyhton permettant de faire crasher le service bluetooth	https://github.com/ojasookert/CVE-2017-0781
21	**Script python 3**	Script pyhton permettant d'effectuer une attaque KRACK.	https://github.com/vanhoefm/krackattacks-scripts
22	**Skipfish**	scanner d'application web	https://tools.kali.org/web-applications/skipfish
23	**SonarQube**	- Code source analyzer	https://www.sonarqube.org/
24	**Spooftooph**	- Permet d'effectuer du cloning ou du spoofing pour des équipements bluetooth.	https://tools.kali.org/wireless-attacks/spooftooph
25	**Spotbugs**	- Code source analyzer	https://spotbugs.github.io/
26	**SQLmap**	C'est un outil de test de pénétration open source qui automatise le processus de détection et d'exploitation des injections SQL.	https://sqlmap.org/
27	**SSLscan**	Cet outil interroge les services SSL/TLS afin de s'assurer que les certificats sont fiables.	https://github.com/rbsec/sslscan
28	**SSLStrip**	Cet outil redirige le trafic https vers http.	https://tools.kali.org/information-gathering/sslstrip
29	**Unicornscan**	TBD	https://tools.kali.org/information-gathering/unicornscan
30	**Wafw00f**	Cet outil permet de savoir si un équipement utilise un web serveur.	https://github.com/EnableSecurity/wafw00f
31	**WiFi-Jammer**	C'est un outil python permettant de brouiller tout le trafic de clients ou de réseaux sélectionnés à portée et de les déconnecter de leur AP.	https://github.com/zflemingg1/WiFi-Jammer
32	**Wireshark**	Wireshark est un analyseur de paquets réseau.	https://www.wireshark.org/docs/wsug_html/#ChIntroWhatIs
33	**XSSer**	Cet outil permet de détecter, d'exploiter et de signaler les vulnérabilités XSS dans les applications Web.	https://tools.kali.org/web-applications/xsser

TABLE A.1 – Liste des outils utilisés

Annexe B

Wireless scanning

Le *wireless scanning* consiste à détecter les équipements qui communiquent via des réseaux non filaires. Parmi ces connexions non filaires, on retrouve des protocoles de communication tels que le zigbee, le WiFi, le bluetooth, le rfid, etc. Afin de détecter les équipements présents sur le réseau sans fil et afin d'analyser le trafic du réseau, il faut utiliser un appareil sur lequel est installé et configuré un logiciel d'analyse (par exemple, un ordinateur portable, certains dispositifs spécifiques portables, etc.). Il est également parfois intéressant d'utiliser une antenne afin de permettre une plus grande couverture du réseau. Certains analyseurs de réseau permettent même de cartographier et de tracer l'emplacement physique des appareils. Dans certains cas, la cartographie est basée sur le système de positionnement global (GPS). On peut procéder à deux types différents de scanning : le scanning passif et le scanning actif.

Passive scanning : on capture les données qui transitent sur le réseau. On récupère alors des informations telles que le SSID, l'adresse MAC, le nom de l'équipement, etc. Ce scanning ne transmet aucune donnée et n'affecte pas le fonctionnement des dispositifs sans fil. Le fait de ne pas transmettre de données empêche les systèmes de détection et les utilisateurs de détecter le scan.

Active scanning : ce scanning se base sur les résultats obtenus lors du scanning passif. Le but ici est d'essayer de se connecter aux équipements afin d'effectuer des tests de pénétration ou afin d'exploiter des vulnérabilités.

WiFi scanning

Comme expliqué précédemment, certains adaptateurs ou équipements s'avèrent utiles et nécessaires afin de scanner le réseau. Les cartes WiFi standards ne permettent pas de scanner et d'écouter le réseau, elles permettent juste d'effectuer les actions de base (à savoir échanger une clé d'authentification, se connecter à un access point, etc.). Pour pouvoir analyser le réseau il faut une carte WiFi capable de passer en mode monitoring. Nous proposons ci-dessous (B.1 et B.2) deux dispositifs de la marque alfa. Ces dispositifs sont compatibles avec l'environnement Kali Linux.

