Escuela Politécnica Superior de Jaén

Máster Universitario en Seguridad Informática

UNIVERSIDAD DE JAÉN

Escuela Politécnica Superior (Jaén)

ESPÍA BLUETOOTH

Alumno/a: **Serrano Martínez, Carlos Jesús**

Tutor/a: Prof. D. Manuel Carlos Díaz Galiano

Dpto.: Departamento de Informática

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Don Manuel Carlos Díaz Galiano , tutor del Proyecto Fin de Carrera titulado: Espía Bluetooth, que presenta Carlos Jesús Serrano Martínez, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, Noviembre de 2019

El alumno: Los tutores:

SERRANO MARTINEZ CARLOS JESUS - 77381316R

Firmado digitalmente por SERRANO MARTINEZ CARLOS JESUS - 77381316R
Fecha: 2019.11.22 17:24:44 +01'00'

Carlos Jesús Serrano Martínez Manuel Carlos Díaz Galiano

Índice

1. INTRODUCCIÓN

No cabe duda de que el constante avance y proliferación de la tecnología, ha hecho que nuestra vida, rutinas y la forma de hacerlo todo, sea muy diferente respecto a unos años atrás. Y no solo eso, puesto que la incorporación de cada vez más dispositivos y diferentes elementos tecnológicos a nuestro día a día está haciendo que seamos cada vez más dependientes de estos y les otorgamos cada vez más control de nuestro espacio, nuestro tiempo y nuestra privacidad.

Se trabaja constantemente para que esta tecnología sea cada vez más invisible y menos invasiva para el usuario que la utiliza ya que se busca obtener todas las comodidades y el beneficio que puede ofrecernos, pero intentando integrar su uso en nuestra rutina sin que prácticamente, nos demos cuenta que la estamos usando.

Para ello, se desarrollan continuamente dispositivos que son capaces de realizar más y más funciones de forma independiente pero que haciendo uso de **tecnologías inalámbricas,** son capaces de interconectarse entre sí e incluso a internet.

Una de las tecnologías más se está usando en la actualidad para la interconexión entre dispositivos es sin ninguna duda el **Bluetooth** que será el gran protagonista de este trabajo. Es una tecnología muy asentada y fácil de usar, pero también tiene fallos de seguridad que lo hacen vulnerable a los ataques.

El ejemplo más claro de todo esto, es sin duda el auge de los dispositivos relacionados con **IoT**, porque ¿Quién no tiene algún dispositivo de uso diario que se conecte al móvil de forma inalámbrica? Ya sea una **pulsera deportiva, un smartwatch, unos auriculares inalámbricos, el coche conectado, algún dispositivo de domótica, etc.**

Sin prácticamente darnos cuenta, tenemos multitud de productos tecnológicos que siempre van con nosotros, y que están intercambiando información de manera constante entre ellos y como hemos dicho antes, también conectándose a internet.

En este trabajo, se ilustrará, mediante una investigación teórica y un caso práctico, utilizando un software que se ha desarrollado para este proyecto, que algo

que se usa en nuestro día a día cómo es el **Bluetooth**, puede hacer que, de una forma u otra, estemos expuestos y un posible atacante, pueda obtener algún tipo de información nuestra.

2. DESCRIPCIÓN DEL TRABAJO

En este trabajo, como se ha comentado con anterioridad, se va a realizar un estudio teórico acerca del Bluetooth. Este estudio incluirá la evolución de esta tecnología a través de los años, así como su funcionamiento, vulnerabilidades conocidas, librerías para usar Bluetooth entre otros elementos y que se detallarán a lo largo de todo este documento.

Además del estudio teórico, se va a desarrollar un software que sea capaz de captar cuáles son los dispositivos Bluetooth que se encuentran abiertos en el rango de alcance del hardware donde sea procesado. En este caso, ya adelantamos que se trabajará con una Raspberry Pi y utilizando el lenguaje de programación Python. Este software será desarrollado después de buscar e investigar qué librerías software son capaces de ayudarnos a trabajar de una forma más sencilla con dispositivos Bluetooth, puesto que es inviable el desarrollo desde cero, trabajando a bajo nivel directamente con el hardware. Este software, servirá como base para llevar a cabo un caso práctico en el que se buscará obtener datos de diferentes dispositivos Bluetooth visibles en un determinado lugar, y posteriormente llevar a cabo un procesamiento de dichos datos.

3. OBJETIVOS DEL TRABAJO

3.1. Objetivo general

En primer lugar, se tiene el objetivo de informar acerca del Bluetooth, su historia, vulnerabilidades, protocolos y demás que se verán con más detalle posteriormente. Para cumplir con este objetivo, se desarrollará el ya comentado estudio teórico acerca de esta tecnología.

Pero el objetivo real del proyecto, es ilustrar cómo, a través de una tecnología tan común y tan integrada en nuestras vidas, un atacante podría obtener información nuestra con el simple hecho de tener el Bluetooth activo y abierto. Para cumplir con

este objetivo se desarrollará un software capaz de utilizar el Bluetooth para obtener una serie de datos y posteriormente obtener información y conocimiento de los mismos

3.2. Objetivos concretos

- Estudio teórico acerca de la tecnología Bluetooth.
- Búsqueda de librerías que puedan ayudarnos a trabajar con Bluetooth y nos ayuden a decidir que tecnologías, lenguajes y herramientas podemos utilizar
- Desarrollo de un software que sea capaz de obtener información de dispositivos Bluetooth abiertos.
- Desarrollo de un caso práctico en el que hagamos uso de los datos obtenidos con el software anterior, donde se expondrá, con datos reales, diferentes acciones que un usuario malintencionado haciendo uso de dicha información.

4. BLUETOOTH

4.1. Introducción

La tecnología Bluetooth ha evolucionado y mejorado a lo largo de los años, ha ganado una amplia aceptación y es que se encuentran cada vez más en muchos aspectos de la vida cotidiana.

Bluetooth se ha convertido en el estándar para las tecnologías de corto alcance para comunicaciones inalámbricas que más se usa en la actualidad. Se puede encontrar en los hogares en los dispositivos como **smartphones, auriculares, altavoces, impresoras, teclados, automóviles y dispositivos médicos, etc**... Y todo eso sin mencionar todo lo relacionado con **IoT** (mencionado en la introducción de este documento).

En este apartado, y como hemos comentado antes, se va a hablar sobre esta tecnología. Este estudio servirá para conocerla un poquito mejor, y ver cuál es su situación actual.

4.2. ¿Qué es el Bluetooth?

Bluetooth [1, 2] es una especificación industrial para redes inalámbricas de área personal (WPAN) que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia en la banda ISM de los 2.4 GHz. Algo clave es que puede transmitir tanto voz como datos simultáneamente. Además, soporta enlaces (de datos) asíncronos y síncronos. Los principales objetivos que se pretenden conseguir son:

- Facilitar las comunicaciones entre equipos móviles.
- Eliminar los cables y los conectores entre ellos.
- Creación de pequeñas redes inalámbricas para facilitar la sincronización de datos entre equipos personales.

Ilustración 1. Bluetooth Logo [2]

4.3. Origen del Bluetooth

Un ingeniero llamado **Sven Mattisson**, que se graduó como ingeniero en 1979, fue contratado por la compañía **Ericsson Mobile Communications** en 1995, donde comenzó a trabajar en radioenlaces de corto alcance con baja potencia. La empresa tenía la intención de desarrollar las comunicaciones móviles que permitiera a un dispositivo comunicarse sin tener que conectarse por medio de cables.

