Université Bretagne Sud

Master 1 Ingénierie de Systèmes Complexes

Spécialité Cybersécurité des Systèmes Embarqués

Promotion 2019-2020

Étude du protocole BLE

Stage Master 1

Gidon Rémi

Avril/Juin 2020

Sommaire

Tableaux

Figures

Dans le cadre du master CSSE nous étudions l'internet des objets (IoT) et leurs aspects sécurité. Le protocole réseau sans fil Bluetooth Low Energy (BLE) permet une consommation réduite pour les objets fonctionnant sur batterie, visant notamment les objets connectés. Comme tout les protocoles de la spécification *Bluetooth*, le *BLE* est un protocole à part entière et est donc incompatible avec les autres.
Aujourd'hui integré dans la plupart des appareils de bureautique, il est rapidement devenu populaire dans l'internet des objets.

La première itération du BLE (sortie en 2011) ne répond plus aux exigences de sécurité contemporaine et même si le protocole à su évoluer depuis pour répondre à ces besoins, beaucoup d'appareils utilisent encore la version originale n'intégrant pas ces mécanismes. Ce sont pour la plupart des appareils conçus pour fonctionner sur batterie et communiquer en point à point. On va retrouver les capteurs corporels pour santé ou fitness mais également des mecanismes plus sensibles tels des cadenas ou serrures. Les communications (incluant parfois des données personnelles) peuvent êtres interceptées, voir modifiées pour permettre des actions aux depends de l'utilisateur (comme l'ouverture d'un cadenas).

1 Objets connectés

Avec l'explosion de l'internet de objets au cours de ces dernières années, toute une floppée d'objet du quotidien ont étés augmentés pour permettre la communication avec d'autres systèmes informatique dont nos smartphones ou encore des serveurs distants (via notre WiFi). Ces objets dits intelligents étendent leur équivalent mécanique en intégrant des composants éléctroniques, permettant notamment le contrôle à distance.
Face à l'engouement du public, les constructeurs s'efforcent de proposer des objets toujours plus *intelligents* et connectés, souvent au détriment de la sécurité. Ces améliorations engendrent une augmentation de la surface d'attaque: les objets connectés sont confrontés aux mêmes challenges que ceux des systèmes informatiques traditionnels en plus de leur fonction primaire.

1.1 Architecture

Les objets connectés ont commencés par proposé des communications avec nos smartphones, notre routeur WiFi ou notre ordinateur. Celles-ci permettent d'utiliser ces objets comme télécommande de controle (via une application dediee la plupart du temps) ainsi que de communiquer aux services distants du constructeur en s'appuyant sur les fonctionnalités de celle-ci (WiFi, données mobiles).
Ces architectures point a point connectent un appareil directement à un controleur (*smartphone*, *PC*) duquel il est dependant pour acceder aux services distants (si il y a). C'est notamment tres utilisé pour les appareils ajoutant une fonction d'augmentation seule sur un produit existant (par exemple les cadenas connectés). Ces appareils n'ont souvent pas besoin d'un service distant puisqu'ils proposent un modele d'interactions simple et local avec un utilisateur.

Les objets connectes ont rapidement etes utilises pour mettre en place des reseaux de capteurs. Cette utilisation à été largement introduite pour un usage domestique avec la domotique. Même si pouvoir controler sa temperature depuis chez soi en utilisant son smartphone est interessant, la demande a poussé les constructeurs a relier leurs objets connectes au reseau mondial pour permettre un controle depuis n'importe quel emplacement. Certains ont optés pour incorporer des puces WiFi directement dans leurs objets quant a d'autres ont proposés une solution plus long terme en mettant un place une passerelle (appelee *hub*) gerant les interactions avec le monde exterieur.

L'architecture *hub* est aujourd'hui vastement utilisée pour la domotique. Un appareil dédié est considéré comme *hub* auquel les capteurs, beaucoup plus simples, communiquent. Des protocoles spécialisé ont fait leur apparition comme *Zigbee*[1], *Z-Wave*[2], *ANT+*[3] ou encore *Thread*[4]. La ou les objets connectes "simples" (realisant une fonctionnalite d'augmentation seule) profitait de la standardisation des protocoles disponibles dans les smartphones (bluetooth principalement), les reseaux basés sur un *hub* imposent l'utilisation d'un protocole (parfois proprietaire) et d'appareils compatibles avec celui-ci.

[1] https://zigbeealliance.org/
[2] https://www.z-wave.com/
[3] https://www.thisisant.com/
[4] https://www.threadgroup.org/

1.2 Protocoles

Comme évoqué precedemment, beaucoup d'objets connectes ont profiter des protocoles integres dans les appareils utilisés comme controleur. Cela englobe le WiFi, le Bluetooth et dernierement le NFC. De ces trois, le Bluetooth a largement pris le dessus car plus adapté avec son standard *Low Energy*. Concu en tant que *WPAN* (reseau sans fil personnel) il permet de communiquer dans un rayon de 10 metres, suffisant pour les interactions locales. Le NFC est assez récent, il a ete concu pour les interaction proches (une dizaine de centimetres) et concerne des actions intentionnelles comme les paiements. Enfin le WiFi aurait du etre largement utilisé, puisque chaque foyer dispose d'une box internet, mais sa consommation est telle qu'un objet sur batterie ne tient que quelques heures au maximum (il n'y a qu'a voir les ordinateurs portables, disposants pourtant de larges batteries). Il n'est tout simplement pas adapteé au besoin puisqu'il permet des hauts debits et faible latence pour une consommation elevee la ou les objets connectes utilisent de faibles debits occasionnellement pour economiser leurs batteries.

Avec la democratisation des objets connectés de nouveaux protocoles specialises ont fait leur apparition. Meme si le *BLE* (*Bluetooth Low Energy*) a su s'adapté pour repondre aux besoins de ce marché, il n'a pas ete initialement concu pour repondre aux besoins contemporains dans ce milieu.
Tout les plus grands constructeurs (notamment Google et Apple) ont developpé leur standard, le vantant et l'imposant avec leurs produits et architectures proprietaires. Apple Home utilise *Darwin* (iOS, macOS) comme controleur ainsi que son propre protocole (*HAP*) pour ses objets connectes. Google a mis en place le protocole *Thread*, interoperable avec *Google Home* (*hub*) et *Android* (*controleur*). Bien avant, des constructeurs specialisés ont developpés *Zigbee*, *Z-Wave* ou encore *ANT+*.
Bref, beaucoup de protocoles se battent pour avoir acces a un marché juteux encore instable car en plein developpement. Il n'empeche que tout les objets connectés, meme contemporains, n'utilisent pas ces architectures et se basent encore sur beaucoup sur le *BLE* (qui continu d'avancé, proposant des ameliorations interessantes).

