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

1 Objets connectés **6**
1.1 Architecture . . . 6
1.2 Protocoles . . . 6
1.3 Marché . . . 6

2 Bluetooth Low energy **7**
2.1 Différences . . . 7
2.2 Protocole . . . 7
2.3 Versions . . . 13

3 Vulnerabilites **14**

4 Outils offensif **15**
4.1 Logiciels . . . 15
4.2 Materiels . . . 15

5 Mirage **16**
5.1 Presentation . . . 16
5.2 Integration . . . 16

6 Spécifications **17**
6.1 Fonctionnalités . . . 17
6.2 Architecture . . . 18
6.3 Interface . . . 19
6.4 Tests . . . 19
6.5 Livrables . . . 19

7 Preuve de concept **21**
7.1 Sniffing . . . 21
7.2 Localization . . . 21
7.3 MITM . . . 22

List of Tables

1 Cas d'utilisation et protocole Bluetooth adapté 7
2 Capacités d'entrée possibles 10
3 Capacités de sortie possible 10
4 Capacité d'entrées/sorties de l'appareil 10
5 Méthode d'appairage utilisée en fonction des capacités échangées (JW=JustWorks PK=PassKey NC=NumComp) 11

List of Figures

1 Répartition du spectre BLE en canaux 8
2 Étapes d'un échange BLE 9
3 Client et serveur GATT 13
4 Architecture du système 18
5 Interface du système 19

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. Même si intégré à la spécification Bluetooth, il est incompatible avec les autres variantes de celui-ci.
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 repondre 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 réaliser une action non voulue (ouverture de cadenas).

1 Objets connectés

Avec l'explosion de l'internet de objets (TODO chiffres) toute une floppée d'objet du quotidien ont étés augmentés pour permettre la communication avec d'autres systemes 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.
Cependant 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.

TODO securite plus en plus pris en compte mais secondaire tj

1.1 Architecture

Point à point

Architecture reseau domotique - simple: appareil non relie au reseau, dependant gateway utilisateur, remplissant une fonction d'augmentation seul (smart lock)

Réseau

- avancee: appareil s'appuyant sur un reseau domotique pour realiser ses fonctions, relie a une gateway "sure" hub

1.2 Protocoles

Protocoles generaux supportes par tout appareil (smartphone notamment) et peu cher
WiFi (WLAN) ~50m: Local = remplace cables pour appareils fixes dans pieces / appart
BLE (WPAN) ~10m: Personnal = remplace cables pour appareils portable personnels NFC

Protocoles specifiques concus pour ces reseaux Zigbee Zwave Thread ANT(+)

1.3 Marché

Premiere generation point a point "smart"

Seconde generation networks IoT

BLE

Gadgets (fitness)

Domotique (IoT)

Entreprise / warehouse / smart city (beacons)

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).

Table 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.

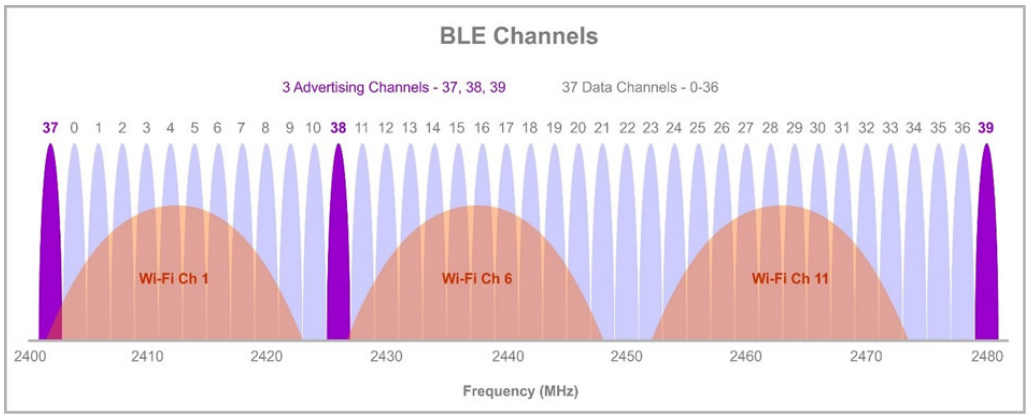

Figure 1: Répartition du spectre BLE en canaux[1]