En Ericsson vieron que se trataba de una tecnología con mucho potencial, pero que necesitarían ayuda para seguir avanzando en el desarrollo de la tecnología, y en 1997 se unió a **Intel** para seguir trabajando en ella. Concretamente, **Jim Kardach**, que se unió a Mattisson, y juntos vieron que esta tecnología, tenía mucho más potencial que el simplemente usarla en teléfonos móviles. Querían que la tecnología sirviese para conectar de forma inalámbrica, prácticamente cualquier dispositivo.

Finalmente salió al mercado en 1998. Esta tecnología, no se encontraba inmediatamente lista para ser producida en masa, y a las empresas, al principio, eran reacias a unirse al uso de esta tecnología. Algunas de ellas lo descartaron debido a la baja tasa de transferencia de datos, que sólo era de unos 721 kbps. Pero poco a poco la tecnología comenzó a asentarse, y en un plazo de 10 años había tres especificaciones y velocidades de transferencia superiores a 20 Mbps. [3]

4.3.1. Etimología y logo

El nombre procede del rey danés y noruego **Harald Blåtand** [4], cuya traducción al inglés era **Harald Bluetooth**. Este rey fue conocido por unificar a las tribus noruegas y a las suecas-danesas y convertirlas al cristianismo. El nombre de Bluetooth para esta tecnología fue propuesto por Jim Kardach. [5, 6]

El logo de Bluetooth combina las runas Hagall (ᚺ) y Berkana (ᛒ), que corresponden a las iniciales de **H**arald **B**luetooth. [5, 6]

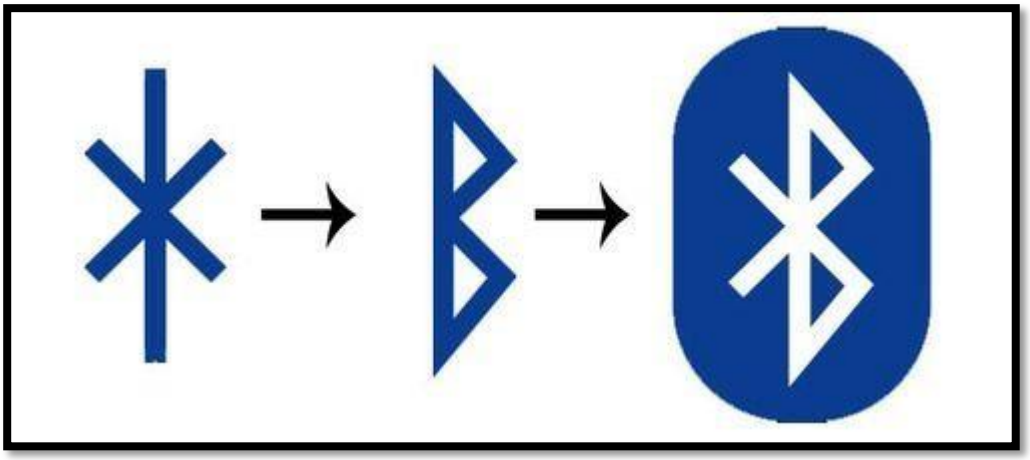


Ilustración 2. Runas que forman el logo de Bluetooth [6]

4.4. Clasificación Bluetooth

4.4.1. Introducción

En este apartado, se lleva a cabo una clasificación de los dispositivos Bluetooth. Estos se clasifican de acuerdo a la **clase** y a la **versión** de Bluetooth empleada.

4.4.2. Clases de módulos Bluetooth

Podemos clasificar los módulos Bluetooth en **4 clases diferentes**, según su rango de alcance y potencia de consumo (a más potencia, más alcance), y son las siguientes:

- **Clase 1**: Tiene un rango Tiene un rango de alcance hasta 100 metros, que implica una potencia de consumo de unos 100 mW.
- **Clase 2**: Tiene un rango de alcance de entre unos 5 y 10 metros, que tiene una potencia de consumo de unos 2'5 mW. Este es el más común de todos y que podemos encontrar en nuestros smartphones por ejemplo.
- **Clase 3**: Tiene un rango de operación de hasta 1 metro, con potencia de consumo promedio de 1mW.
- **Clase 4**: Tiene un rango de cobertura de hasta 0'5 metros y una potencia de 0'5 W.

Clase	Potencia máxima permitida (mW)	Potencia máxima permitida (dBm)	Alcance (aproximado)
Clase 1	100 mW	20 dBm	~ 100 metros
Clase 2	2.5 mW	4 dBm	~ 10 metros
Clase 3	1 mW	0 dBm	~ 1 metro
Clase 4	0.5 mW	0 dBm	~ 0'5 metros

Tabla 1. Potencias y rangos de alcance según clase de dispositivo. [1]

4.4.3. Versiones Bluetooth

Bluetooth 1.0 [7]

Esta fue la primera versión usada para transmitir datos, pero que actualmente se encuentra en desuso. Al ser la primera versión, hubo muchos problemas de comunicación entre dispositivos. Posteriormente, llegaron las actualizaciones 1.1 y 1.2, que fueron las **primeras en ser reconocidas como un estándar de comunicación IEEE**, por proveer una conexión más rápida y poder detectar otros dispositivos con Bluetooth. La tasa de transmisión de datos era de unos 721 kbps.

Bluetooth 2.0 [7]

Una de las versiones clave en este punto fue la actualización a Bluetooth 2.1 + BR/EDR, que permitió que los usuarios se conectaran más fácilmente. Esto, permitió que un dispositivo pudiera agregar otro dispositivo con Bluetooth de un menú que permitiera detectar y conectarse automáticamente con otro. Los conceptos BR/EDR (Basic Rate/Enhaced Data Rate) hacen referencia a la tasa de transmisión de datos, por un lado, en BR se puede transmitir 1 Mbps mientras que EDR tiene tasas de transferencia hasta de 2 Mbps. [7]

Bluetooth 3.0 [7]

Se incorporó la característica **HS** (High Speed), lo que hace que pueda transmitir **paquetes que contienen más datos**, como archivos de video y audio. Además, su tasa de transferencia es de 24 Mbps.

Bluetooth 4.0 [7]

Es el estándar más extendido en la actualidad, más concretamente sus actualizaciones 4.1 y 4.2. En esta versión, se introduce el concepto de **Bluetooth Smart**, que **incluye los protocolos Bluetooth clásico** (los que corresponden a las versiones 1, 2 y 3), **Bluetooth HS**, además de **Bluetooth Low Energy (BLE)**. BLE aparece justo en esta versión, el cual está enfocado principalmente para elementos que funcionen con internet de las cosas (IoT), puesto que su consumo energético es menor para dispositivos que tienen que funcionar durante un periodo largo de tiempo. En general Bluetooth 4.0 permite tasas de transferencia de entre 25 Mbps hasta unos 32 Mbps.

Debido a que hay dispositivos con la versión 4 que no tiene el protocolo LE, se decidió renombrar esta versión para distinguir los dispositivos compatibles con el protocolo LE:

- **Modo único (single-mode devices)**: sólo implementan LE y no Bluetooth BR/EDR. Se les denomina dispositivos **Bluetooth Smart**. Los dispositivos de este tipo, sólo podrán funcionar con dispositivos Bluetooth LE o Bluetooth Smart Ready.

- **Modo dual (dual-mode devices)**: implementan LE y también Bluetooth BR/EDR. Se les denomina **Bluetooth Smart Ready**. Los dispositivos de este tipo pueden interactuar tanto con Bluetooth clásico como con dispositivos Bluetooth LE.

La mayoría de los **ordenadores y smartphones** son duales, debido a que en ocasiones puede requerirse una tasa de transferencia de datos alta, o una distancia de operación más elevada.

