



UNIVERSITÉ
DE LORRAINE

UFR MATHÉMATIQUES INFORMATIQUE
MÉCANIQUE ET AUTOMATIQUE

Sniffing Bluetooth

présenté par

Anthony Ly

François Panisset

Sécurité des Systèmes et des Réseaux

Proposé et encadré par A. Dulaunoy

Sommaire

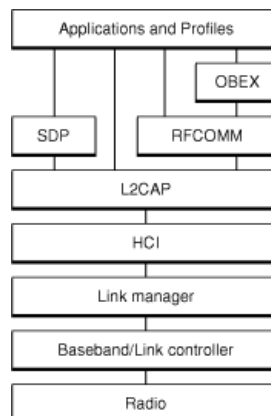
Présentation du Bluetooth	4
Bluetooth Low Energy	14
Expérience	15

Présentation du Bluetooth

Bluetooth est un standard de communication sans fil à faible portée et est basé sur des ondes radio d'une fréquence de 2,4 GHz. Il a été créé afin de simplifier les communications entre les appareils et donc de supprimer les connexions filaires. Il concerne essentiellement des petits appareils physiques portables, comme les téléphones cellulaires, les écouteurs sans fil, les assistants numériques personnels (PDA). L'avantage de cette technologie est qu'elle est également peu coûteuse et peu consommatrice d'énergie. Toutefois, comme d'autres protocoles de communication, il existe de nombreuses failles de sécurité. Dans la suite de ce rapport, nous allons l'étudier en détail, puis voir les différents types d'attaques de Bluetooth et enfin nous verrons les différents mécanismes permettant de se prémunir à ces attaques.

L'architecture Bluetooth

Bluetooth dispose de nombreuses couches contrôlant plusieurs aspects que nous allons décrire ci-dessous.



Architecture de Bluetooth

1) La couche radio

La couche radio, aussi appelée couche RF (Radio Fréquence), est la couche la plus basse. Elle permet l'émission et la réception des ondes radio. Elle dispose également de caractéristiques concernant la bande de fréquence, l'arrangement des canaux, du transmetteur, du récepteur et bien d'autres.

Cette bande de fréquence est située entre 2400 et 2483,5 MHz.

79 canaux sont numérotés et sont séparés de 1 MHz, commençant à 2402 MHz, permettant le codage de l'information.

2) La couche bande de base (Baseband)

Les adresses matérielles des dispositifs Bluetooth sont définies dans cette couche et sont codées sur 48 bits. Ces adresses sont comparables aux adresses MAC d'une carte réseau et sont appelées BD_ADDR, représentant *Bluetooth Device Address*.

De plus cette couche gère les connexions entre deux appareils, qui peuvent être soit synchrones ou soit asynchrones.

- Les liaisons SCO (Synchronous Connection-Oriented)

Ces connexions permettent une transmission bidirectionnelle. Elles sont utilisées pour des connexions concernant la transmission de voix.

- Les liaisons ACL (Asynchronous Connection-Less)

Ces connexions sont utilisées pour l'échange de données.

3) La couche Link Manager

Cette couche contrôle les connexions. C'est également elle qui implémente des systèmes de sécurité comme :

- L'authentification
- Le pairing
- La création et la modification des clés
- Le chiffrement

4) L'interface de contrôle (HCI)

L'interface de contrôle (Host Controller Interface) permet de relier les couches matérielles aux couches logicielles. La séparation entre ces deux couches permet le développement indépendant du matériel et du logiciel.

HCI dispose de commandes et d'évènements qui sont utilisés lors de la communication entre le périphérique et la puce Bluetooth. Le périphérique envoie des commandes à la puce et reçoit des évènements en retour.

5) La couche L2CAP

L2CAP (Logical Link Control & Adaptation Protocol) permet de créer un lien entre les données, qui sont sous forme de paquets, et la couche Baseband. Elle est donc capable de multiplexer les protocoles de niveau supérieur, de segmenter et de réassembler les paquets. Elle est également à la base de la connexion entre deux appareils, à l'aide de canal.

Les services Bluetooth

SDP (Service Discovery Protocol) : ce protocole permet aux dispositifs Bluetooth de découvrir l'environnement, c'est-à-dire qu'il permet de détecter d'autres appareils Bluetooth et de connaître les services qui sont accessibles.

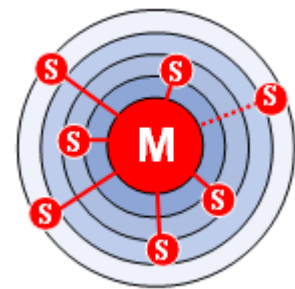
RFCOMM (Radio Frequency Communications Protocol) : RFCOMM émule les ports série de RS-232 sur L2CAP. Il permet le transport des paquets et est donc largement sollicité dans les autres protocoles de niveau supérieur.

OBEX (Object Exchange protocol) : permet aux périphériques d'échanger des fichiers, dits standards. Ces fichiers sont généralement des fichiers de données, des cartes de visite, ou encore le calendrier. Obex est une application de couche haute utilisant RFCOMM.