Couche logique

1. Annonces

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

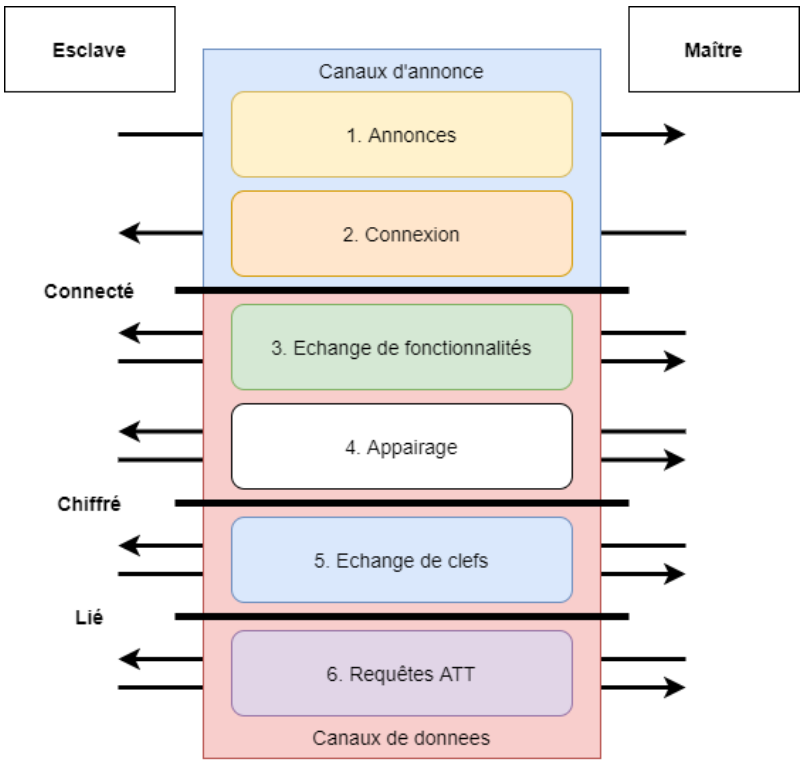

Figure 2: Étapes d'un échange BLE

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*[2]). 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[3]. 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[4].

2. Connexion

[2]https://www.bluetooth.com/specifications/assigned-numbers/generic-access-profile/
[3]https://www.bluetooth.com/specifications/gatt/services/
[4]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*).

Table 2: Capacités d'entrée possibles[5]

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*

Table 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

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

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

[5]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).

Table 5: Méthode d'appairage utilisée en fonction des capacités échangées[6] (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

[6] 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).

Communication

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.[7]

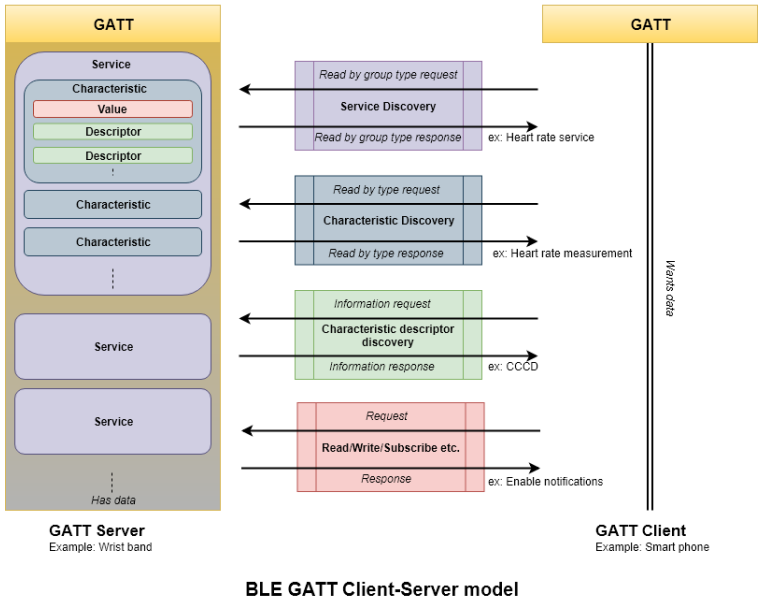

Figure 3: Client et serveur GATT[8]

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*).