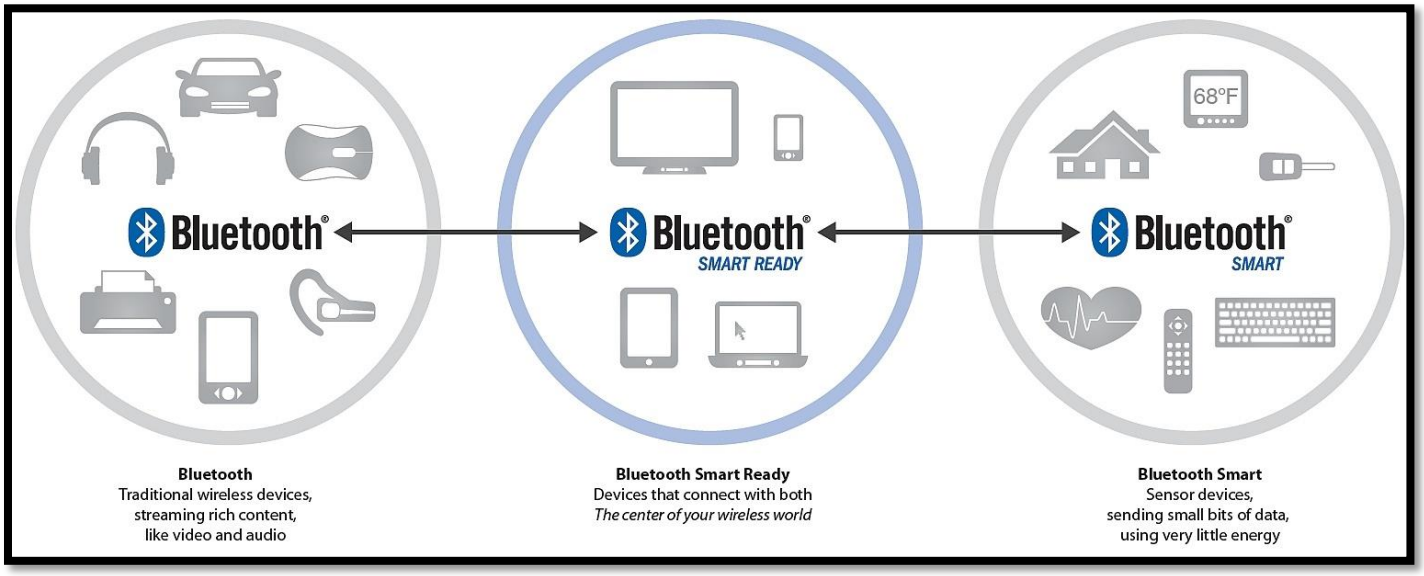


Ilustración 3. Cómo se pueden conectar los tipos de dispositivos. [7]

Bluetooth 5.0 [7]

Esta es la última versión de Bluetooth, y que se encuentra en los dispositivos que están saliendo al mercado en la actualidad. En esta versión se optimiza el **Bluetooth Low Energy (BLE)**, y con esto se pretende mejorar su funcionalidad para dispositivos IoT por medio de una doble tasa de transferencia de datos y un rango de cobertura cuatro veces mayor con respecto a las versiones 4.1 y 4.2, así como la capacidad de soportar flujos de datos con varios dispositivos simultáneamente.

4.4.4. Compatibilidad entre dispositivos

Cuando se habla de **clases de bluetooth**, es posible la interacción entre ellos sin problemas, siempre y cuando los dispositivos a interactuar, se sostengan al dispositivo con la clase más alta, debido al rango de alcance de este.

Atendiendo a la **versión**, de forma teórica y tomando como referencia la **versión 4** (la versión 5 es muy actual y no hay demasiada información), es compatible con versiones anteriores de Bluetooth siempre y cuando la última versión de Bluetooth que usamos, no sea del tipo Smart Device debido a que utiliza protocolos de comunicación distintos a los del Bluetooth clásico (Bluetooth Smart Ready). Como puede ser un poco confuso, a continuación, lo ilustramos con una tabla.

Si dispositivo es de este tipo	**Es compatible con esos tipos**		
Bluetooth SMART READY	Bluetooth SMART READY	Bluetooth	Bluetooth SMART
Bluetooth	Bluetooth SMART READY	Bluetooth	
Bluetooth SMART	Bluetooth SMART READY		

Tabla 2. Compatibilidad entre dispositivos Bluetooth. [7]

4.5. Arquitectura Bluetooth

La arquitectura de Bluetooth [8] está formada por una serie de capas, que definen las **funcionalidades** y **los protocolos**. Esta pila, se puede dividir en dos partes, una de ellas formada por las capas superiores de la pila que forman una entidad denominada **Host** y la otra, formada por las capas inferiores de la pila que forman otra entidad denominada **Controller** (Controlador).

Existen dos tipos de controladores: los **primarios** y los **secundarios**.

- **Controlador primario**:
  - **Controlador BR/EDR**, usado por la tecnología Basic Rate/ Enhanced data Rate.
  - **Controlador LE**, usado por la tecnología Low Energy.
  - Una **combinación** de controlador LE y controlador BR/EDR.
- **Controlador secundario**:
  - **Controlador AMP** (Alternate MAC/PHY), usado por la tecnología HS (High Speed) para hacer uso del canal 802.11 en la transferencia de datos de alta velocidad.

Cabe destacar, que el Host y el controlador pueden estar juntos o separados (por ejemplo, en dos procesadores diferentes). Cuando esto sucede (se encuentran separados), es necesaria la capa HCI (Host Controller Interface) que es una interfaz entre el Host y el Controller.

Las diferentes implementaciones de Bluetooth, pueden estar formadas de diferentes maneras:

- Host, un controlador primario y ninguno secundario.
- Host, un controlador primario y uno secundario.
- Host, un controlador primario y varios secundarios.

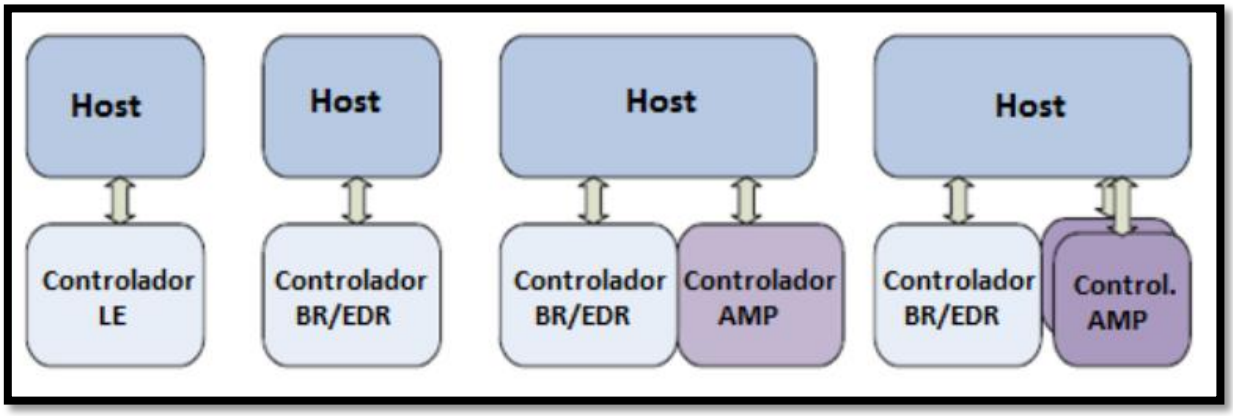


Ilustración 4. Combinaciones de Host y Controladores Bluetooth. [8]

En la siguiente imagen, se pueden ver las distintas **capas de la pila de Bluetooth**, correspondientes con las entidades Host y Controller para las distintas tecnologías que se han comentado con anterioridad.