2 Bluetooth Low energy

Le protocole a ete designé par Nokia et d'autres entreprises pour repondre au besoin d'un protocole sans fil peu gourmand en energie pour les peripheriques personnels (telephone portable, montre, casque). Nommé Wibree, il a ete integre au standard Bluetooth sous le nom *Low Energy*.
Le Bluetooth ne comprend pas seulement un protocole mais une multitude d'entre eux (BR, EDR, HS) qui ont en commun de permettre la communication (et l'echange de donnees) sans fil avec des peripheriques personnels. Ils font partit des protocoles WPAN (reseau personnel sans fil) et leur distance d'emission est de quelques metres jusqu'a 30 metres.
La specification Bluetooth 4.0, sortie en 2011, integre le protocole LE (Low Energy) et permet au Bluetooth de touché le marché des systemes embarques, fonctionnants sur batterie.

2.1 Différences

Les autres protocoles du Bluetooth sont principalement connus et utilises pour le transfert de contenu multimédia, que ce soit des fichiers entre ordinateurs comme de la musique avec un casque ou encore une voiture. Ils fonctionnement avec une connexion continue et un transfert en flux.

Le BLE, visant a reduire la consommation d'energie, n'etablie pas de connexion continue. L'appareil reste la plupart du temps en mode veille, pouvant emettre des annonces, dans l'attente d'une connexion qui aura pour effet d'arreter la transmission d'annonce. Pour chaque requete recu, une reponse pourra etre renvoyee directement ou une notification mise en place periodiquement.

Les appareils BLE et Bluetooth BR/EDR ne sont pas compatibles, n'utilisant pas les memes technologies, protocoles et repondant a des besoin differents (voir tbl. 1).

Tableau 1: Cas d'utilisation et protocole Bluetooth adapté

Besoin	Flux données	Transmisson données	Localisation	Reseau capteurs
Appareils	ordinateur, smartphone, casque, enceinte, voiture	accessoires bureautique ou fitness, equipement medical	beacon, IPS, inventaire	automatisation, surveillance, domotique
Topologie	point à point	point à point	diffusion (1 à N)	mesh (N à N)
Technologie	Bluetooth BR/EDR	Bluetooth LE	Bluetooth LE	Bluetooth LE

2.2 Protocole

Pour permettre une interoperabilité maximale entre les appareils BLE, le standard defini 4 profiles en fonction du role de l'appareil: Peripheral, Central, Broadcaster, Observer. Ces

roles constituent le *GAP* (*Generic Access Profile*).
Chaque appareil se conformant au standard ne doit implementer qu'un seul de ces roles a la fois.

Le *Broadcaster* ne communique qu'avec des annonces, on ne peut pas s'y connecter. Ce mode est tres populaire pour les beacons. L'*Observer* est sont opposé, il ne fait qu'ecouter les annonces, n'etabliera jamais de connexion.

Le *Peripheral* et le *Central* forment la seconde pair et permettent la mise en place d'une architecture client-serveur. Le *Peripheral* joue le role du serveur et est dit *esclave* du *Central* qui endosse le rôle du client et *maître*.
L'esclave transmet des annonces jusqu'a recevoir une connexion d'un maitre, apres quoi il arrete de s'annoncer car il ne peut etre connecté qu'a un maitre a la fois. Le maitre ecoute les annonces d'esclave (annonces connectables) pour se connecter, puis interroge ses services via le *GATT* (*Generic Attribute*).

Couche physique

Le BLE opère dans la bande ISM 2.4GHz tout comme le Wi-Fi. Contrairement aux canaux Wi-Fi de 20MHz, le BLE découpe le spectre en 40 canaux de de 2MHz (plage de 2400 à 2480MHz).
Le protocole met en place le *saut de fréquence*, consistant à changer de canal d'émission tout les laps de temps donné, pour réduire le risque de bruit sur les fréquences utilisées (la bande ISM 2.4Ghz étant libre d'utilisation).
Sur les 40 canaux que compose le spectre, 3 sont utilisés pour la transmission d'annonce. Ils sont choisit pour ne pas interferer avec les canaux Wi-Fi car les deux protocoles sont amenés à coexister (voir fig. 1).
Les 37 autres canaux sont utilisés pour les connexions. Chaque connexion va utiliser un sous-ensemble des 37 canaux (appelé carte des canaux) pour éviter les interferences avec les autres connexions BLE. Un seul canal transmet des donnees a la fois mais tous les canaux de la carte sont utilises pour le saut de frequences.

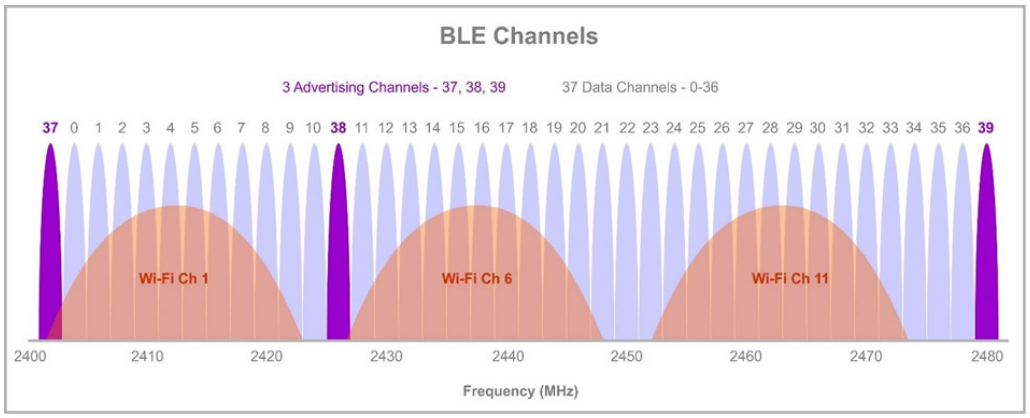

Figure 1: Répartition du spectre BLE en canaux[5]

Couche logique

1. Annonces

[5]https://www.accton.com/Technology-Brief/ble-beacons-and-location-based-services/