Les réseaux Bluetooth

1) Le piconet

Un piconet, picoréseau en français, désigne le réseau établi entre des appareils Bluetooth. Ce concept est basé sur une architecture maître-esclaves. Le maître peut être connecté à 7 périphériques actifs ou 255 périphériques inactifs en même temps.



➤ Le maître

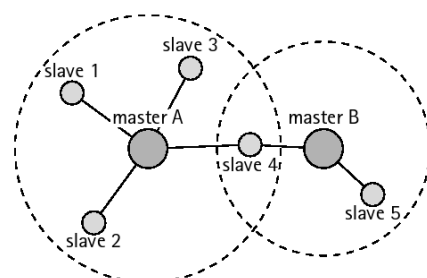
Le maître doit être capable de gérer un certain nombre de caractéristiques pour le bon fonctionnement du piconet. En effet, 3 aspects doivent être fixés : l'horloge, la séquence de saut de fréquence et le code d'accès de la liaison.

➤ Les esclaves

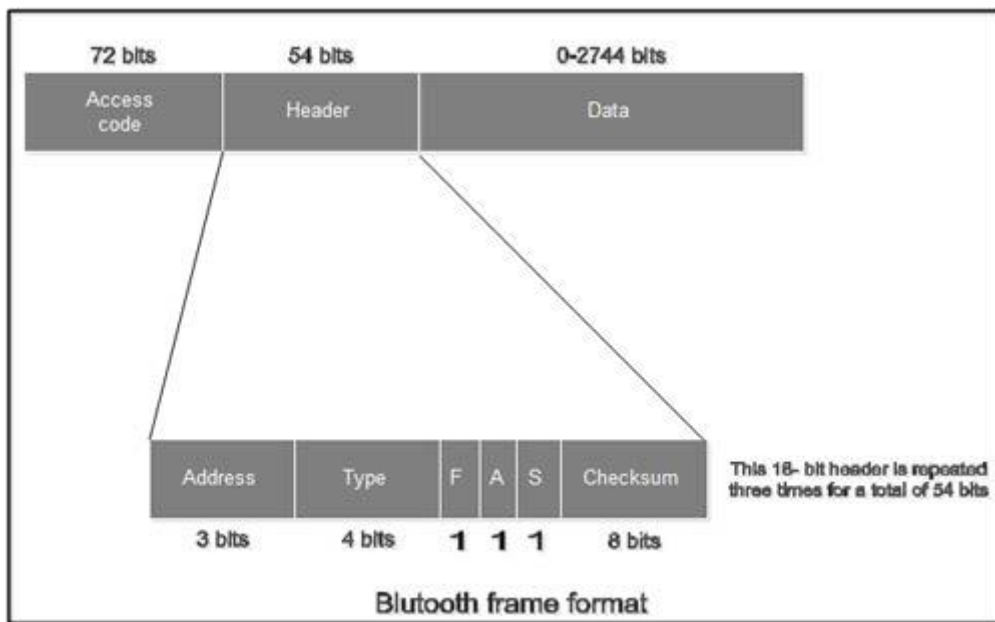
Ce sont des dispositifs qui communiquent avec le maître et ne peuvent pas communiquer entre eux. Ils sont synchronisés sur l'horloge du maître.

2) Le scatternet

Un scatternet est l'interconnexion de plusieurs piconet. Cette structure est possible car un esclave peut avoir plusieurs maîtres.



Format d'une frame Bluetooth



Format d'une frame Bluetooth

- Le code d'accès contient les 72 bits de synchronisation et permet d'identifier le piconet et donc le maître.
- Le Header contient un champ de 18 bits qui est répété 3 fois. Ce champ contient l'adresse, le type de données, des flags (F,A,S) et un checksum pour la détection d'erreur.
 - F** (FLOW) : Bit de contrôle de flux
 - A** (ARQN) : Bit d'accusé de réception
 - S** (SEQN) : Bit de numéro de séquence
- Le champ de donnée correspond aux données à transmettre.

Le jumelage

Afin que deux dispositifs Bluetooth puissent communiquer entre eux, il faut tout d'abord qu'ils se jumèlent. Pour se faire, il est nécessaire de passer par plusieurs étapes pour établir la connexion.

1) Découverte de l'environnement

Cette étape est également appelée *phase d'inquisition*, phase durant laquelle le maître réalise un broadcast vers les périphériques à proximité. Ces périphériques présents dans la zone répondent alors avec leur adresse BD_ADDR.

2) Synchronisation avec le dispositif choisi

Après avoir choisi un dispositif, le maître synchronise, avec ce dernier, son horloge et sa fréquence. Ce procédé est également appelé le *paging*.

3) Phase SDP

La phase SDP permet au maître de découvrir les services que l'esclave peut fournir.

4) Création du canal de communication

En utilisant L2CAP, le maître peut alors créer un canal de communication.

5) Pairage

Si l'esclave dispose d'un mécanisme de sécurité, un code PIN est alors nécessaire pour confirmer la connexion. L'esclave envoie une demande au maître, pour que celui-ci saisisse le code. Le code PIN est basé sur une clé de chiffrement.