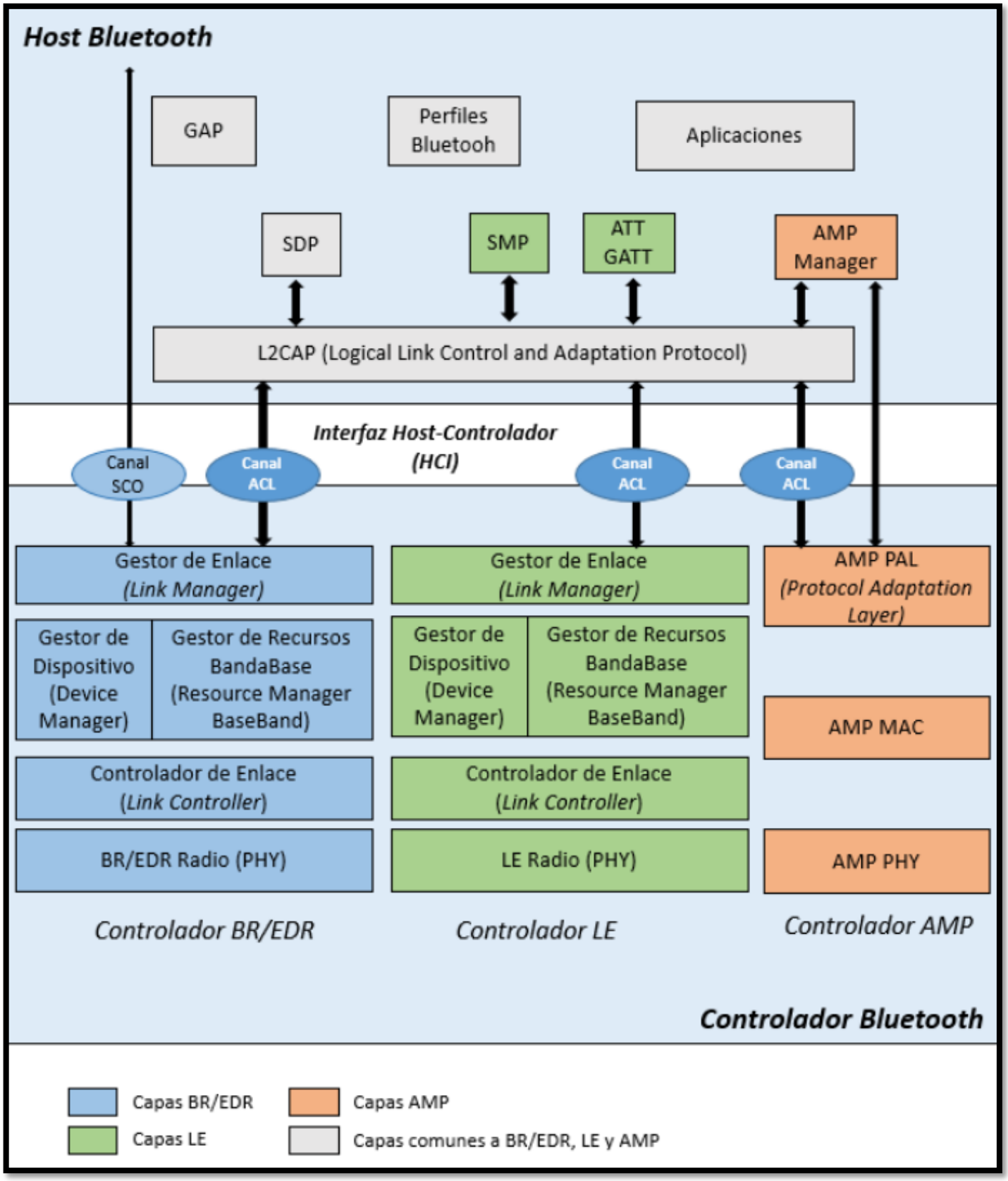


Ilustración 5. Arquitectura Bluetooth. [8]

4.5.1. Módulo Host

- **L2CAP** (Logical Link Control and Adaptation Protocol) [8]: Protocolo de adaptación y control de enlace lógico. Esta capa está encargada de las tareas que se detallan a continuación:
  - Establecimiento de conexiones a través de enlaces ACL (Asíncronos) existentes, o la solicitud de un enlace ACL si no existe debido a una solicitud anterior.
  - Multiplexación de los datos que recibe de los protocolos de capas superiores para permitir que varias aplicaciones puedan utilizar el mismo enlace ACL.
  - Tareas relacionadas con la calidad de servicio (QoS), fragmentación y re-ensamblado de tramas Bluetooth.

  Este protocolo, utiliza el concepto de **canales**. Estos canales permiten la transmisión de datos procedentes de múltiples aplicaciones sobre un mismo enlace Bluetooth. Los canales se identifican con un identificador de canal (CID).

- **SDP** (Service Discovery Protocol) [8]: Este se denomina Protocolo de descubrimiento de Servicio y permite a un dispositivo descubrir los servicios que ofrecen otros dispositivos que se encuentran al alcance. Por ejemplo. cuando usas un teléfono móvil con unos auriculares Bluetooth, el teléfono usa SDP para determinar qué perfil de Bluetooth pueden usar los auriculares y los ajustes del protocolo de multiplexación necesarios para que el teléfono pueda conectarse con los auriculares. Cada servicio está identificado por un **UUID** (Universally Unique Identifier)
- **GAP** (Generic Access Profile) [8]: Representa la funcionalidad básica común a todos los dispositivos Bluetooth: modos y procedimientos de conexión, descubrimiento de servicios, seguridad, autenticación, modelos de asociación, etc.
- **SMP** (Security Manager Protocol) [8]: Protocolo de gestión de la seguridad encargado de generar y almacenanar las claves de la conexión. Opera sobre un canal L2CAP dedicado, y gestiona la funcionalidad de privacidad LE.

Esta capa sólo existe en el Host de sistemas LE. En BR/EDR, esta funcionalidad la proporciona el Gestor de Enlace (Link Manager) de la Controladora BR/EDR.

- **ATT** (Attribute Protocol) [8]: Protocolo de atributos que forma el protocolo punto a punto entre un servidor y un cliente de atributos. Un cliente se comunica con un servidor de tributos a través de un canal L2CAP dedicado.
- **GATT** (Generic Attribute Profile) [8]: Esta capa representa la funcionalidad del servidor de atributos y del cliente de atributos. También describe cual es la jerarquía que sigue los diferentes servicios. Solo se emplea en los servicios Low Energy.
- **Gestor de AMP** (AMP Manager) [8]: Es una capa que utiliza canales de señalización L2CAP para comunicarse extremo a extremo con el AMP Manager de un dispositivo remoto con objeto de recolectar información para el establecimiento y gestión de los enlaces físicos AMP. También interacciona directamente con el AMP PAL para propósitos de control AMP.

4.5.2. Controlador BR/EDR/LE

- **Gestor de Dispositivos** (Device Manager) [8]: Es un bloque funcional dentro de la capa de Banda Base que controla el comportamiento general del dispositivo. Es el encargado de todas aquellas tareas que no estén relacionadas con el transporte de datos, como el descubrimiento, conexión, gestión de modo visible/oculto, etc.
- **Gestor de Enlace** (Link Manager) [8]: es el encargado de crear, configurar y gestionar de los enlaces lógicos. Para ello se comunica con el Gestor de Enlace del dispositivo remoto, utilizando el protocolo LMP (Link Manager Protocol) en tecnología BR/EDR, y el protocolo LL (Link Layer Protocol) en tecnología LE.