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

3 Vulnerabilites

COnfidentialite Appairage

Authentification Appairage

...

4 Outils offensif

4.1 Logiciels

Etude des communications bluetooth: Wireshark Scappy etc Possible avec n'importe quel chip Bt deja sur la machine ou dongle USB pour une etude du traffic interne et des appareils emettants des adv.

Interceptions des communications - BTLE (C) - BTLEJack (lib python + firmware C) - Mirage (framework python)

Proprietaires: - nRF sniffer - nRF Connect - smartRF (TI)

Attaques - GATTacker (NodeJS) MiTM - BTLEJuice (NodeJS) MiTM - BTLEJack (Jamming/ Hijacking) - Mirage (MiTM / jam / hijack / crack)

4.2 Materiels

We can BLE dedicated devices to sniff or modify it. Internal Bt chips can only adv or connect to peripherals but never scan or modify it. They only see internal traffic (locked firmware).

Full featured HackRF PandwaRF Ubertooth

BLE HCI Dongle nRF52840 (https://www.nordicsemi.com/Products/Low-power-short-range-wireless/nRF52840) - https://www.nordicsemi.com/Software-and-tools/Development-Kits/nRF52840-Dongle Some using CSR8510 (https://www.qualcomm.com/products/csr8510) - Adafruit Bluetooth 4.0 USB Module (https://www.adafruit.com/product/1327) - https://www.amazon.co.uk/CSR8510-Bluetooth-Adapter-Classic-Headset/dp/B01G92CNY8

Qualcomm, Broadcom, Realtek, NordicSemiconductor ... Featured in documentation is Qualcomm one

Sniffer - Ubertooth One ($$) - BTLEJack BBC Micro:Bit, Bluefruit, Waveshare BLE400, nRF51822 Eval kit (tweak) (https://github.com/virtualabs/btlejack) - Bluefruit https://www.adafruit.com/product/2269 (limited) - nRF51 https://www.nordicsemi.com/Software-and-tools/Development-Kits/nRF51-Dongle (close) - TI CC2540 USB Dongle BLE sniffer (http://www.ti.com/tool/CC2540EMK-USB) - Crazy Radio PA 2.4GHz (https://store.bitcraze.io/collections/kits/products/crazyradio-pa)

Board - HackRF - PandwaRF

5 Mirage

5.1 Presentation

5.2 Integration

Si je me concentre sur Mirage, cela restreint pas mal les outils possible: - dongle BLE HCI standard - sniffer BLE adaptable avec BTLEJack (micro:bit, bluefruit, ble400, nRF51) Les appareils dépendent des besoin, dans mon cas il me faudrait: - inventaire: Sniffer (BTLEJack) - obtention d'informations (crack, mit): dongle HCI x2 (un slave et un master, a voir si un BTLEJack peut remplacer un HCI) - localisation / tracking (rssi + autres méthodes): Mirage ne permet pas cela nativement mais les informations demandées doivent être récupérables dans le framework pour l'implementer manuellement (RSSI, angle antenne ?). Cela demande au minima un dongle HCI, meme si les travaux trouvés sur le sujet utilisent un sniffer Bluefruit. Dans les travaux étudiés, la localisation demande 3+ appareils BLE pour permettre la trilatération

Il me manque donc a voir si un BTLEJack peut remplacer un HCI dans l'attaque MITM, ainsi que trouver des informations pour implementer la localisation IPS avec Mirage. Mirage supporte également d'autres appareils (comme Ubertooth) mais leurs fonctionnalités ne nous sont pas nécessaires, un sniffer flashé avec BTLEJack suffit (et coute moins cher). Pour les sniffers BTLEJack eligibles: - Bluefruit et nRF51 (~20e) demandent reprogrammation via un "external SWD" (assez cher + 100e) - la carte BBC Micro:bit (20e, non vendue en France directement) permet une reprogrammation sans appareil supplémentaire, et semble donc la plus simple

Pour résumer: - dongle BLE (https://www.adafruit.com/product/1327 / https://www.amazon.co.uk/CSR8510
Bluetooth-Adapter-Classic-Headset/dp/B01G92CNY8) - carte Micro:Bit (https://microbit.org/buy/)

6 Spécifications

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

6.1 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.