La sécurité

1) Les modes de sécurité

Bluetooth dispose de 3 modes de sécurité.

Mode 1 : Aucune sécurité. Ce premier mode ne protège en aucun cas l'appareil. Toute connexion peut être établie avec le dispositif.

Mode 2 : Sécurité au niveau applicatif. Il entre en compte après l'établissement de la connexion. Il permet d'instaurer des politiques de sécurité qui auront pour but de protéger l'appareil au niveau applicatif.

Mode 3 : Sécurité au niveau liaison. Des mécanismes de sécurité sont établis dès la connexion. En effet, une authentification avec chiffrement est nécessaire pour permettre la connexion entre deux appareils.

2) Les clés

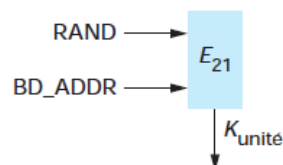
Pour gérer la sécurité, Bluetooth dispose de plusieurs clés. La création de ces clés se basent sur différents paramètres :

- L'adresse BD_ADDR de 48 bits
- Une clé privée d'authentification qui est un nombre aléatoire sur 128 bits
- Une clé privée de 8 à 128 bits pour le codage
- Un nombre aléatoire qui est modifié régulièrement par le dispositif Bluetooth

Clé de liaison : elle correspond à un nombre aléatoire sur 128 bits et est utilisée pour chaque échange, dans une communication sécurisée, entre deux appareils. Elle est également utilisée lors de l'authentification et sert à créer la clé de cryptage.

Cette clé peut être assimilée par l'une des 4 autres clés définies dans Bluetooth :

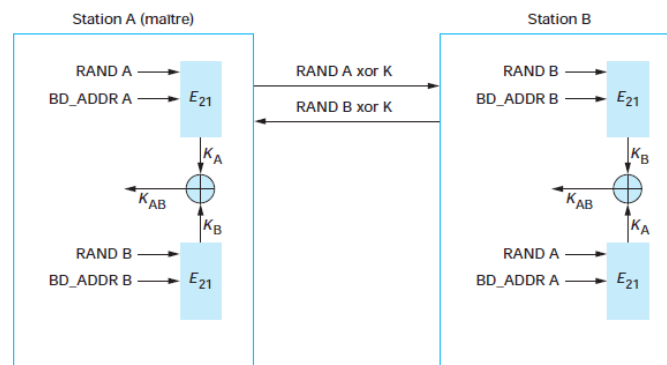
- **Clé Unit** : elle est créée, à l'aide de l'algorithme E_{21} , lors de l'établissement de sa première communication Bluetooth.



Génération de la clé Unit

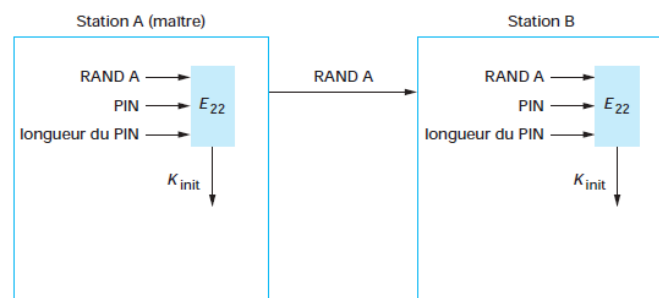
- **Clé maître** : c'est une clé temporaire, générée avec l'algorithme E_{22} et 2 nombres aléatoires de 128 bits. Elle est utilisée lorsque le maître souhaite envoyer des informations à plusieurs esclaves.

- **Clé de combinaison** : elle est créée à partir des informations des deux appareils, par le biais de l'algorithme E_{21} .



Génération de la clé de combinaison

- **Clé d'initialisation** : elle est utilisée lorsqu'il n'y a l'existence d'une clé de liaison ou d'une clé de combinaison. Sa création s'effectue lors de l'initialisation de la communication et résulte d'une opération entre l'algorithme E_{22} , le code PIN, le BD_ADDR de l'autre appareil et d'un nombre aléatoire de 128 bits.