Los dos protocolos permiten la creación de nuevos enlaces lógicos entre dispositivos, además del control general en los enlaces y el control en la configuración del transporte de datos. Entre los dispositivos Bluetooth existen dos tipos de enlaces:

- **SCO** (Synchronous Connection-Oriented): Enlaces Síncronos, Orientados a Conexión para transmisiones de voz.
- **ACL** (Asynchronous Connectionless): Enlaces Asíncronos, No orientados a conexión para la transmisión de datos.

- **Gestor de Recursos en Banda Base** (Baseband Resource Manager) [8]: Es responsable de todos los accesos al medio de radio (PHY). Tiene dos funciones principales:
  - Programar slots temporales para el uso del canal físico por parte de todas las entidades que hayan contratado el acceso y uso.
  - Negociar las características de estos contratos de acceso (aspecto como QoS)
- **Controlador de enlace** (Link Controller) [8]: Es el encargado de la codificación y decodificación de los paquetes de datos y sus parámetros, para su transmisión por el canal físico (PHY). También lleva a cabo la señalización de los protocolos LMP (en BR/EDR) y LL (en LE) para control de flujo y señales de aceptación y retransmisión de peticiones.
- **PHY** (Physical Layer): Capa Física, que es la encargada de la transmisión y recepción de paquetes a través del canal físico. Lleva a cabo la modulación y demodulación de los datos, en señales de radiofrecuencia para su transmisión en el medio aéreo.

4.5.3. Controlador AMP

- **AMP PAL** (Protocol Adaptation Layer) [8]: Esta capa hace de interfaz entre la capa AMP MAC y la parte del Host AMP (L2CAP y AMP Manager) ya que traduce los comandos del Host en primitivas MAC y viceversa. Además, proporciona soporte para la gestión de los canales AMP, el tráfico de datos y la eficiencia de consumo energético.
- **AMP MAC** (Media Access Control) [8]: Capa MAC tal y como se define en el modelo de referencia IEEE 802, que se encarga de proporcionar servicios como el direccionamiento físico y mecanismos de control y acceso a los canales.
- **AMP PHY** (Physical Layer) [8]: Al igual que antes, se trata de la capa física.

4.6. Perfiles Bluetooth (Bluetooth Profiles)

Los perfiles Bluetooth (Bluetooth Profiles) [8] nos permite lograr la interoperabilidad de aplicaciones a través de conexiones Bluetooth. Si dos dispositivos implementan un mismo perfil Bluetooth, se garantiza la interoperabilidad de una aplicación en ambos dispositivos.

Un perfil Bluetooth se podría explicar cómo la definición de un conjunto de funciones y características requeridas en cada una de las arquitecturas Bluetooth, incluyendo cualquier otro protocolo externo necesario fuera de la especificación Bluetooth y especificando, además, comportamientos, configuraciones y formatos de datos que deben emplear las aplicaciones

Los perfiles Bluetooth se agrupan en una jerarquía de grupos. Cada grupo depende de las características proporcionadas por el grupo predecesor. En la raíz de esta jerarquía se encuentra el perfil más básico denominado **GAP** (Generic Access Profile), que todo dispositivo Bluetooth implementa y en el que se basan todos los demás. Este perfil define los mecanismos básicos para el establecimiento de los enlaces de Banda Base entre dispositivos Bluetooth, y que además define:

- Qué características básicas deben implementarse en todos los dispositivos.
- Procedimientos básicos para el descubrimiento y enlace de los dispositivos.
- Aspectos básicos de la interfaz de usuario.

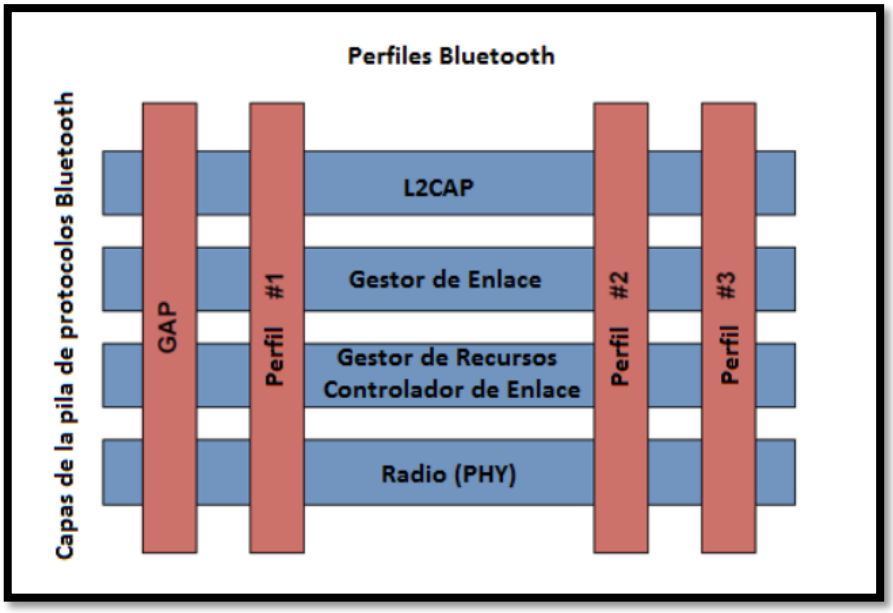


Ilustración 6. Perfiles Bluetooth. [8]

Bluetooth SIG proporciona las especificaciones de todos los perfiles que han sido adoptados para los dispositivos Bluetooth. [9]

4.7. Topología de los dispositivos Bluetooth

Como se ha dicho con anterioridad, Bluetooth opera en el espectro de radiofrecuencia de 2’4 GHz. Esto tiene la desventaja de que comparte el espectro de radiofrecuencia con muchos aparatos de consumo, como por ejemplo microondas, monitores de bebés, routers wifi, etc… Esto hace que sea posible que se produzcan interferencias cuando está en uso. [10]

4.7.1. Piconet y Scatternet

Una **Piconet** es un grupo de dos o más dispositivos Bluetooth que se comunican entre sí. La conexión entre un teléfono móvil y unos auriculares inalámbricos, es un ejemplo de una **Piconet Bluetooth Simple**. [10]

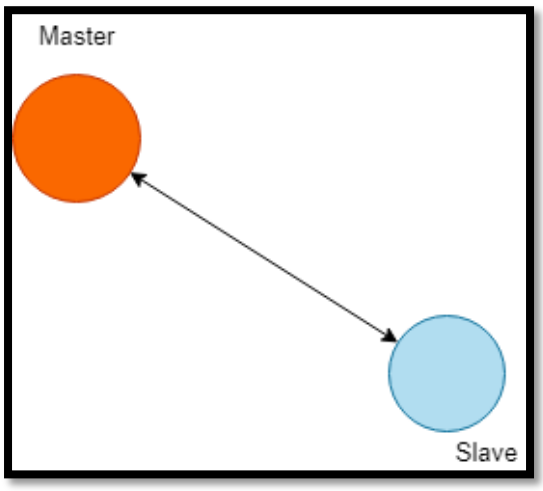


Ilustración 7. Piconet simple. [65]

La conectividad en una piconet es espontánea de manera **ad hoc** [11]. En una piconet, solo uno de los dispositivos se designa como **Master** (maestro) y todos los demás se denominan **Slaves** (esclavos). La mayoría de dispositivos Bluetooth pueden funcionar como maestro o como esclavo. Los dispositivos pueden pertenecer a más de una piconet. Los dispositivos que no forman parte de una piconet, se consideran **en espera**. Mientras que un dispositivo es maestro en una piconet, puede ser esclavo de otra. [10]