FIGURE B.1 – Alfa - AWUS036ACM

FIGURE B.2 – Alfa - AWUS1900

Afin de pouvoir utiliser l'adaptateurr WiFi, il faudra peut être installer le firmware Atheros via la commande suivante :

```
# installer le firmware Atheros necessaire
$ sudo apt-get install firmware-atheros
```

Une fois la carte WiFi installée et connectée au pc sur lequel tourne Kali Linux (par exemple), il suffit de lancer les commandes suivantes pour la passer en mode monitoring :

```
# permet d'obtenir des infos (nom de la carte) sur l'interface reseau
$ sudo airmon-ng
# permet de voir les process incompatibles avec le mode monitor
$ sudo airmon-ng check
# permet d'arreter les process incompatibles
$ sudo airmon-ng check kill
# permet de demarrer le monitoring(ici le nom de la carte est wlan0)
$ sudo airmon-ng start wlan0
# permet de verifier la config reseau
$ sudo iwconfig
# permet d'arreter le mode monitoring
$ sudo airmon-ng stop wlan0mon
# permet de redemarrer le network manager
$ sudo systemctl start NetworkManager
```

Bluetooth scanning

Le bluetooth possède une portée assez faible, on parle en général d'une dizaine de mètres, il faut donc se trouver relativement proche de l'appareil que l'on souhaite tester. *Le scanning actif peut être utilisé pour évaluer le mode de sécurité dans lequel un dispositif Bluetooth fonctionne, et la force des numéros d'identification par mot de passe (PIN) Bluetooth. Le scanning actif peut également être utilisé pour vérifier que ces dispositifs sont réglés sur la puissance opérationnelle la plus faible possible afin de minimiser leur portée.* [73]

Afin de scanner un réseau bluetooth, plusieurs outils différents peuvent être utilisés. On peut par exemple utiliser un ordinateur portable possédant une antenne bluetooth intégrée. On peut utiliser un ordinateur n'en possédant pas mais qui utiliserait un Raspberry Pi 4 (sur lequel on

a installé Kali Linux par exemple). Ubertooth est également un moyen pour capter le signal bluetooth

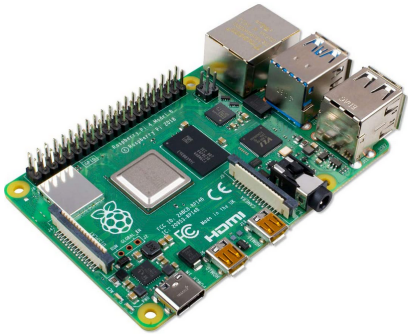


FIGURE B.3 – Raspberry Pi 4

FIGURE B.4 – Ubertooth

Bibliographie

[1] Kali linux - penetration testing and ethical hacking linux. URL https://www.kali.org/. (accédé la dernière fois le 05 août 2020).

[2] H. Abie and I. Balasingham. Risk-based adaptive security for smart iot in ehealth. In *Proceedings of the 7th International Conference on Body Area Networks*, pages 269–275, 2012.

[3] S. Al-Sarawi, M. Anbar, K. Alieyan, and M. Alzubaidi. Internet of things (iot) communication protocols. In *2017 8th International conference on information technology (ICIT)*, pages 685–690. IEEE, 2017.

[4] I. Andrea, C. Chrysostomou, and G. Hadjichristofi. Internet of things : Security vulnerabilities and challenges. In *2015 IEEE Symposium on Computers and Communication (ISCC)*, pages 180–187. IEEE, 2015.

[5] P. Appavoo, E. K. William, M. C. Chan, and M. Mohammad. Indriya2 : A heterogeneous wireless sensor network (wsn) testbed. In *International Conference on Testbeds and Research Infrastructures*, pages 3–19. Springer, 2018.

[6] A. Arora. Preventing wireless deauthentication attacks over 802.11 networks. *arXiv preprint arXiv :1901.07301*, 2018.

[7] A. Arora, E. Ertin, R. Ramnath, M. Nesterenko, and W. Leal. Kansei : A high-fidelity sensing testbed. *IEEE Internet Computing*, 10(2) :35–47, 2006.

[8] A. Arsenault and S. Farrell. Securely available credentials-requirements. Technical report, RFC 3157, August, 2001.

[9] L. Atzori, A. Iera, and G. Morabito. " the internet of things : A survey," computer networks, vol. 54, no. 15, pp. 2787-2805, 2010.