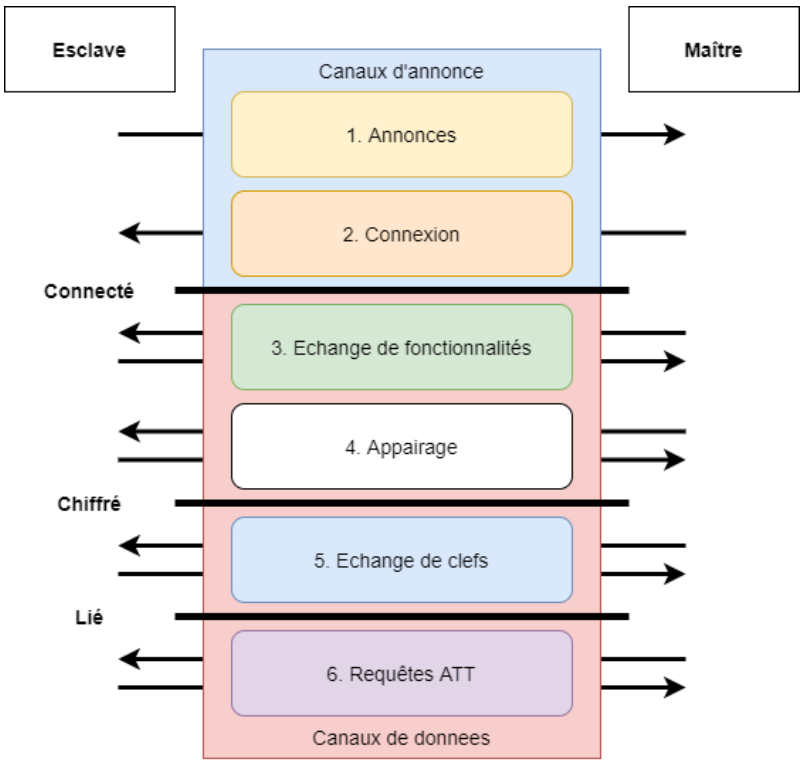

Figure 2: Étapes d'un échange BLE

L'esclave indique sa présence avec des annonces émises périodiquement. Ces annonces contiennent sont addresse Bluetooth (permettant une connexion) et des données qui consituent un profile (appelé *GAP*[6]). Ces données permettent aux maîtres de savoir si il est capable de realiser les fonctionnalités recherchées.
La specification Bluetooth definit des profiles type pour des applications communes dans les appareils BLE[7]. Cela inclus par exemple les capteurs corporels pour le sport, les capteurs médicaux de surveillance (pour les diabetiques notamment), la domotique (termometres, lampes), etc.

Dans un environnement BLE, les maîtres ne peuvent pas reconnaitre leurs esclaves a part avec une addresse Bluetooth fixe, mecanisme de moins ne moins utilisé car vulnérable a l'usurpation. Les esclave generent donc des addresses aleatoires et l'identification se fait via les donnees du *GAP* contenues dans l'annonce. Ce mecanisme permet a n'importe quel maitre de s'appairer a n'importe quel esclave proposant le profil recherché.
Par exemple, une application de smartphone BLE pouvant gerer la temperature pourrait s'appairer et utiliser n'importe quel appareil BLE qui implemente le profil standardisé pour les termometres dans le *GAP*.
Les profils ne sont certes pas exhaustifs mais permettent une integration fonctionnelle avec un maximum d'appreils et prévoient un moyen d'integrer des donnees proprietaires non standardisées[8].

2. Connexion

[6]https://www.bluetooth.com/specifications/assigned-numbers/generic-access-profile/
[7]https://www.bluetooth.com/specifications/gatt/services/
[8]https://www.silabs.com/community/wireless/bluetooth/knowledge-base.entry.html/2017/02/10/bluetooth_advertisin-hGsf

Lorsqu'un maitre recois une annonce d'un esclave auquel il souhaite se connecter, il lui envoit une intention de connexion sur les canaux d'annonce. Ce message contient tout les parametres communs pour etablir une connexion sur les canaux de donnees: carte des canaux utilisés, temps entre chaque saut de frequence, nombre de canaux sautés par saut, addresse unique de la connexion (appelée *Access Adress*).

Ce message (nommé *CONNECT_REQ*) est crucial lors d'attaques car il permet la synchronisation avec une connexion pour l'ecoute passive et est donc jugé sensible puisque transmit sur les canaux d'annonces avant la mise en place du chiffrement.

3. Capacités

Le BLE voulant garder une interoperabilité maximale entre les appareils et tout les appareils ne disposant pas des memes fonctionnalites embarques, il est definit plusieurs methodes d'appairage en fonction des capacites disponibles sur les deux appareils.

Chaque appareil va transmet ses capacités a l'autre ainsi que ses exigences sur la connexion a etablir. Les capacités sont deduites des fonctionnalités presentent physiquement sur l'appareil et les exigences de la version du procole actuellement supportée par celui-ci.
Les exigences comprennent la protection aux attaques *MITM* par l'authentification de l'appairage, l'etablissement d'une connexion sécurisée (*LE secure connection*), la mise en place d'une session (*Bonding*) pour une reconnexion future ainsi que l'utilisation d'un canal autre que le BLE (comme le *NFC*) pour la transmission de secrets menant au chiffrement (*Out Of Band*).

Tableau 2: Capacités d'entrée possibles[9]

Capacité	Description
No input	pas la capacité d'indiquer *oui* ou *non*
Yes/No	mécanisme permettant d'indiquer *oui* ou *non*
Keyboard	claver numérique avec mécanisme *oui*/*non*

Tableau 3: Capacités de sortie possible

Capacité	Description
No output	pas la capacité de communiquer ou afficher un nombre
Numeric Output	peut communiquer ou afficher un nombre

Tableau 4: Capacité d'entrées/sorties de l'appareil

	No output	Numeric output
No input	NoInputNoOutput	DisplayOnly
Yes/No	NoInputNoOutput	DisplayYesNo
Keyboard	KeyboardOnly	KeyboardDisplay

[9]https://www.bluetooth.com/blog/bluetooth-pairing-part-1-pairing-feature-exchange/

4. Appairage

En fonction des capacites et des exigences emits par chacun des appareils, une methode d'appairage est selectionnée (voir tbl. 5).

Tableau 5: Méthode d'appairage utilisée en fonction des capacités échangées[10] (JW=JustWorks PK=PassKey NC=NumComp)

	DisplayOnly	**DisplayYesNo**	**KbdOnly**	**NoIO**	**KbdDisplay**
DisplayOnly	JW	JW	PK	JW	PK
DisplayYesNo	JW	JW/NC	PK	JW	PK/NC
KbdOnly	PK	PK	PK	JW	PK
NoIO	JW	JW	JW	JW	JW
KbdDisplay	PK	PK/NC	PK	JW	PK/NC

Je m'interesse principalement a la methode *JustWorks*. C'est la methode par defaut lorsque deux appareils ne disposent pas des capacites necessaires pour une autre. Elle est notamment utilisee dans les objets connectes puisqu'ils n'integrent pas de mecanisme pour un appairage plus complexe.
Passkey et *NumComp* permettent d'authentifier l'appairage pour se proteger des usurpations d'identite (*MITM*) puisque partageant un secret via l'utilisateur (ou un autre canal dans le cas du *OOB*). *JustWorks* ne permet pas d'authentifier les appareils et le chiffrement est moins robuste que les autres methodes mais permet tout de meme d'etablir une communication chiffree.

La méthode d'appairage choisie permet de transmettre un des materiaux cryptographique: la clef temporaire (ou *Temporary Key*). Cette phase est plus ou moins sensible à l'ecoute passive en fonction de la methode d'appairage et des exigences emits lors de l'echange des capacites.
JustWorks avec connexion BLE 4.0 (dite *legacy*) est le mode le plus sensible puisque la clef temporaire est tout simplement zéro, ne disposant pas de moyen de transmettre une données par autre voie, et peut donc etre trouvee rapidement par bruteforce.