6.2 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.

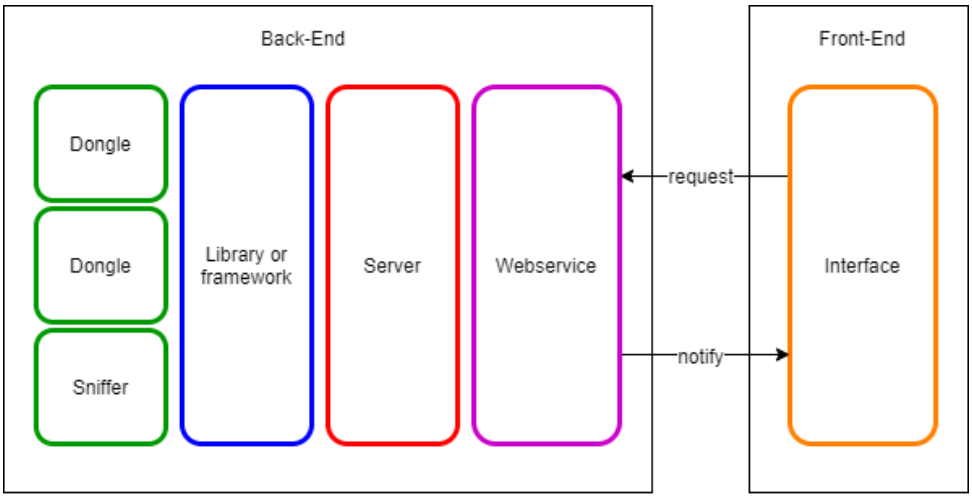

Figure 4: Architecture du système

6.3 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).

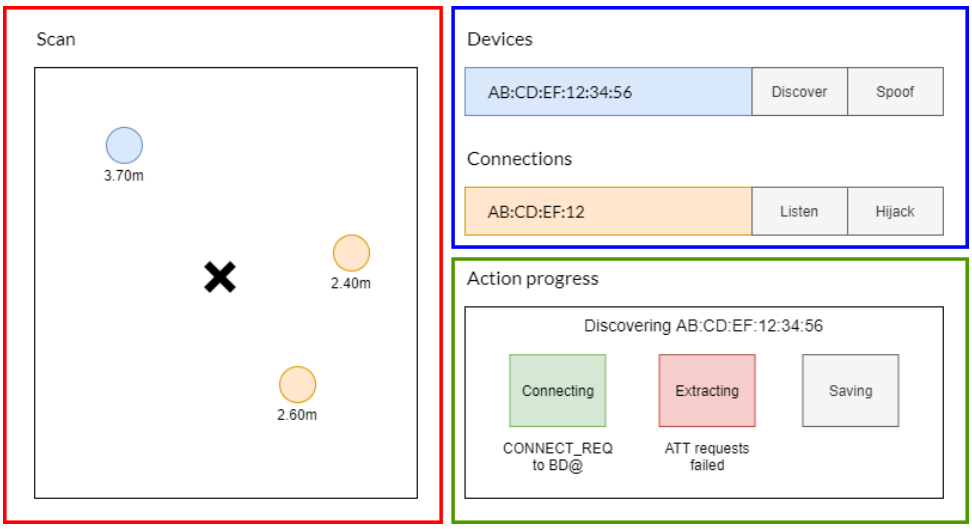

Figure 5: Interface du système

6.4 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 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.

6.5 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.

7 Preuve de concept

7.1 Sniffing

7.2 Localization

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

7.3 MITM