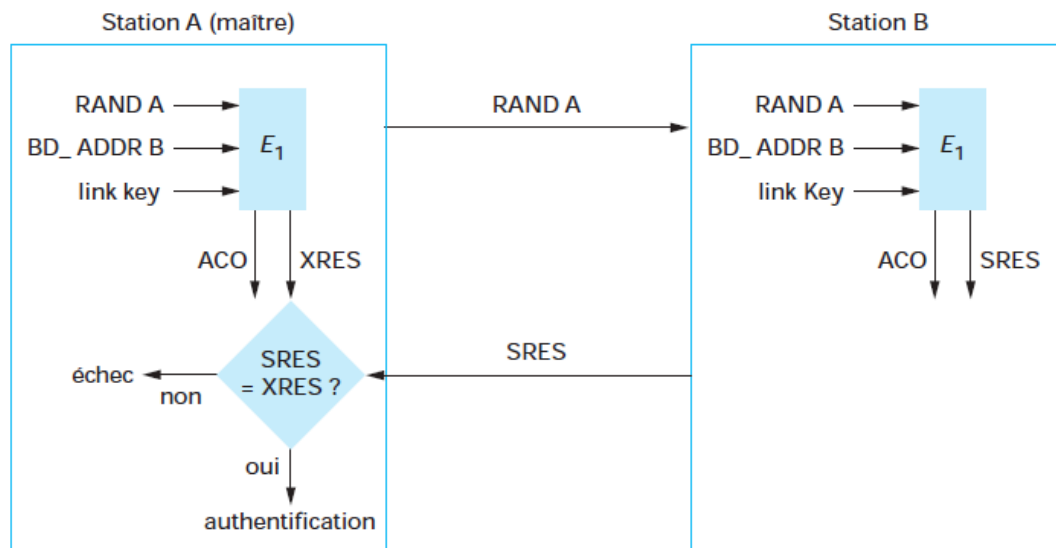


Génération de la clé d'initialisation

3) Authentification

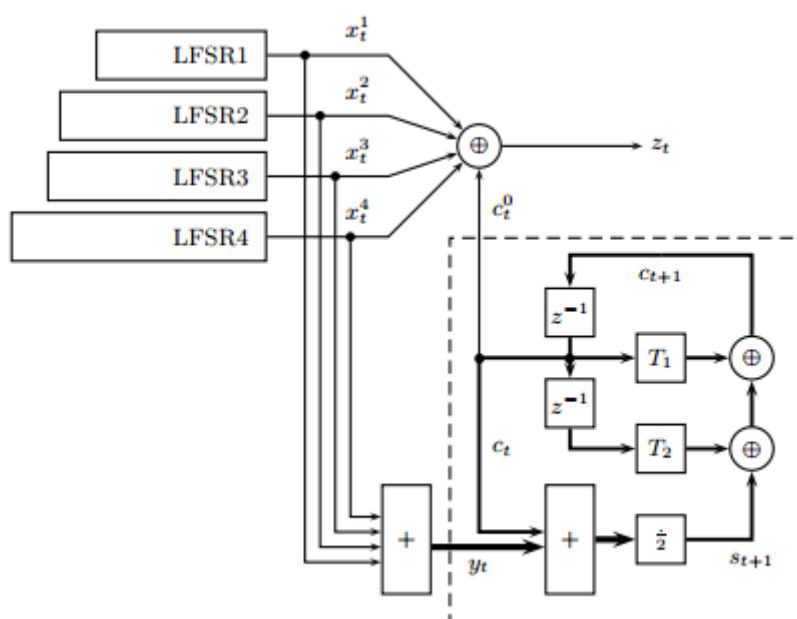
L'authentification est la procédure de vérification d'un autre appareil Bluetooth. La procédure entre un appareil maître A et un appareil B est la suivante :

- A envoie un nombre aléatoire à B
- A et B calculent respectivement XRES (eXpected Result) et SRES (Signed Result) à l'aide de la fonction d'authentification E_1 . La fonction d'authentification E_1 prend en paramètre le nombre aléatoire de la première étape, la BD_ADDR de B et de la clé de liaison.
- B envoie alors son SRES à A
- A vérifie que $XRES = SRES$. Si c'est le cas, alors B possède la clé de liaison correspondante à A. L'authentification est réussie.



Procédure d'authentification

4) Le chiffrement par flux E0



Générateur de séquence de clé E0

Ce générateur de séquence de clé est constitué de 4 registres à décalage à rétroaction linéaire (linear feedback shift registers ou LFSR) dont les sorties sont x_t^i et d'une machine constituée de plusieurs autres opérations, ce générateur est initialisé avec une clé d'au plus 128 bits.

La spécification des LFSR sont listées dans le tableau suivant, on notera que la somme des longueurs des LSFR est de 128.

LFSR	Longueur	Fonction de rétroaction
1	25	$1 + D^8 + D^{12} + D^{20} + D^{25}$
2	31	$1 + D^{12} + D^{16} + D^{24} + D^{31}$
3	33	$1 + D^4 + D^{24} + D^{28} + D^{33}$
4	39	$1 + D^4 + D^{28} + D^{36} + D^{39}$

L'état de la machine est déterminé par 4 bits stockés sous la forme d'une paire de 2-bits élément que l'on peut respectivement écrire c_{t-1} et c_t . Le nouvel élément c_{t+1} est calculé de la manière suivant :

$$c_{t+1} = s_{t+1} \oplus T_1(c_t) \oplus T_2(c_{t-1})$$

Avec

$$y_t = x_t^1 + x_t^2 + x_t^3 + x_t^4$$

$$s_{t+1} = \frac{y_t + c_t}{2}$$

T_1 et T_2 sont deux bijections linéaires sur \mathbb{Z}_2^2 définies selon :

$$T_1 : \mathbb{Z}_2^2 \rightarrow \mathbb{Z}_2^2, (x_1, x_0) \mapsto (x_1, x_0)$$

$$T_2 : \mathbb{Z}_2^2 \rightarrow \mathbb{Z}_2^2, (x_1, x_0) \mapsto (x_0, x_1 \oplus x_0)$$

Enfin, les bits de séquence de clé z_t sont calculés de la façon suivant :

$$z_t = x_t^1 \oplus x_t^2 \oplus x_t^3 \oplus x_t^4 \oplus x_t^0$$

Le chiffrement est effectué en faisant un ou-exclusif entre la séquence z_t et la séquence de bits des paquets de données. La taille maximum d'un paquet est 2745 bits, un fois qu'un paquet est émis, le générateur est réinitialisé.