[10] M. Azer, S. El-Kassas, and M. El-Soudani. A full image of the wormhole attacks-towards introducing complex wormhole attacks in wireless ad hoc networks. *arXiv preprint arXiv :0906.1245*, 2009.

[11] S. Babar, P. Mahalle, A. Stango, N. Prasad, and R. Prasad. Proposed security model and threat taxonomy for the internet of things (iot). In *International Conference on Network Security and Applications*, pages 420–429. Springer, 2010.

[12] S. Babar, A. Stango, N. Prasad, J. Sen, and R. Prasad. Proposed embedded security framework for internet of things (iot). In *2011 2nd International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology (Wireless VITAE)*, pages 1–5. IEEE, 2011.

[13] P. Barnaghi and A. Sheth. The internet of things : The story so far. *IEEE Internet of Things*, 915, 2014.

[14] Y. Berhanu, H. Abie, and M. Hamdi. A testbed for adaptive security for iot in ehealth. In *Proceedings of the International Workshop on Adaptive Security*, pages 1–8, 2013.

[15] T. Bhattasali, R. Chaki, and S. Sanyal. Sleep deprivation attack detection in wireless sensor network. *arXiv preprint arXiv :1203.0231*, 2012.

[16] K. P. Bhaturkar and K. G. Bagde. Prevention of session hijacking and ip spoofing with sensor nodes and cryptographic approach. *International Journal of Computer Science and Mobile Computing*, 3(5) :1198–1206, 2014.

[17] H. Bojinov, E. Bursztein, and D. Boneh. Xcs : cross channel scripting and its impact on web applications. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 420–431, 2009.

[18] P. Cao, E. C. Badger, Z. T. Kalbarczyk, R. K. Iyer, A. Withers, and A. J. Slagell. Towards an unified security testbed and security analytics framework. In *Proceedings of the 2015 Symposium and Bootcamp on the Science of Security*, pages 1–2, 2015.

[19] M. Caselli and F. Kargl. A security assessment methodology for critical infrastructures. In *International Conference on Critical Information Infrastructures Security*, pages 332–343. Springer, 2014.

[20] I. Chatzigiannakis, S. Fischer, C. Koninis, G. Mylonas, and D. Pfisterer. Wisebed : an open large-scale wireless sensor network testbed. In *International Conference on Sensor Applications, Experimentation and Logistics*, pages 68–87. Springer, 2009.

[21] J. A. Chaudhry, U. Tariq, M. A. Amin, and R. G. Rittenhouse. Dealing with sinkhole attacks in wireless sensor networks. *Advanced Science and Technology Letters*, 29(2) :7–12, 2013.

[22] K. Chen, S. Zhang, Z. Li, Y. Zhang, Q. Deng, S. Ray, and Y. Jin. Internet-of-things security and vulnerabilities : Taxonomy, challenges, and practice. *Journal of Hardware and Systems Security*, 2(2) :97–110, 2018.

[23] J.-S. Cho, S.-S. Yeo, and S. K. Kim. Securing against brute-force attack : A hash-based rfid mutual authentication protocol using a secret value. *Computer communications*, 34 (3) :391–397, 2011.

[24] O. communauty. Owasp web security testing guide. URL https://github.com/OWASP/wstg.

[25] B. Coppens, I. Verbauwhede, K. De Bosschere, and B. De Sutter. Practical mitigations for timing-based side-channel attacks on modern x86 processors. In *2009 30th IEEE Symposium on Security and Privacy*, pages 45–60. IEEE, 2009.

[26] L. Da Xu, W. He, and S. Li. Internet of things in industries : A survey. *IEEE Transactions on industrial informatics*, 10(4) :2233–2243, 2014.

[27] T. Denning, T. Kohno, and H. M. Levy. Computer security and the modern home. *Communications of the ACM*, 56(1) :94–103, 2013.

[28] C. B. Des Rosiers, G. Chelius, E. Fleury, A. Fraboulet, A. Gallais, N. Mitton, and T. Noël. Very large scale open wireless sensor network testbed. 2011.

[29] D. Djenouri, L. Khelladi, and A. N. Badache. A survey of security issues in mobile ad hoc and sensor networks. *IEEE Communications Surveys and Tutorials*, 7(4), 2005.