La connexion *LE secure*, introduite a partir de la version 4.2, utilise l'algorithme Diffie-Hellman (*ECCDH* exactement) pour l'echange des materiaux cryptographiques et est donc resistante a l'ecoute passive (*eavesdropping*) mais toujours vulnerables a l'usurptation d'identite (*MITM*) avec certaines methodes d'appairage (*JustWorks*).

5. Echange de clefs

L'etablissement du chiffrement de la connexion est ensuite realisé par derivation a partir d'une premiere clef temporaire transmise via la methode d'appairage choisie et d'autre parametres echanges via le protocole BLE. La clef obtenue est dite court terme (*Short Term Key*) car elle ne sera utilisee que pour cette connexion et devra etre re-generee a chaque nouvelle connexion.

Dans le cas des connexion securisees, l'algorithme *ECCDH* est utilisé pour échanger la clef temporaire duquelle une clef long terme (*Long Term Key*) est dérivée.
Il est possible de reutiliser les clefs long terme avec la mise en place d'une session si cela a

[10] https://www.bluetooth.com/blog/bluetooth-pairing-part-2-key-generation-methods/

ete exigé lors de l'echange des capacités. La clef long terme (*LTK* pour *Long Term Key*) est stockée et associée à l'appareil communiquant pour rétablir une connexion future sans avoir à refaire une phase d'appairage.

A partir de la comprehension actuelle du protocole BLE et du fonctionnement de l'appairage, il semble recommandé de mettre en place une connexion securisee des que possible. Il est egalement necessaire d'eviter la methode *JustWorks* au maximum.
Cependant, il est assez simple de forger un echange de capacités pour retrograder la connexion en *legacy* et forcer *JustWorks* via les capacités echangees.
C'est pourquoi certains appareils attendent des capacites et exigences minimales pour etablir une connexion, sans quoi celle-ci est avortee. C'est notamment le cas d'appareils proprietaires concus pour fonctionner ensemble.

6. Requêtes

Les échanges sont realises sur la base d'une architecture client-serveur. Le maître (client) interroge l'esclave (serveur) avec le protocole *ATT* (*ATTribute Protocole*).
Chaque requete mene soit a une reponse du serveur soit a la mise en place d'une notification lors d'un evenemment (valeur changée ou disponible).

Les requetes et reponses possibles sont standardisées sous le *GATT* (*Generic ATTributes*) pour permettre une interoperabilité maximale entre les appareils (comme pour le *GAP*). *GATT* et *GAP* partagent les memes profiles, seul la structure change. Le serveur *GATT* peut etre interrogé pour etablir une liste exhaustive de toutes les fonctionnalites d'un appareil la ou le *GAP* choisit ce que contient l'annonce mais est limite par la taille du paquet (31 octets).

GAP

Dans le cas des *Peripherals* et *Centrals*, le *GAP* est principalement utilisé pour etablir un profil de l'esclave permettant la decision de connexion de la part du maitre.
Pour les *Boardcasters* et *Observers* il permet la communication unidirectionnelle (*Broadcaster* vers *Observer*) via les annonces, ceux-ci utilisant la diffusion plutot qu'une connexion point a point. On retrouve cette utilisation pour les beacons publicitaires ou de localisation interieur.

GATT

Pour l'echange de données lors de connexion point à point, le *GATT* est utilisé en mode client-serveur. L'architecture du serveur *GATT* est en entonnoir, la plus haute couche s'appelle un *service*, il encapsule des *caracteristiques*, chacune contenant un *attribut* (valeur) et un ou plusieurs *descripteurs* fournissants des informations additionnelles sur l'attribut (voir fig. 3).
A chacune de ses couches (service, caracteristique, attribut, descripteur) est attribué un identifiant unique appelé *handle*. La plage des indentifiant est partagée entre toutes les couches donc si un service a l'identifiant `0x01` aucun autre service/caracteristique/attribut/descripteur ne peut l'utiliser.

Un service correspond generalement a un profil (standardise ou non) comme un termometre par exemple. Ce service exposerait des caracteristiques comme la temperature, l'humidite ou autres. Chacune de ces caracateristique contient la valeur (donnée brute) et les descripteurs

peuvent indiquer l'unité ou encore un facteur ou formule pour convertir la valeur donnée en resultat exploitable.

A moins de connaitre exactement l'appareil et de l'interroger en mode aveugle via les identifiants (ce qui peut etre le cas entre des appareils proprietaires), il faut proceder par etape en decouvrant d'abors les services disponibles, puis chaque caracteristique par service et enfin les attributs de celles-ci.
Pour proceder a cette decouverte d'un appareil, le protocole *ATT* dispose d'un type de requete par couche a interroger (voir fig. 3). Une fois le service voulu trouvé (ou la cartographie totale de l'appareil realisée), on peut lire, ecrire ou souscrire a des attributs directement par *handle*. Le *GATT* met en place un systeme de droits par attribut pour proteger la lecteur, l'ecriture et la souscription par le client.

Le *GATT* définit egalemet des services standardisés appelé primaire et secondaire censés etres present sur tout les appareils BLE afin de connaitre les fonctionnalites principales de l'appareil. Comme les *handle* sont definies arbitrairement par le serveur *GATT*, ces services sont identifiés par un *UUID* identique dans tout les appareils BLE.[11]

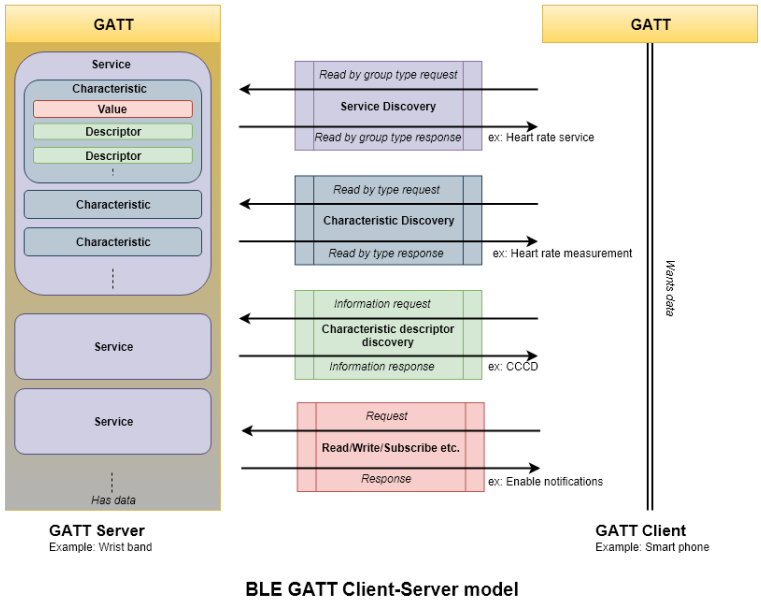

Figure 3: Client et serveur GATT[12]

2.3 Versions

Depuis sa premiere iteration en 2011 dans la version `4.0` des specifications Bluetooth le BLE a evoluer pour integrer des mesures de securite avec l'ajout des connexions securisees LE en `4.2` puis la diversification des topologies avec l'introduction du *mesh* pour les reseaux de capteurs en `5.0` et dernierement l'amelioration de la localisation interieur (*Indoor Positionning System*) pour une precision de l'ordre du centimetre grace aux systemes angle d'arrivée et de depart (*AOA/AOD*).