Bluetooth Low Energy

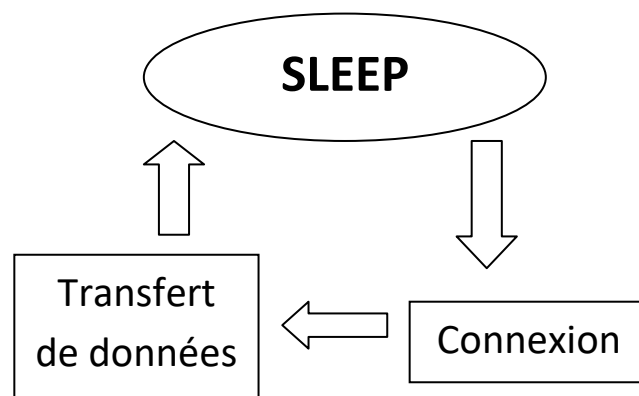
Bluetooth LE ou Bluetooth Low Energy, également appelé Bluetooth Smart, est un dérivé du Bluetooth dit « classique ». En effet, celui-ci a été conçu de manière à moins consommer d'énergie, augmentant ainsi la croissance des périphériques d'Internet of Things. Ces derniers tendent à travailler non pas comme un maître, mais plus comme des esclaves qui s'occupent de collecter les données.

Plusieurs systèmes d'exploitation mobile supportent nativement le BLE.

Les changements apportés par BLE

Comme dit plus haut, BLE permet aux appareils de moins consommer d'énergie. En effet, cela est possible grâce au fait que ces derniers restent dans un "sleep mode", à moins que la connexion soit établie pour un transfert de données.

Ce comportement peut se schématiser comme suit :



Technologie BLE

De ce fait, BLE est idéal pour des appareils nécessitant seulement d'une courte période de connexion et non pas des appareils qui ont besoin d'émettre un flux constant.

Exemple d'appareils pour BLE : éclairage, domotique, ...

Exemple d'appareils pour Bluetooth classique : enceintes, casques, écouteurs, ...

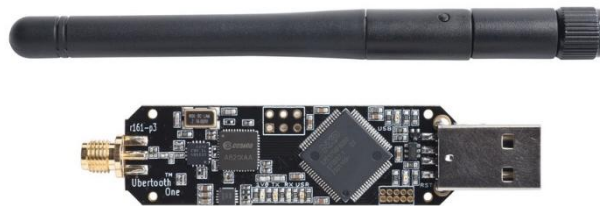
Expérience

Cette expérience a pour but de réaliser l'écoute (sniffing) d'une communication Bluetooth entre deux appareils utilisant du Bluetooth low energy (BLE) à l'aide d'un Ubertooth One (Project Ubertooth) et de Wireshark.

Project Ubertooth est une plateforme de développement open source créée en 2010 pour effectuer des expérimentations sur le Bluetooth, ils ont notamment développé l'Ubertooth qui agit comme un récepteur/émetteur Bluetooth doté d'un firmware permettant par exemple du sniffing en mode passif.

Pour cette expérience, deux sniffing ont été effectués, le premier dans le bâtiment de notre université où on suppose la présence de communication bluetooth et le deuxième dans une maison avec uniquement quelques appareils bluetooth allumés.

Un guide d'installation et le protocole d'expérimentation sont présents dans l'annexe de ce rapport.



Ubertooth one

Première expérience

La premier sniffing est celui réalisé à l'université, la capture complète est présente sur le répertoire GitHub sous le nom de *capture_univ.pcapng*, en voici un extrait :