[30] O. Doh and I. Ha. A digital door lock system for the internet of things with improved security and usability. *Advanced Science and Technology Letters*, 109 :33–38, 2015.

[31] M. C. Domingo. An overview of the internet of things for people with disabilities. *Journal of Network and Computer Applications*, 35(2) :584–596, 2012.

[32] R. Dorai and V. Kannan. Sql injection-database attack revolution and prevention. *J. Int'l Com. L. & Tech.*, 6 :224, 2011.

[33] H. P. Enterprise. Internet of things research study. *Internet of Things Research Study*, 2015.

[34] FIESTA-IoT. Federated interoperable semantic iot testbeds and applications. URL http://fiesta-iot.eu/. (accédé la dernière fois le 24 juillet 2020).

[35] M. T. Garip, M. E. Gursoy, P. Reiher, and M. Gerla. Congestion attacks to autonomous cars using vehicular botnets. In *NDSS Workshop on Security of Emerging Networking Technologies (SENT), San Diego, CA*, 2015.

[36] Geroges. Introduction aux réseaux iot : une vue d'ensemble. URL https://www.matooma.com/fr/s-informer/actualites-iot-m2m/m2m-comment-connecter-vos-objets.

[37] B. Gourdin, C. Soman, H. Bojinov, and E. Bursztein. Toward secure embedded web interfaces. In *USENIX Security Symposium*, volume 14, page 113, 2011.

[38] S. Greengard. *The internet of things*. MIT press, 2015.

[39] M. L. Hale, K. Lotfy, R. F. Gamble, C. Walter, and J. Lin. Developing a platform to evaluate and assess the security of wearable devices. *Digital Communications and Networks*, 5(3) :147–159, 2019.

[40] T. Hammond. The scary truth about data security with wearables, 2014. URL http://www.techrepublic.com/article/the-scary-truth-about-data-security-with-wearables/. (accédé la dernière fois le 23 juillet 2020).

[41] 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*, 61(3) :527–542, 2011.

[42] M. M. Hossain, M. Fotouhi, and R. Hasan. Towards an analysis of security issues, challenges, and open problems in the internet of things. In *2015 IEEE World Congress on Services*, pages 21–28. IEEE, 2015.

[43] F. IoT-LAB. Iot experimentation at a large scale, 2018. URL https://www.iot-lab.info/. (accédé la dernière fois le 24 juillet 2020).

[44] P. H. ISECOM. *OSSTMM 3 – The Open Source Security Testing Methodology Manual*. ISECOM, 14 december 2010.

[45] X. Jia, Q. Feng, T. Fan, and Q. Lei. Rfid technology and its applications in internet of things (iot). In *2012 2nd international conference on consumer electronics, communications and networks (CECNet)*, pages 1282–1285. IEEE, 2012.

[46] A. Khairi, M. Farooq, M. Waseem, and S. Mazhar. A critical analysis on the security concerns of internet of things (iot). *Perception*, 111, 2015.

[47] R. Khan, S. U. Khan, R. Zaheer, and S. Khan. Future internet : the internet of things architecture, possible applications and key challenges. In *2012 10th international conference on frontiers of information technology*, pages 257–260. IEEE, 2012.

[48] W. Z. Khan, Y. Xiang, M. Y. Aalsalem, and Q. Arshad. The selective forwarding attack in sensor networks : Detections and countermeasures. *International Journal of Wireless and Microwave Technologies (IJWMT)*, 2(2) :33, 2012.

[49] W. Knowles, A. Baron, and T. McGarr. The simulated security assessment ecosystem : Does penetration testing need standardisation ? *Computers & Security*, 62 :296–316, 2016.

[50] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao. A survey on internet of things : Architecture, enabling technologies, security and privacy, and applications. *IEEE Internet of Things Journal*, 4(5) :1125–1142, 2017.

[51] C. H. Liu, B. Yang, and T. Liu. Efficient naming, addressing and profile services in internet-of-things sensory environments. *Ad Hoc Networks*, 18 :85–101, 2014.

[52] A. M. Lonzetta, P. Cope, J. Campbell, B. J. Mohd, and T. Hayajneh. Security vulnerabilities in bluetooth technology as used in iot. *Journal of Sensor and Actuator Networks*, 7 (3) :28, 2018.