[11] https://www.bluetooth.com/specifications/gatt/services/
[12] https://fr.mathworks.com/help/comm/examples/modeling-of-ble-devices-with-heart-rate-profile.html

3 Étude de l'existant

3.1 Outils

Les outils offensifs sur le protocole BLE permettent de recuperer, analyser et modifier les echanges entre appareils permettant de realiser des audits de securite ou mettre en place des attaques exposant des vulnerabilites. L'analyse du traffic sans fil BLE demande une antenne couvrant la bande utilisee par les 40 canaux du protocole ainsi qu'un systeme assez rapide pour scanner puis suivre les communications lors des sauts de frequence. Les radio-logiciels (*SDR*) ne sont donc pour la plupart pas adaptés car trop lents ou trop cher pour les fonctionnalités voulues: des outils specialises dans l'analyse et l'attaque du BLE sont disponibles pour une fraction du prix.

Le premier outil, utilisé dans tout nos appareils BLE, est la puce intégrée pour les communications BLE. La recuperation (*sniffing*) et analyse ou modification d'un traffic sans fil etant interdit, ces puces utilisent un *firmware* proprietaire ne permettant que la communication en *master* ou *slave* et respectant les specifications BLE. Même si il reste possible d'analyser le traffic (notamment en utilisant *Wireshark*[13]) entre la puce et un appareil BLE, il n'est pas possible d'etendre les capacites de celle-ci sans modifier le *firmware*.
Tout les appareils ne disposant pas d'une puce BLE dédiée, les constructeurs ont developpes des *dongles* integrant ces puces et permettant de communiquer avec tout appareil USB via une interface nommée *HCI* (*Host-Controller Interface*). Allié aux outils standard du protocole BLE comme *BlueZ*, la pile protocolaire BLE du noyau Linux, ces *dongles* permettent de découvrir les appareils a proximité et d'endosser le role d'esclave ou maitre pour etablir une communication avec n'importe quel autre appareil BLE. Les utilitaires *hcitool*, *hciconfig* et *gatttool* de *BlueZ* permettent par exemple de manipuler les annonces et extraire le profil *GATT* d'un appareil BLE. Meme si certains de ces *dongles* proposent des fonctionnalites interessantes comme le changement d'adresse Bluetooth, ils n'ont pas été conçus dans une optique de sécurité, et sont peu flexibles pour un usage offensif.

La plupart des attaques sur le protocole BLE requierts un moyen d'intercepter le traffic. Les *dongles* et *puces* embarquants des firmware ne permettant pas cette fonctionnalite puisque destines au grand public, beacoups d'outils specialises ont etes developpes. On va retrouver des outils d'analyse de protocole sans fil generaux comme la *HackRF* ou sa version specialement concue pour le BLE nommée *Ubertooth One*. Ces cartes sont assez cher mais hautement personnalisables depuis les couches bas niveau. Elles demandent un certain background de connaissances sur le protocole et les modulations sans fil pour arriver a un resultat precis (comme la realisation d'une attaque).
Viennent ensuite les *sniffers* sous forme de dongle USB arrangées et plus ou moins personnalisables. Beaucoup sont basés sur les memes puces de *Nordic Semiconductor* ou *Texas Instrument* qui eux meme proposent leurs sniffers[14,15] et logiciels[16,17] pour l'analyse du protocole BLE. Dans les initiatives plus open-source, mais pas encore totalement personnalisable sans reprogrammation de la puce, on peut citer le Bluefruit[18] de *Adafruit*.
Enfin, un outils open-source nommé *BTLEJack* permet non seulement l'etude mais la mise

[13] https://www.wireshark.org/
[14] https://www.nordicsemi.com/Software-and-tools/Development-Kits/nRF51-Dongle
[15] http://www.ti.com/tool/CC2540EMK-USB
[16] https://www.nordicsemi.com/Software-and-tools/Development-Tools/nRF-Sniffer-for-Bluetooth-LE
[17] http://www.ti.com/tool/PACKET-SNIFFER
[18] https://www.adafruit.com/product/2269

en place d'une multitude d'attaques sur le protocole Ble via reprogrammation de la carte avec un firmware personnalisé. Cet outils a ete developpe pour la carte *BBC Micro:Bit*[19], une carte de developpement bon marché a but educatif, et est aujourd'hui compatible avec plusieurs autres cartes intégrant une puce `nRF51` (notamment la *Bluefruit*). Basé sur les travaux de *BTLEJack*[20] et d'autres librairies BLE en python, *Mirage*[21] permet des fonctionnalités identiques en supportant encore plus de cartes, de protocoles et d'attaques. Il comble le manque de flexibilité des precedants outils en integrant plusieurs mecanismes permettant la mise en place d'attaques scénarisées entierement personnalisees depuis les couches protocolaires basses et facilite l'ajout de fonctionnalités au sein du framework.

3.2 Attaques

Scanning

Le *scanning* consiste à répertorier des appareils BLE à proximité. Dans le cas d'attaque on etendra l'inventaire avec les connexion établies entre 2 appareils BLE. Là ou les *dongles HCI* suffisent pour intercepter les annonces diffusées, l'analyse des communications établies requiert un *sniffer* capable de suivre les 37 canaux de données.
La pile protocolaire *BlueZ* permet le *scan* des *advertisements* (annonces) tandis que plusieurs outils precedements evoques comme *smartRF* ou *nRFSniffer* suffisent pour repérer une communication.

Spoofing

C'est l'une des étape du *Man-In-The-Middle* qui permet d'usurper un esclave BLE. Après identification de la victime (via annonce ou adresse BD), l'attaquant la clone en s'y connectant et extractant son profile *GATT*. L'attaquant peut resté connecté pour garder la victime silencieuse (un esclave connecté n'emettant pas d'annonces) puis, via un second *dongle* BLE, s'annonce comme étant l'appareil precedemment cloné.
Cette attaque est realisable en utilisant simplement un *dongle HCI* et l'utilitaire *BLueZ*. Bien sur, les librairies et frameworks d'attaque discutés plus auparavant (*GATTAcker*, *BTLEJack*) integrent egalement ces mecanisme.