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10:4f:f0:47:20:bc	Broadcast	LE LL	70	ADV_NONCONN_IND
2	0.002800400	3c:42:15:ea:6a:37	Broadcast	LE LL	70	ADV_NONCONN_IND
3	0.075534500	05:d0:da:52:56:53	Broadcast	LE LL	70	ADV_NONCONN_IND
4	0.102519000	10:4f:f0:47:20:bc	Broadcast	LE LL	70	ADV_NONCONN_IND
5	0.104068400	3c:42:15:ea:6a:37	Broadcast	LE LL	70	ADV_NONCONN_IND
6	0.184925500	05:d0:da:52:56:53	Broadcast	LE LL	70	ADV_NONCONN_IND
7	0.209083600	3c:42:15:ea:6a:37	Broadcast	LE LL	70	ADV_NONCONN_IND
8	0.210038800	10:4f:f0:47:20:bc	Broadcast	LE LL	70	ADV_NONCONN_IND
9	0.244319100	25:a8:2f:58:9b:d7	Broadcast	LE LL	70	ADV_NONCONN_IND
10	0.284943700	05:d0:da:52:56:53	Broadcast	LE LL	70	ADV_NONCONN_IND
11	0.308475600	3c:42:15:ea:6a:37	Broadcast	LE LL	70	ADV_NONCONN_IND
12	0.315679200	10:4f:f0:47:20:bc	Broadcast	LE LL	70	ADV_NONCONN_IND
13	0.366941000	32:ed:44:d6:3b:29	Broadcast	LE LL	70	ADV_NONCONN_IND
14	0.388712600	05:d0:da:52:56:53	Broadcast	LE LL	70	ADV_NONCONN_IND
15	0.491856000	05:d0:da:52:56:53	Broadcast	LE LL	70	ADV_NONCONN_IND
16	0.518215700	10:4f:f0:47:20:bc	Broadcast	LE LL	70	ADV_NONCONN_IND
17	0.526640100	3c:42:15:ea:6a:37	Broadcast	LE LL	70	ADV_NONCONN_IND
18	0.579442100	32:ed:44:d6:3b:29	Broadcast	LE LL	70	ADV_NONCONN_IND
19	0.600626400	05:d0:da:52:56:53	Broadcast	LE LL	70	ADV_NONCONN_IND
20	0.619483800	10:4f:f0:47:20:bc	Broadcast	LE LL	70	ADV_NONCONN_IND
21	0.634159800	3c:42:15:ea:6a:37	Broadcast	LE LL	70	ADV_NONCONN_IND
22	0.704395400	05:d0:da:52:56:53	Broadcast	LE LL	70	ADV_NONCONN_IND
23	0.720751800	10:4f:f0:47:20:bc	Broadcast	LE LL	70	ADV_NONCONN_IND
24	0.742929400	3c:42:15:ea:6a:37	Broadcast	LE LL	70	ADV_NONCONN_IND
25	0.808784700	05:d0:da:52:56:53	Broadcast	LE LL	70	ADV_NONCONN_IND
26	0.827020800	10:4f:f0:47:20:bc	Broadcast	LE LL	70	ADV_NONCONN_IND
27	0.849193400	3c:42:15:ea:6a:37	Broadcast	LE LL	70	ADV_NONCONN_IND
28	0.893505900	32:ed:44:d6:3b:29	Broadcast	LE LL	70	ADV_NONCONN_IND

On remarque qu'il s'agit de paquet de broadcast, il s'agit donc surement d'appareil ayant le Bluetooth d'activé mais sans partenaire. Regardons maintenant le détail d'un paquet :

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10:4f:f0:47:20:bc	Broadcast	LE LL	70	ADV_NONCONN_IND
2	0.002800400	3c:42:15:ea:6a:37	Broadcast	LE LL	70	ADV_NONCONN_IND
3	0.075534500	05:d0:da:52:56:53	Broadcast	LE LL	70	ADV_NONCONN_IND

▷ Frame 1: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
▷ PPI version 0, 24 bytes
DLT: 147, Payload: btle (Bluetooth Low Energy Link Layer)
Bluetooth Low Energy Link Layer
Access Address: 0x8e89bed6
▷ Packet Header: 0x2542 (PDU Type: ADV_NONCONN_IND, ChSel: #1, TxAdd: Random)
Advertising Address: 10:4f:f0:47:20:bc (10:4f:f0:47:20:bc)
Advertising Data
Manufacturer Specific
Length: 30
Type: Manufacturer Specific (0xff)
Company ID: Microsoft (0x0006)
▷ Data: 01092000a08900808e82ccd499df9e8d16b05b28aebcaf7...
CRC: 0x86a397

0000	00 00 18 00 93 00 00 00	36 75 0c 00 00 62 09 00 6u...b..
0010	87 2e 01 00 19 19 00 00	d6 be 89 8e 42 25 bc 20B%.
0020	47 f0 4f 10 1e ff 06 00	01 09 20 00 a0 89 00 80	G.O..... ..
0030	8e 82 cc d4 99 df 9e 8d	16 b0 5b 28 ae bc af c7[(....
0040	c1 f2 5e 61 c5 e9		..^a..

Nous récupérons quelques informations comme par exemple l'adresse MAC ou le constructeur (il s'agit ici de Microsoft). Nous avons essayé de chercher plus d'information avec l'adresse mac mais nous n'avons rien trouvé.