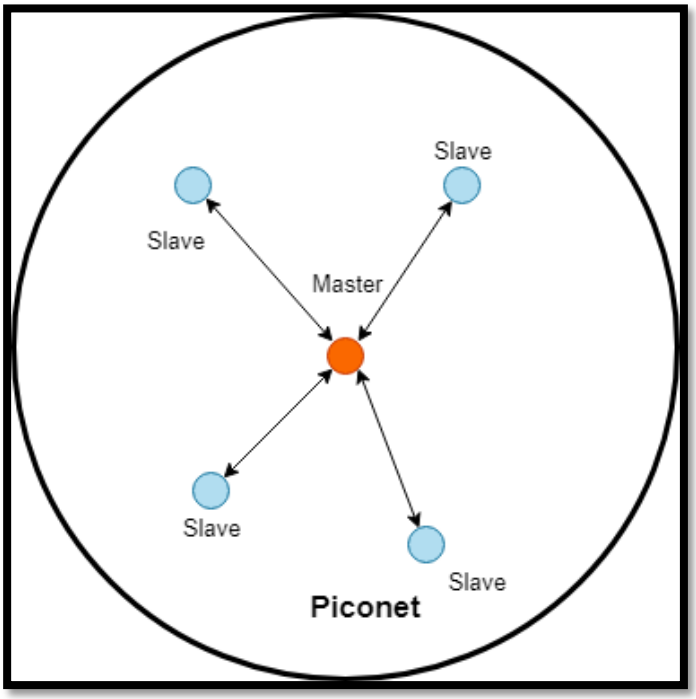


Ilustración 8. Piconet. [65]

En áreas muy próximas, es posible que existan varias piconets, de tal forma que un mismo dispositivo puede ser maestro de una piconet, y esclavo en otra (o varias) de forma simultánea. La combinación de 2 o más piconets forman una **Scatternet**. [10]

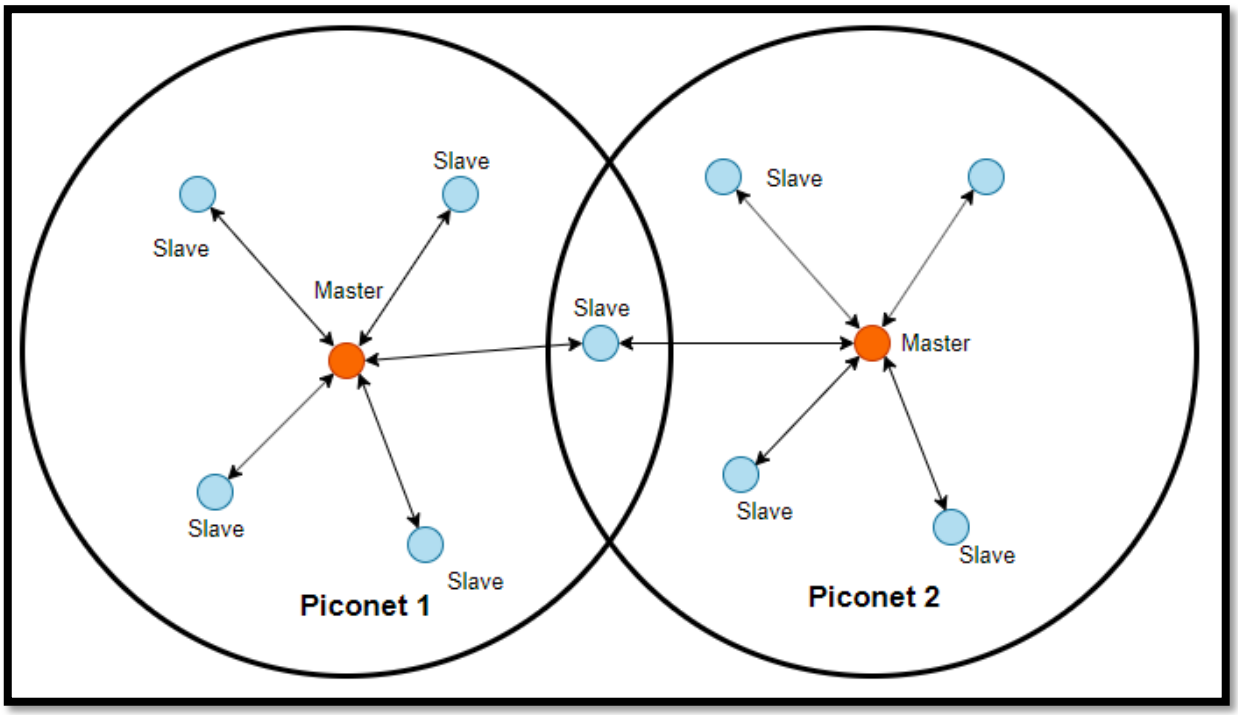


Ilustración 9. Scatternet [65]

Un maestro puede tener registrados muchos esclavos, pero solo un máximo de 7, pueden estar activos a la vez. Esto se consigue, debido a que el maestro asigna a cada esclavo activo una dirección única denominada **AM_ADDR** (Active Member Address o dirección de miembro activo). Los demás dispositivos que el Master tiene registrados, estarán "aparcados" y cuando el Master lo desee, podrá invitarlos para activarse.

La excepción en cuanto al **número de dispositivos activos a la vez**, se produce con el **Bluetooth LE**, que pueden tener un número **ilimitado** [10]

4.7.2. Topología BR/EDR vs Topología LE [8]

La forma normal que tiene el **Bluetooth BR/EDR**, es un canal físico de radio, compartido por un conjunto de dispositivos sincronizados a un reloj común. Esta sincronización se lleva a cabo utilizando un patrón concreto de salto de frecuencias. El **Master** es el dispositivo que proporciona la referencia de sincronización. Los

Slaves (esclavos), son todos los dispositivos que se sincronizan con a ese reloj y a ese salto de frecuencias del Master.

En la imagen de abajo, se puede ver gráficamente como sería esta conexión entre diferentes dispositivos.