[53] L. Ma. Detecting masqueraders in 802.11 wireless networks. In *Proceedings of the International Conference on Wireless Networks (ICWN)*, page 1. Citeseer, 2011.

[54] R. Mahmoud, T. Yousuf, F. Aloul, and I. Zualkernan. Internet of things (iot) security : Current status, challenges and prospective measures. In *2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)*, pages 336–341. IEEE, 2015.

[55] C. M. Medaglia and A. Serbanati. An overview of privacy and security issues in the internet of things. In *The internet of things*, pages 389–395. Springer, 2010.

[56] A. Mitrokotsa, M. R. Rieback, and A. S. Tanenbaum. Classification of rfid attacks. *Gen*, 15693(14443) :14, 2010.

[57] M. Mohsin, Z. Anwar, G. Husari, E. Al-Shaer, and M. A. Rahman. Iotsat : A formal framework for security analysis of the internet of things (iot). In *2016 IEEE Conference on Communications and Network Security (CNS)*, pages 180–188. IEEE, 2016.

[58] R. N. Murty, G. Mainland, I. Rose, A. R. Chowdhury, A. Gosain, J. Bers, and M. Welsh. Citysense : An urban-scale wireless sensor network and testbed. In *2008 IEEE conference on technologies for homeland security*, pages 583–588. IEEE, 2008.

[59] M. Nati, A. Gluhak, H. Abangar, and W. Headley. Smartcampus : A user-centric testbed for internet of things experimentation. In *2013 16th International Symposium on Wireless Personal Multimedia Communications (WPMC)*, pages 1–6. IEEE, 2013.

[60] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani. Demystifying iot security : an exhaustive survey on iot vulnerabilities and a first empirical look on internet-scale iot exploitations. *IEEE Communications Surveys & Tutorials*, 21(3) :2702–2733, 2019.

[61] D. Niewolny. How the internet of things is revolutionizing healthcare, 2014. URL http://cache.freescale.com/files/corporate/doc/white_paper/IOTREVH/EALCARWP.pdf. (accédé la dernière fois le 22 juillet 2020).

[62] OISSG. *Information Systems Security Assessment Framework (ISSAF) Draft 0.2.1B*. OISSG, 01 may 2006.

[63] OWASP. Clickjacking. URL https://owasp.org/www-community/attacks/Clickjacking. (accédé la dernière fois le 30 juillet 2020).

[64] OWASP. Owasp - top 10 internet of things, 2018. URL https://owasp.org/www-pdf-archive/OWASP-IoT-Top-10-2018-final.pdf. (accédé la dernière fois le 09 août 2020).

[65] D. G. Padmavathi, M. Shanmugapriya, et al. A survey of attacks, security mechanisms and challenges in wireless sensor networks. *arXiv preprint arXiv :0909.0576*, 2009.

[66] D. Papp, Z. Ma, and L. Buttyan. Embedded systems security : Threats, vulnerabilities, and attack taxonomy. In *2015 13th Annual Conference on Privacy, Security and Trust (PST)*, pages 145–152. IEEE, 2015.

[67] L. Pycroft and T. Z. Aziz. Security of implantable medical devices with wireless connections : The dangers of cyber-attacks. *Expert review of medical devices*, 15(6) :403–406, 2018.

[68] V. Sachidananda, S. Siboni, A. Shabtai, J. Toh, S. Bhairav, and Y. Elovici. Let the cat out of the bag : A holistic approach towards security analysis of the internet of things. In *Proceedings of the 3rd ACM International Workshop on IoT Privacy, Trust, and Security*, pages 3–10, 2017.

[69] M. G. Samaila, J. B. Sequeiros, M. M. Freire, and P. R. Inácio. Security threats and possible countermeasures in iot applications covering different industry domains. In *Proceedings of the 13th International Conference on Availability, Reliability and Security*, pages 1–9, 2018.

[70] J. Sametinger, J. W. Rozenblit, R. L. Lysecky, and P. Ott. Security challenges for medical devices. *Commun. ACM*, 58(4) :74–82, 2015.

[71] L. Sanchez, L. Muñoz, J. A. Galache, P. Sotres, J. R. Santana, V. Gutierrez, R. Ramdhany, A. Gluhak, S. Krco, E. Theodoridis, et al. Smartsantander : Iot experimentation over a smart city testbed. *Computer Networks*, 61 :217–238, 2014.