Deuxième expérience

La deuxième sniffing est celui réalisé à la maison, la capture complète est présente sur le répertoire GitHub sous le nom de *capture_home.pcapng*. En voici un extrait :

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	6e:53:52:32:91:c8	15:3b:6e:27:80:f1	LE LL	45	SCAN_REQ
2	48.620450100	de:aa:6e:f4:60:b8	Broadcast	LE LL	70	ADV_IND[Malformed Packet]
3	81.886184500	SunitecE_ad:78:c0	Broadcast	LE LL	69	ADV_IND
4	83.892429300	SunitecE_ad:78:c0	Broadcast	LE LL	69	ADV_IND
5	85.892424300	SunitecE_ad:78:c0	Broadcast	LE LL	69	ADV_IND
6	85.892751800	57:01:4b:07:7a:ae	SunitecE_ad:78:c0	LE LL	45	SCAN_REQ
7	89.902432400	SunitecE_ad:78:c0	Broadcast	LE LL	69	ADV_IND
8	91.911158600	SunitecE_ad:78:c0	Broadcast	LE LL	69	ADV_IND
9	93.912403500	SunitecE_ad:78:c0	Broadcast	LE LL	69	ADV_IND
10	95.914898300	SunitecE_ad:78:c0	Broadcast	LE LL	69	ADV_IND
11	95.915226000	57:01:4b:07:7a:ae	SunitecE_ad:78:c0	LE LL	45	SCAN_REQ
12	95.915487400	SunitecE_ad:78:c0	Broadcast	LE LL	39	SCAN_RSP
13	97.918643300	SunitecE_ad:78:c0	Broadcast	LE LL	69	ADV_IND
14	99.926138000	SunitecE_ad:78:c0	Broadcast	LE LL	69	ADV_IND
15	100.962275500	1c:bb:6c:42:e1:f9	Broadcast	LE LL	38	ADV_NONCONN_IND[Malformed Packet]
16	101.926132900	SunitecE_ad:78:c0	Broadcast	LE LL	69	ADV_IND
17	103.933627500	SunitecE_ad:78:c0	Broadcast	LE LL	69	ADV_IND
18	105.939872400	SunitecE_ad:78:c0	Broadcast	LE LL	69	ADV_IND
19	107.943617100	SunitecE_ad:78:c0	Broadcast	LE LL	69	ADV_IND
20	109.946112000	SunitecE_ad:78:c0	Broadcast	LE LL	69	ADV_IND
21	111.953606700	SunitecE_ad:78:c0	Broadcast	LE LL	69	ADV_IND
22	115.210253500	f8:dd:77:07:a0:fb	Broadcast	LE LL	70	ADV_IND[Malformed Packet]
23	123.991076400	SunitecE_ad:78:c0	Broadcast	LE LL	69	ADV_IND
24	125.992321400	SunitecE_ad:78:c0	Broadcast	LE LL	69	ADV_IND
25	127.997316300	SunitecE_ad:78:c0	Broadcast	LE LL	69	ADV_IND
26	130.004811400	SunitecE_ad:78:c0	Broadcast	LE LL	69	ADV_IND
27	132.011058900	SunitecE_ad:78:c0	Broadcast	LE LL	69	ADV_IND
28	134.011051400	SunitecE_ad:78:c0	Broadcast	LE LL	69	ADV_IND
29	136.018546400	SunitecE_ad:78:c0	Broadcast	LE LL	69	ADV_IND

On retrouve des paquets de broadcast mais également des « scan request » et des « scan response ». En regardant en détail un paquet on observe également quelques informations intéressantes :

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	6e:53:52:32:91:c8	15:3b:6e:27:80:f1	LE LL	45	SCAN_REQ
2	48.620450100	de:aa:6e:f4:60:b8	Broadcast	LE LL	70	ADV_IND[Malformed Packet]
3	81.886184500	SunitecE_ad:78:c0	Broadcast	LE LL	69	ADV_IND

Frame 3: 69 bytes on wire (552 bits), 69 bytes captured (552 bits) on interface 0

PPI version 0, 24 bytes

DLT: 147, Payload: btle (Bluetooth Low Energy Link Layer)

Bluetooth Low Energy Link Layer

Access Address: 0x8e89bed6

Packet Header: 0x2400 (PDU Type: ADV_IND, ChSel: #1, TxAdd: Public)

Advertising Address: SunitecE_ad:78:c0 (fc:a8:9a:ad:78:c0)

Advertising Data

Flags

Appearance: Generic Tag

Device Name: JBL Flip 3

Length: 11

Type: Device Name (0x09)

Device Name: JBL Flip 3

Manufacturer Specific

Length: 10

Type: Manufacturer Specific (0xff)

Company ID: Harman International Industries, Inc. (0x0057)