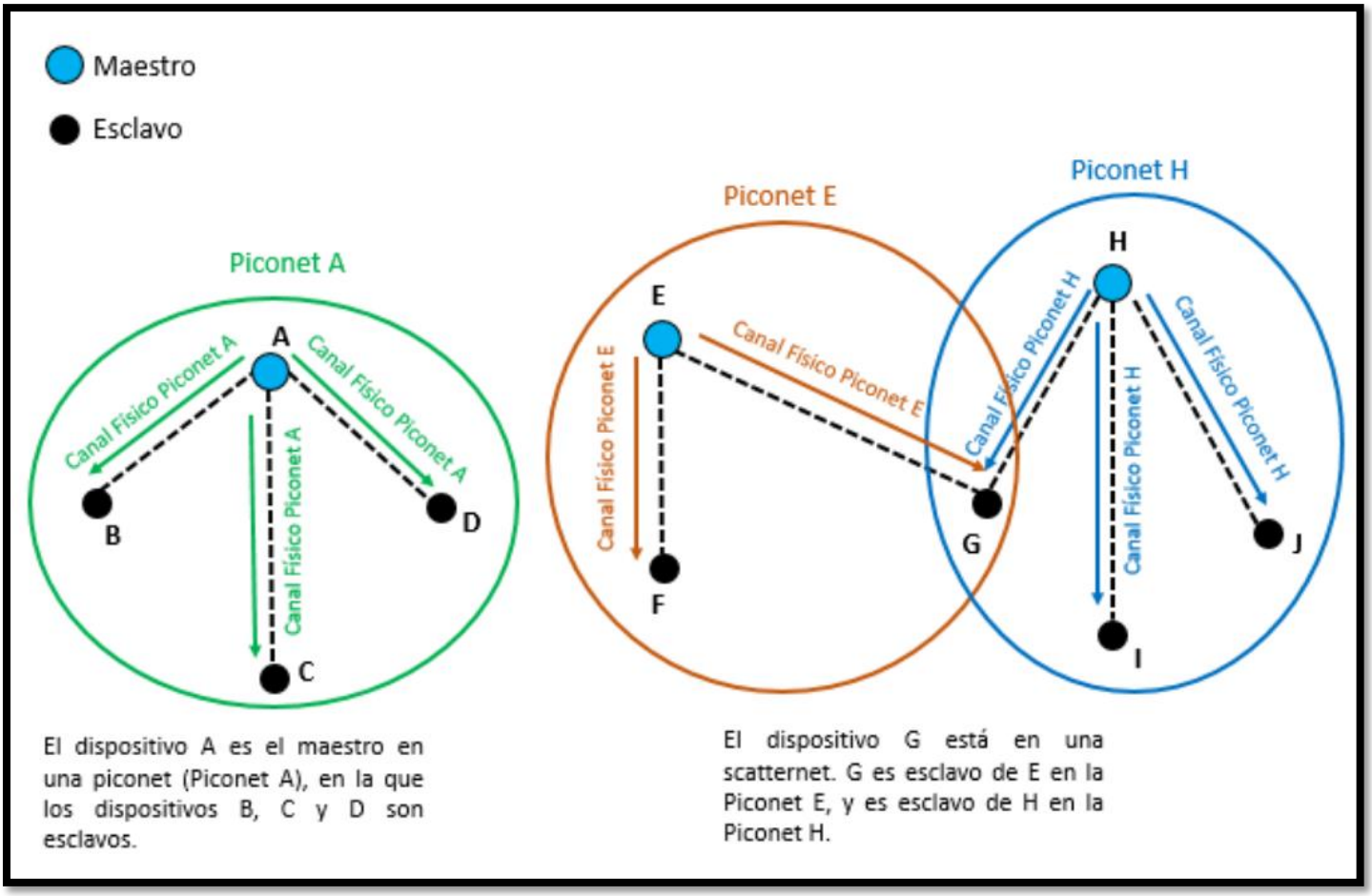


Ilustración 10. Topología Bluetooth BR/EDR [8]

En la tecnología de **Bluetooth LE**, el canal físico se subdivide en unidades de tiempo llamadas **eventos**. Los dispositivos del tipo LE, transmiten datos entre ellos, que se posicionan en estos eventos. Se pueden encontrar eventos de dos tipos diferentes: **Advertising** (de Publicidad) y **Connection** (de conexión).

Dentro de los eventos del tipo **Advertising,** existe un tipo específico de paquetes, que los dispositivos emiten para indicar a otros dispositivos, que se ya

encuentran disponibles para establecer una conexión. Estos paquetes se denominan **Advertising connectable packets** (paquetes de publicidad de conexión).

Los dispositivos que necesitan establecer una conexión con otro, estarán a la escucha de los **Advertising connectable packets** emitidos por aquellos dispositivos disponibles para llevar a cabo una conexión. Los dispositivos que están a la escucha se denominan **initiators** (iniciadores). Los que emiten la publicidad de conexión se denominan **advertisers** (publicadores).

Cuando se produce dicha conexión entre ambos dispositivos, se genera una piconet en la que el dispositivo **iniciador se convierte en master** y **el publicador en esclavo**. Una vez establecida esta conexión, se puede llevar a cabo una transmisión de datos entre los dispositivos, a través de eventos de conexión utilizando el mismo canal físico.

A diferencia de BR/EDR, en LE los esclavos en una piconet no comparten el mismo canal físico con el maestro. Cada esclavo se comunica con el maestro a través de un canal físico diferente.

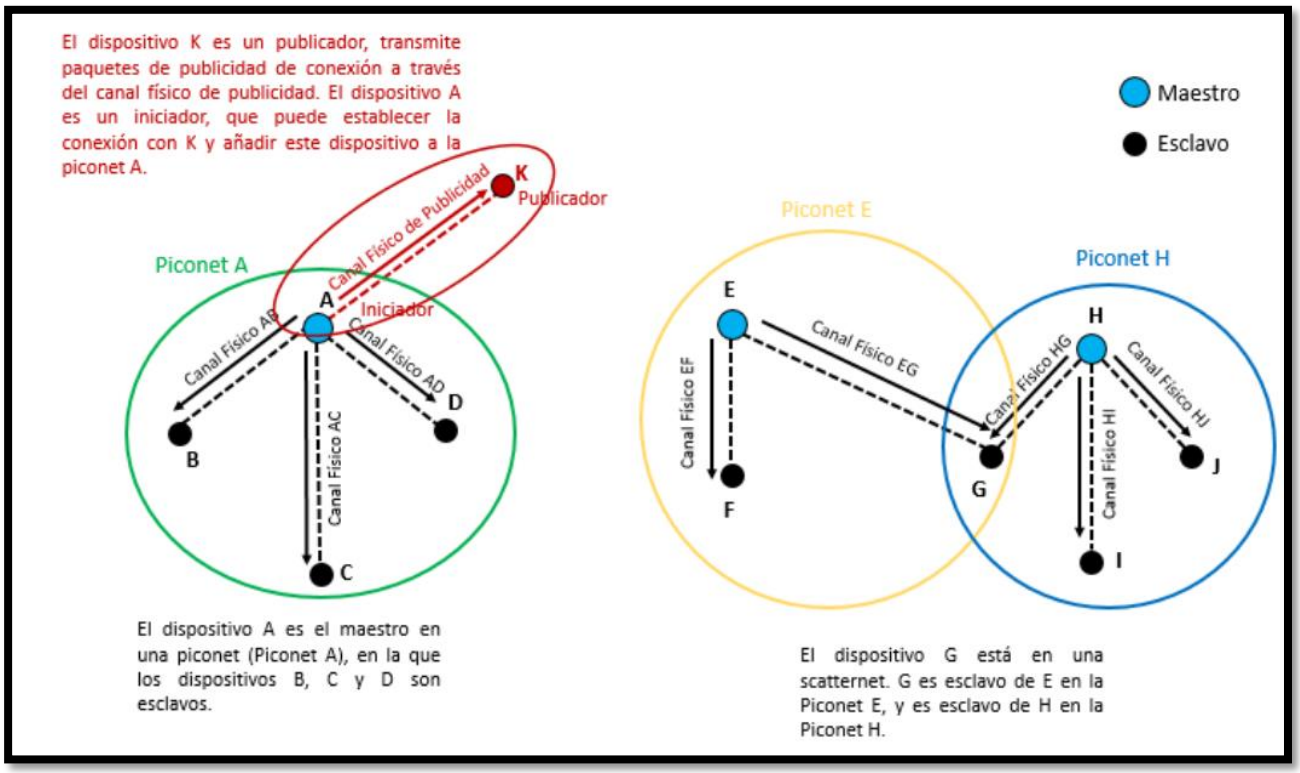


Ilustración 11. Topología Bluetooth LE. [8]

4.8. Arquitectura de seguridad en Bluetooth

En este apartado, se va a hablar de la arquitectura de seguridad que nos ofrecen los dispositivos Bluetooth. Básicamente, esto define algunas características y servicios de seguridad que el Bluetooth es capaz de proporcionarnos. [8]

- **Generación de claves**: Las diferentes claves criptográficas, utilizadas en el intercambio de información entre dispositivos, se generan durante un proceso denominado **pairing** (emparejamiento). En este proceso, se lleva a cabo la autenticación entre dos dispositivos, y después, se establecen una o varias claves secretas compartidas entre los dispositivos.
- **Bonding**: Lo que este proceso permite, es almacenar las claves generadas en el pairing. Esto, permitirá a los dispositivos poder hacer uso de las mismas en futuras conexiones entre ellos.
- **Autenticación de dispositivos**: En este proceso, se lleva a cabo que dos dispositivos poseen las mismas claves compartidas.
- **Cifrado**: Aquí se emplean una serie de algoritmos criptográficos para que exista confidencialidad en la información cuando los dispositivos intercambian información.
- **Integridad**: Se emplean diferentes mecanismos que hacen que se la información sea protegida ante posibles alteraciones ilícitas.
- **Privacidad**: Bluetooth dispone de un mecanismo para ocultar las direcciones de los dispositivos, con el objetivo de que alguien sin autorización, pueda realizar un seguimiento al dispositivo y, por lo tanto, a su propietario.
- **Firma de datos**: Bluetooth dispone de un mecanismo que permite autenticar los datos enviados a través de una conexión entre dos dispositivos, en aquellas situaciones en la que la conexión no implementa cifrado.