Sniffing

Le *sniffing* est l'analyse voir le suivis d'une connexion BLE (suivant les capacites du *sniffer* utilisé). Un premier cas de figure est l'attente d'une nouvelle connexion pour se synchroniser avec afin de suivre les echanges. La seconde option, et la plus courante, est la synchronisation avec une connexion deja etablie: la difficulté ici réside en la recuperation des parametres de connexion. Il est necessaire de retrouver la carte des canaux utilisés (*channel map*) ainsi que le *hop increment* (nombre de canaux sautés) et *hop interval* (temps entre chaque saut) pour se synchroniser, sans quoi il est impossible de suivre une connexion car les sauts impredictibles et trop frequents.
BTLEJack, et par conséquent *Mirage*, mettent en place un mecanisme permettant de retrouver ces informations de connexion a partir des echanges interceptés lors du *scanning*. Le *sniffing* de communications sans synchronisation quant a lui est une fonctionnalité très répandue et integrée a tout les sniffers BLE vu antierieurement.

[19] https://microbit.org/
[20] https://github.com/virtualabs/btlejack
[21] https://homepages.laas.fr/rcayre/mirage-documentation/index.html

Man-In-The-Middle

L'attaque *Man-In-The-Middle* concerne n'importe quelle communication: l'attaquant peut modifier les communications en vennant se placer entre l'emetteur et le recepteur ciblés, se faisant passer pour l'un apres de l'autre en usurpant leurs identités. Dans le cas du BLE on utilisera deux *dongles*, un pour usurpé l'esclave cible et un autre pour maintenir une connexion avec celui-ci. On doit d'abors usurpé l'esclave cible via du *spoofing* puis attendre la connexion d'un maitre, une fois le maitre connecté à notre *dongle* usurpateur on se retrouve en situation de *Man-In-The-Middle* entre l'esclave et le maitre: on peut suivre et modifier le traffic avant de le retransmettre. A noté que l'on peut egalement usurpé le maitre si les appareils ciblés attendent un appareil précis pour s'appairer.
Plusieurs outils sont dediés a cette attaque car populaire et simple a mettre en oeuvre: *GATTAcker* et *BTLEJuice* facilitent la mise en place en automatisant les etapes a partir de la cible choisie. Ce sont d'assez ancien outils qui aujourd'hui souffrent de lacunes de part les technologies utilisees. Basés sur *Noble* et *Bleno*, des librairies en JavaScript basées sur NodeJS et permettant de manipuler le BLE, ils manque de flexibilité et ne permettent pas entre autre la coexistence d'appareils BLE, obligant l'utilisation de machine virtuelle pour chaque *dongle*. *Mirage* reprend le fonctionnement de ces outils, l'integrant en tant que module, mais basé sur de nouvelles librairies, notamment *PyBT*[22] permettant de simuler le comportement d'un appareil BLE en s'affranchissant des contraintes imposees par leurs equivalent JavaScript, *Noble* et *Bleno*.

Jamming

Le brouillage de communication est egalement une attaque assez populaire et implementee dans bon nombre d'outils sur le marché. Le but est de creer du bruit sur le canal au moment de la transmission pour corrompre le message, le rendant inutilisable par le recepteur. Concernant le BLE, *Ubertooth One* dispose des capacités nécessaire pour brouiller les canaux d'annonce ainsi qu'une communication etablie par retransmission simultanee. *BTLEJack* implemente egalement le brouillage au sein de son firmware personnalisé et, ayant deja un mecanisme permettant de se synchroniser avec une communication, peut egalement brouiller une communication etablie.

Hijacking

Le principe est de voler une connexion entre 2 appareils en forcant une deconnexion de l'un pour prendre sa place. Cette nouvelle attaque, implementée par *BTLEJack* et reprise dans *Mirage*, utilise les différences de *timeout* entre le *Master* et le *Slave* pour forcer le *Master*, via l'utilisation de jamming sur les paquets du *Slave*, à se déconnecter et prendre ainsi sa place au sein de la communication.

3.3 Mirage

Après comparaison entre les outils disponibles (voir tbl. 6), j'ai choisie *Mirage* car il dispose de la flexibilite voulue pour implementee des attaques scenarisee: acces aux couches bas niveau pour recuperer informations comme force du signal et calibrage (necessaires pour calculer la position d'un appareil lors de la localisation). Il dispose egalement d'une implementation d'un *Master* et *Slave* personnalisables pour realiser un reseau de tests sur lequel verifie l'implementation des attaques. Enfin, il supporte une varieté de composants

[22] https://github.com/mikeryan/PyBT

matériel[23] ainsi que toutes les attaques necessaires[24] pour la preuve de concept, rendant le developpement plus simple.

Concernant le matériel necessaire a la preuve de concept il me faudra d'abors deux *dongles HCI* compatibles avec Mirage et supportant le changement d'adresse *BD* pour le *spoofing*. *Mirage* se base sur les numeros de constructeurs des *dongles* definit par le Bluetooth[25] pour savoir s'ils sont compatibles. Il supporte une variete des constructeur dont le *CSR8510*[26] de Qualcomm, puce tres populaire dans les *dongles HCI* basiques et permettant le changement d'adresse *BD*.
Concernant le *sniffer* necessaire pour la pupart des attaques, *Mirage* se basant sur *BTLE-Jack*, les cartes supportées par celui-ci le sont aussi par *Mirage*. Même si *Mirage* supporte d'autres cartes comme celles de developpement de *Nordic Semiconductor* ou l'*Ubertooth One*, elles ne sont pas adaptées pour mon projet car trop cher pour les fonctionnalités exploitees dans la preuve de concept.
Je me suis donc tourné vers les cartes compatibles avec *BTLEJack*[27]. *Bluefruit* d'Adafruit, Waveshare *BLE400*[28] et les kits *nRF51*[29] demandent une reprogrammation via un peripherique externe utilisant le port *SWD*. Ne disposant pas du materiel necessaire pour la reprogrammation, et celui-ci etant assez onéreux, j'ai choisit la *BCC Micro:Bit*: carte avec laquelle *BTLEJack* à ete originalement developpé.

Tableau 6: Comparaison des outils pour l'étude offensive du BLE

Logiciel	*scan*	*sniff*	*mitm*	*jam*	*hijack*	*locate*	Matériel
nRFSniffer	oui	oui	non	oui	non	non	puce `nRF51`
TIsmartRF	oui	oui	non	non	non	non	puce `CC25xx`
BTLEJuice	oui	non	oui	non	non	non	*dongle HCI* + *Bleno*/*Noble*
GATTAcker	oui	non	oui	non	non	non	*dongle HCI* + *Bleno*/*Noble*
BTLEJack	oui	oui	oui	oui	oui	non	*BBC Micro:Bit*, cartes basées sur puce `nRF51`
Mirage	oui	oui	oui	oui	oui	possible	*dongle HCI*, *Ubertooth*, *nRF*, cartes compatibles avec *BTLEJack*