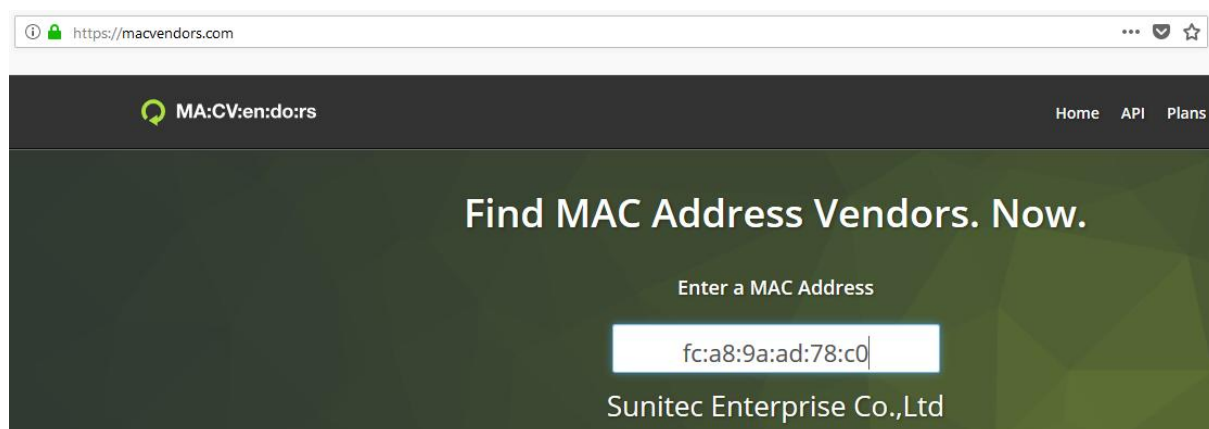
Data: 5300230005ad25

[Expert Info (Note/Undecoded): Undecoded]

CRC: 0x30f51f

Page 17

Cette fois, en plus de l'adresse MAC et du constructeur, on peut voir le nom de l'appareil (JBL Flip 3). En recherchant l'adresse MAC sur un site spécialisé on retrouve le nom de l'entreprise l'ayant produit :



The screenshot shows a web browser window with the address bar displaying "https://macvendors.com". The website has a dark header with the logo "MA:CV:en:do:rs" and navigation links "Home", "API", and "Plans". The main content area has a dark green background with the text "Find MAC Address Vendors. Now." in white. Below this, there is a label "Enter a MAC Address" and a text input field containing the MAC address "fc:a8:9a:ad:78:c0". Underneath the input field, the result "Sunitec Enterprise Co.,Ltd" is displayed.

Annexes

Annexe A : Guide d'installation d'Ubertooth

Annexe B : Protocole de l'expérience

Annexe A : Guide d'installation d'Ubertooth

Installer libusb :

```
sudo apt-get install cmake libusb-1.0-0-dev make gcc g++ libbluetooth-dev \
pkg-config libpcap-dev python-numpy python-pyside python-qt4
```

Installer libbtbb :

```
wget https://github.com/greatscottgadgets/libbtbb/archive/2017-03-R1.tar.gz
-O libbtbb-2017-03-R1.tar.gz
tar xf libbtbb-2017-03-R1.tar.gz
cd libbtbb-2017-03-R1
mkdir build
cd build
cmake ..
make
sudo make install
```

Installer ubertooth :

```
wget https://github.com/greatscottgadgets/ubertooth/releases/download/2017-
03-R1/ubertooth-2017-03-R1.tar.xz -O ubertooth-2017-03-R1.tar.xz
tar xf ubertooth-2017-03-R1.tar.xz
cd ubertooth-2017-03-R1/host
mkdir build
cd build
cmake ..
make
sudo make install
```

Installer le plugin wireshark BTBB :

```
sudo apt-get install wireshark wireshark-dev libwireshark-dev cmake
cd libbtbb-2017-03-R1/wireshark/plugins/btbb
mkdir build
cd build
cmake -DCMAKE_INSTALL_LIBDIR=/usr/lib/x86_64-linux-
gnu/wireshark/libwireshark3/plugins ..
make
sudo make install
```

Installer le plugin wireshark BR/EDR :

```
sudo apt-get install wireshark wireshark-dev libwireshark-dev cmake
cd libbtbb-2017-03-R1/wireshark/plugins/btbredr
mkdir build
cd build
cmake -DCMAKE_INSTALL_LIBDIR=/usr/lib/x86_64-linux-
gnu/wireshark/libwireshark3/plugins ..
make
sudo make install
```

Annexe B : Protocole de l'expérience

Pour réaliser l'expérience il faut être muni d'un Ubertooth one et de deux appareils communiquant grâce au Bluetooth low energy (BLE), d'avoir installé wireshark ainsi que tous les composants indiqués dans le guide d'installation.

Avant de commencer le sniffing il faut créer un pipe qui permettra à l'Ubertooth de communiquer avec wireshark, nous le faisons de la manière suivante :

```
rm /tmp/pipe  
mkfifo /tmp/pipe
```

Il faut ensuite lancer wireshark en l'associant au pipe créé :

```
sudo wireshark -ni /tmp/pipe
```

Enfin il faut lancer Ubertooth avec l'option BLE en le dirigeant vers le pipe :

```
ubertooth-btle -f -c /tmp/pipe
```

Il faut maintenant lancer la communication entre les appareils et le sniffing devrait alors s'effectuer.

En cas d'erreur du type « User encapsulation not handled: DLT=147, check your Preferences->Protocols->DLT_USER » il faut faire les étapes suivantes :

1. Cliquer sur Edit -> Preferences
2. Cliquer sur Protocols -> DLT_USER
3. Cliquer sur Edit (Encapsulations Table)
4. Cliquer sur New
5. Sous DLT, sélectionner "User 0 (DLT=147)" (ajuster cette sélection si le message d'erreur montre un DLT différent du nombre 147)
6. Sous Payload Protocol, entrer : btle
7. Cliquer sur OK
8. Cliquer sur OK