[72] A. S. Sastry, S. Sulthana, and S. Vagdevi. Security threats in wireless sensor networks in each layer. *International Journal of Advanced Networking and Applications*, 4(4) :1657, 2013.

[73] K. Scarfone, M. Souppaya, A. Cody, and A. Orebaugh. Technical guide to information security testing and assessment. *NIST Special Publication*, 800(115) :2–25, 2008.

[74] J. Searle, G. Rasche, A. Wright, and S. Dinnage. *NESCOR Guide to Penetration Testing for Electric Utilities Version 3*. 2016.

[75] K. Sha, W. Wei, T. A. Yang, Z. Wang, and W. Shi. On security challenges and open issues in internet of things. *Future Generation Computer Systems*, 83 :326–337, 2018.

[76] S. Siboni, V. Sachidananda, Y. Meidan, M. Bohadana, Y. Mathov, S. Bhairav, A. Shabtai, and Y. Elovici. Security testbed for internet-of-things devices. *IEEE Transactions on Reliability*, 68(1) :23–44, 2019.

[77] V. P. Singh, S. Jain, and J. Singhai. Hello flood attack and its countermeasures in wireless sensor networks. *International Journal of Computer Science Issues (IJCSI)*, 7(3) :23, 2010.

[78] A. Tekeoglu and A. Ş. Tosun. A testbed for security and privacy analysis of iot devices. In *2016 IEEE 13th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pages 343–348. IEEE, 2016.

[79] B. S. Thakur and S. Chaudhary. Content sniffing attack detection in client and server side : A survey. *International Journal of Advanced Computer Research*, 3(2) :7, 2013.

[80] I. Tomić and J. A. McCann. A survey of potential security issues in existing wireless sensor network protocols. *IEEE Internet of Things Journal*, 4(6) :1910–1923, 2017.

[81] R. Uttarkar and R. Kulkarni. Internet of things : architecture and security. *Int. J. Comput. Appl*, 3(4) :12–19, 2014.

[82] A. Vanathi and B. S. Rani. Cloning attack authenticator in wireless sensor networks 1. 2012.

[83] O. A. Waraga, M. Bettayeb, Q. Nasir, and M. A. Talib. Design and implementation of automated iot security testbed. *Computers & Security*, 88 :101648, 2020.

[84] D. Welch and S. Lathrop. Wireless security threat taxonomy. In *IEEE Systems, Man and Cybernetics SocietyInformation Assurance Workshop, 2003.*, pages 76–83. IEEE, 2003.

[85] G. Werner-Allen, P. Swieskowski, and M. Welsh. Motelab : A wireless sensor network testbed. In *IPSN 2005. Fourth International Symposium on Information Processing in Sensor Networks, 2005.*, pages 483–488. IEEE, 2005.

[86] M. E. Whitman and H. J. Mattord. *Principles of information security*. Cengage Learning, 2011.

[87] R. Williams, E. McMahon, S. Samtani, M. Patton, and H. Chen. Identifying vulnerabilities of consumer internet of things (iot) devices : A scalable approach. In *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 179–181. IEEE, 2017.

[88] W. Xu, W. Trappe, Y. Zhang, and T. Wood. The feasibility of launching and detecting jamming attacks in wireless networks. In *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, pages 46–57, 2005.

[89] K. Zhang, X. Liang, R. Lu, and X. Shen. Sybil attacks and their defenses in the internet of things. *IEEE Internet of Things Journal*, 1(5) :372–383, 2014.

[90] Q. Zhang and X. Wang. Sql injections through back-end of rfid system. In *2009 International Symposium on Computer Network and Multimedia Technology*, pages 1–4. IEEE, 2009.

[91] W. Zhang and B. Qu. Security architecture of the internet of things oriented to perceptual layer. *International Journal on Computer, Consumer and Control (IJ3C)*, 2(2) :37–45, 2013.

[92] D. Zhen-hua, L. Jin-tao, and F. Bo. A taxonomy model of rfid security threats. In *2008 11th IEEE International Conference on Communication Technology*, 2008.

[93] T. Zia and A. Zomaya. Security issues in wireless sensor networks. In *2006 International Conference on Systems and Networks Communications (ICSNC'06)*, pages 40–40. IEEE, 2006.