[23] https://homepages.laas.fr/rcayre/mirage-documentation/devices.html
[24] https://homepages.laas.fr/rcayre/mirage-documentation/modules.html
[25] https://www.bluetooth.com/specifications/assigned-numbers/company-identifiers/
[26] https://www.qualcomm.com/products/csr8510
[27] https://homepages.laas.fr/rcayre/mirage-documentation/devices.html#btlejack-device
[28] https://www.waveshare.com/ble400.htm
[29] https://www.waveshare.com/nrf51822-eval-kit.htm

4 Spécifications

Sujet: Mettre en place des attaques sur le protocole *Bluetooth Low Energy* (Bluetooth Smart)

Target: BLE 4.0 dit legacy pas de LESC avec methode d'appairage JustWorks Pourquoi 4.0 ? - Car 4.2 LESC ECCDH bcp plus difficile a craquer et bcp moins d'outils pour le faire - beaucoup d'appareils sont encore en 4.0 legacy ou meme sans pairing Pourquoi JustWorks ? - target IOT donc objets simples et peu chers (capteurs) n'embarquant pas d'autres moyens d'auth

Meme si mecanisme securite offert par BLE avec appairage niveau lien, bcp de constructeur utilisent mecansimes custom niveau application bases sur les standard de crypto (AES) et methodes comme challenge-response. Clef flashee dans l'appareil BLE et distribuee au controller via un serveur distant par une application mobile lors de la creation de compte/identification de l'appareil BLE pour la premiere fois.

Appareil transportable d'attaques sur les appareils BLE nearby ex: Raspi avec dongles et/ou sniffer BLE (BBC Micro) faisant tourner container DOcker avec le Poc dessus

4.1 Fonctionnalites

- Inventaire ..
- ...
- ...
- ...

4.2 Architecture

serveur flask + websockets application js (hyperapp) + socketio framework offensif Mirage (python) + bindings custom pour communication API et non CLI

4.3 Interface

GUI web HTML + JS

4.4 Tests

Mock reseau BLE avec outils mirage slave + master

4.5 Fonctionnalités

La preuve de concept devra fournir plusieurs fonctionnalités offensive décritent ci-après.

Repérage

Inventaire des appareils et connexions BLE à proximité.

- Écoute des annonces sur les 3 canaux publicitaires pour récupérer les appareils émetteurs.
- Écoute des communications sur les 37 canaux de données pour répertorier celles active.

Localisation

Localisation des appareils BLE alentours.

- Écoute passive des annonces pour extraire le calibrage du signal et calculer la distance à partir de la puissance du signal reçu.
- Si le calibrage n'est pas émit dans l'annonce, établissment d'une connexion pour récuperer la valeur si disponible.

Opération répétables autant de fois que voulu pour améliorer la précision de la localisation (minimum 3 mesures pour une position).

Identification

Connexion directe à un appareil via son adresse bluetooth pour extraire toutes les données exposées.

- Écoute optionnelle des annonces pour identifier un esclave cible.
- Requête de connexion à la cible en tant que maître.
- Récupération des informations standardisées (GAP/GATT) ainsi que services et attributs propriétaires.

Interception

Interception de communications et possible déchiffrement des trames.

- Écoute des communications sur les 37 canaux de données.
- Récupération de l'adresse d'accès et des paramètres d'appairage (carte des canaux, temps et nombre de sauts, etc).
- Synchronisation avec la communication et écoute des trames.
- Si la communication est chiffrée et la phase d'appairage passée, déconnexion des appareils via brouillage des communication jusqu'au temps mort.
- Écoute des canaux d'annonce: attente d'un appairage en supposant qu'il provienne des appareils precedement déconnectés.
- Récupération des informations cryptographique pour déchiffrer la connexion seulement si celle-ci n'utilise pas une clef a long terme deja établie ou une connexion securisée (BLE 4.2).
- Écoute des communications et déchiffrement des trames à la volée.

Modification

Attaque *man in the middle* par clonage et usurpation d'un appareil BLE pour modifier les données echangées.

- Écoute passive des annonces de l'esclave cible de l'usurpation pour retransmission ultérieur et récupération de l'adresse bluetooth.
- Connexion à l'esclave cible d'usurpation pour qu'il n'émette plus d'annonces.
- Changement de l'adresse de l'usurpateur en celle de l'esclave usurpé et réémission des annonces précédement capturées.
- Attente de la connexion du maître.
- Appairage entre l'usurpateur et le maître.
- Retransmission des communications entre le maître et l'esclave par l'usurpateur.

Il sera par la suite envisageable d'associer plusieurs fonctionnalités pour réaliser des scénarios différents. Ce peut être par exemple l'usurpation d'un appareil suite au brouillage lors de l'interception des communications entre 2 appareils.

4.6 Architecture

Le système se compose d'un front-end fournissant une interface utilisateur affichant les appareils BLE et les actions possible ainsi qu'un back-end permettant la réalisation des actions implementées.
Le back-end se compose d'un service web (en violet sur fig. 4) pour communiquer avec le front-end, il transmet les requêtes au serveur (en rouge) qui se base sur un framework BLE offensif (en bleu) pour les traiter. Le framwork BLE offensif utilise plusieurs appareils BLE (en vert) pour mener à bien les attaques.
Le serveur orchestre les attaques même si il ne les implémentent pas lui-même.

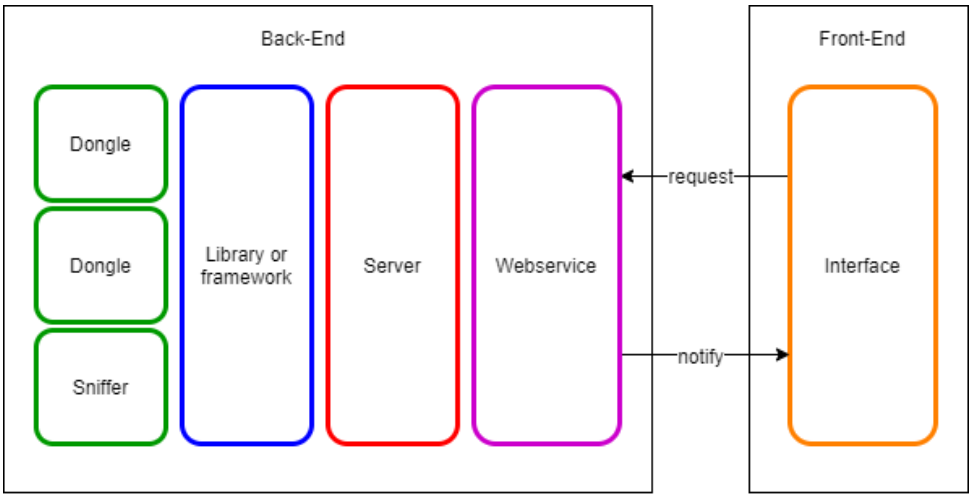

Figure 4: Architecture du système

4.7 Interface

On retrouve la carte des appareils et connexions identifiés avec leur distance et position estimée par rapport au système (voir fig. 5: zone rouge *Scan*).

Pour chaque cible (appareil ou connexion), des attaques sont disponibles: - Récupération du profil ou modification des transimissions par usurpation pour un appareil BLE emettant des annonces (zone bleue *Devices*). - Déconnexion des appareils ou interception des communications entre deux appareils appairés (zone bleue *Connections*).

Une troisieme section permet de suivre le déroulement de l'attaque chosie (zone verte *Action progress*). Celle-ci est découpée en phases, dès que la phase courante est terminée sans erreur (carré vert), la phase suivante est exécutée. Lorsqu'une phase échoue l'attaque s'arrête et le message d'erreur est affiché en dessous (carré rouge).

4.8 Tests

Il est possible de tester toutes les attaques en mettant en place un réseau BLE de test. Toutes les attaques ne ciblent jamais plus de 2 appareils BLE. Il est possible de reproduire les conditions attendues dans l'attaque en imitant un esclave et un maître BLE avec

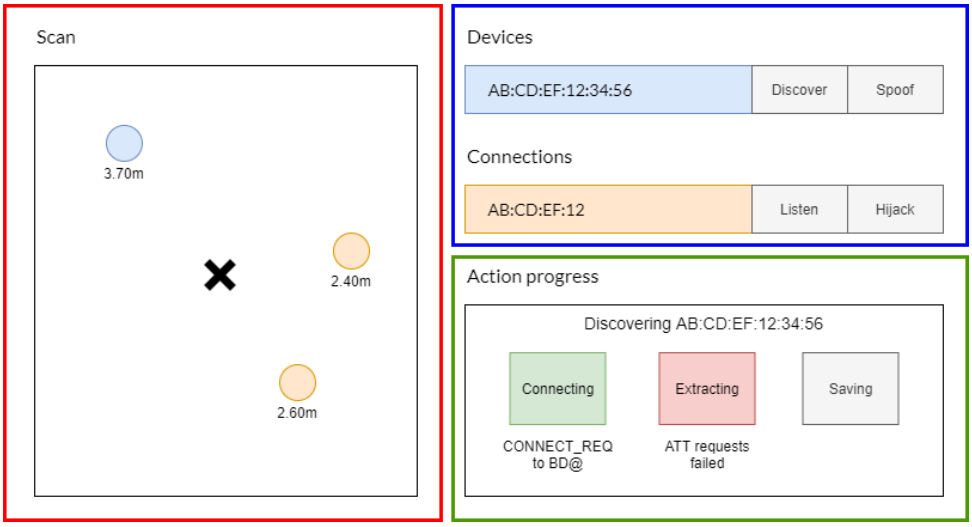

Figure 5: Interface du système

des requêtes et réponses préprogrammées. Sur chaque attaque demande des conditions de départ différentes, les appareils peuvent être en attente (émettant des annonces), en appairage ou connectés.
Une fois notre réseau test mis en place, l'attaque est executée sur celui-ci et les résultats obtenus comparés par rapport à ceux préprogrammés dans le test.

Il est possible d'automatiser ces tests avec 5 appareils (4 dongles et 1 sniffer) branchés à la machine réalisant ceux-ci. Le sniffer réalise la plupart des tâches purement offensive, 2 dongles mettent en place le réseau test pendant que les 2 autres permettent l'usurpation d'identité.

4.9 Livrables

Code source du système fonctionnel: comprend l'intégration de l'outils offensive, le serveur et client pour l'interface ainsi qu'un moyen de déployer le système (Docker).

Documentation du système: rédigée en langage spécifique (markdown, rst) et déployable avec un outils (Sphinx, pandoc), documentation développeur pour mettre en place le système et documenter les choix techniques.

Rapport de projet: rédigé avec un outils spécifique (LaTeX, pandoc), rendue au format PDF, comprend une étude du contexte, analyse de l'existant et de faisabilité puis mise en place de la preuve de concept.

5 Preuve de concept

5.1 Scan

Adv ok seulement 3 channels, utilisation du sweeping sur 1 micro bit

Data difficile car 37 channels et transmissions non constantes dans chaque channel, meme avec sweeping sur 1 micro bit ne peut intercepter que 1/37e des communications, jeu chat et souris car appareils hop et bbc sweep pour trouver des comms

5.2 Localisation

TODO differentes methodes de localisation - avoir une distance (rssi / toa) - avoir un point dans l'espace (aoa/aod ou trilateration/triangulation)

RSSI

incertitude rssi +-6dbm et fortement influence par environnement

peu etre reduit avec echantillonage sur le temps, modele de calculs et filtres (kalmann)

BLE utilise plusieurs puissance emissions donc besoin d'une valeur ref pour estimer distance depuis RSSI. Valeur generalement RSSI mesure a 1m par le constructeur et exposee dans les annonce ou en tant que service et nommee txpower (standardisee par GAP/GATT).

TODO formule distance env factor = 2 pour IPS

Fingerprinting

A partir d'une liste de beacons et leurs position, calcul la position se rapprochant le plus d'un des beacons (a partir du RSSI).

Demande de pouvoir etablir la liste des beacons et les identifies de facon sure. Si le systeme est mit en place pour cet effet on s'assurera qu'ils soient identifiables (MAC unique par exemple) mais dans notre cas de recuperation d'information, les appareils peuvent mettre en place des mesures contre le tracage comme la generation d'adresse mac aleatoire. Il est possible d'utiliser le profile GATT pour identifier un appareil, combiner avec le RSSI dans le temps et les deplacements (capteurs) on peut esperer distinguer deux profils GATT identiques.

~ beacons coverage

Le beacon le plus proche

RSSI / TOA

~ m

Trilateration determines the position of an object by understanding its distance from three known reference points. In the case of Bluetooth, locators estimate their distance to any given asset tag based on the received signal strength from the tag

AOA / AOD

~ cm

Basee sur le nouveau systeme d'angle du BLE 5.1 Demande du materiel en plus (Multiple antennes directionnelles pour former une matrice) Differentes facon de calculee (angle arrivee, angle depart . . .)

https://www.bluetooth.com/blog/bluetooth-positioning-systems/ https://www.bluetooth.com/bluetooth-resources/enhancing-bluetooth-location-services-with-direction-finding/?utm_campaign=location-services&utm_source=internal&utm_medium=blog&utm_content=bluetooth-positioning-systems

Ajouter de la precision

Fusionner les resultats avec un filtre kalmann: - dead reckoning - trilateration / triangulation

Ou RSS (range) + AOA (direction)

RSS

1. Scan devices BTLEJack sniffer
2. find settings (rssi, txPower / measured power . . .) Tx Power service 0x1804 and Tx Power Level Characteristic 0x2A07
3. calculate distance (in a circle around you) 10 ^ ((txPower – RSSI)/(10 * N)) N = loss factor (between 2 and 4), 0 for optimal conditions
4. cross multiple references to determine a position (trilateration) repeat 3 times to 3 devices get OUR position

AOA

5.3 Spoof

5.4 Hijack