4.8.1. Seguridad en BR/EDR

En este apartado se van a comentar algunos mecanismos de seguridad, pero que son implementados en los dispositivos BR/EDR. Estos mecanismos, se llevan a

cabo en la parte del controlador BR/EDR a nivel de Gestor den Enlace (Link Manager). Estos mecanismos han ido evolucionando a lo largo de las versiones Bluetooth.

- **BR/EDR Legacy Pairing**: Se trata de un método de emparejamiento, pero solo implementado hasta la versión 2.0 (incluida). Actualmente se considerá obsoleto.
- **BR/EDR Secure Simple Pairing (SSP)**: Método utilizado a partir de la versión 2.1 hasta la versión 4.0 (incluida). Introduce algoritmos basados en curva Elíptica [12] (ECDH o Eliptic Curve Diffie-Hellman) que permiten que en el momento de establecer una o varias claves en común, estas sean generadas en uno de los dispositivos y no son distribuidas a través de la conexión.
- **BR/EDR Secure Connections**: Método implementado a partir de la versión 4.1. Esto mejora la seguridad de SSP utilizando algoritmos ECDH más evolucionados y haciendo uso de AES [13]para cifrar.

A continuación, mostramos un diagrama de los mecanismos de seguridad de BR/EDR SSP y Secure Connections. Para ver más detalles al respecto, consultar el Bluetooth Core Specifications. [14]

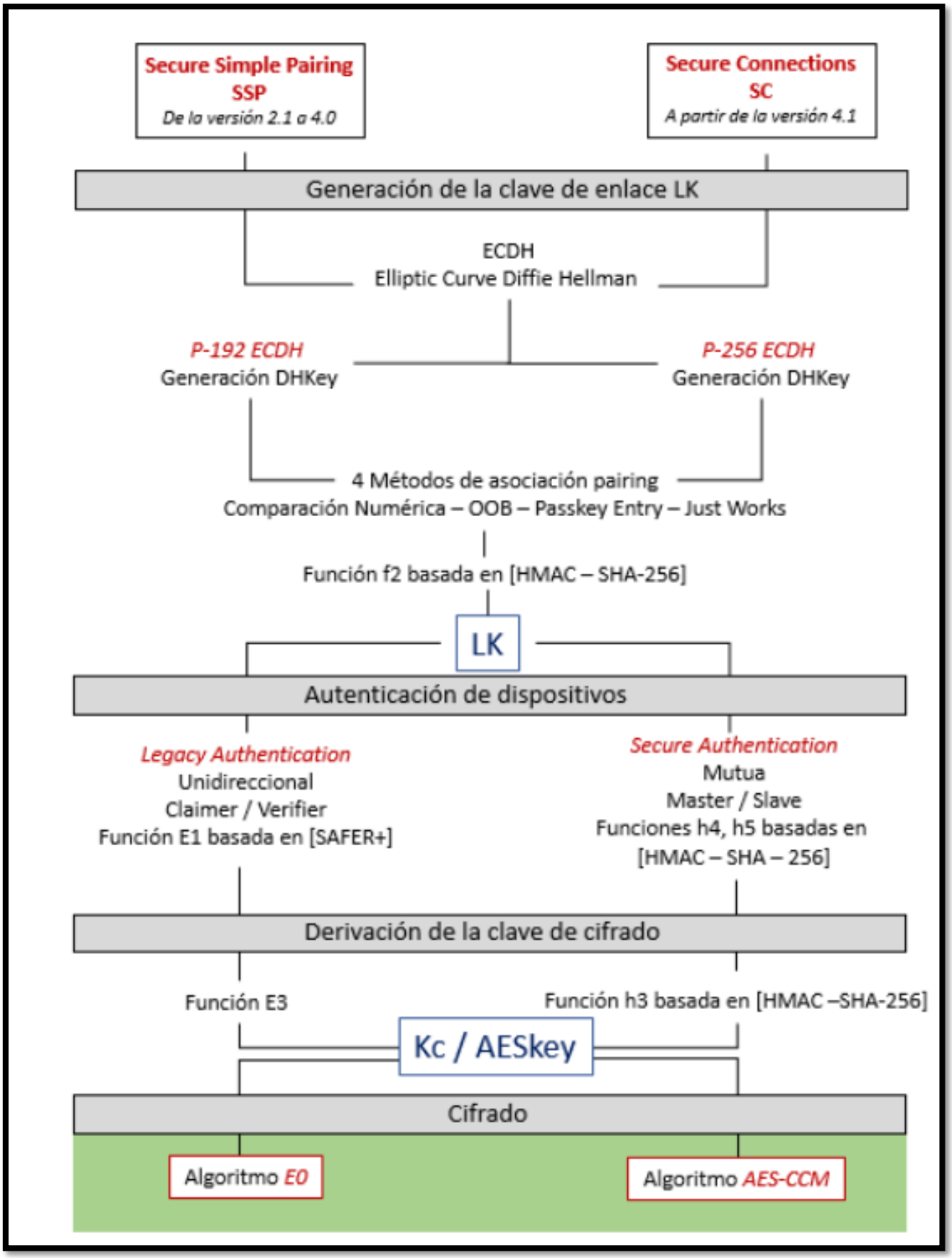


Ilustración 12. Mecanismos de Seguridad Bluetooth BR/EDR. [8]

Modos de seguridad

Los modos de seguridad en Bluetooth, determinan en qué fase del establecimiento de la conexión entre distintos dispositivos, se inicial los mecanismos de autenticación y cifrado. [8]

- **Modo 1**: No inicia ningún mecanismo de seguridad.
- **Modo 2**: Estos mecanismos se inician después del establecimiento del enlace. En este modo, el gestor de seguridad local es el encargado de controlar el acceso a los diferentes servicios.
- **Modo 3**: Los mecanismos de seguridad se inician antes de que se complete el enlace físico. Los dispositivos que operan en este modo, establecen la autenticación y el cifrado de forma obligatoria para cada una de las conexiones.
- **Modo 4**: Este modo se introdujo a partir de la versión 2.1 + EDR. En este modo, los mecanismos de seguridad se inician después de establecer el enlace físico y lógico. Este modo usa algoritmos más avanzados para general la clave de enlace (LK), como son los algoritmos de curva elíptica (ECDH).

4.8.2. Seguridad en Bluetooth LE

La seguridad utilizada en los dispositivos Bluetooth LE es **gestionada por el Security Manager** (SM o Gestor de Seguridad) de la parte Host.

El Security Manager, define el protocolo y el modo de llevar a cabo el proceso de emparejamiento, la autenticación y el cifrado en dispositivos que sólo disponen de tecnología LE o que disponen de BR/EDR/LE.

Con Bluetooth LE, se introducen dos características nuevas de seguridad para la tecnología Bluetooth: **LE Privacy** y **Data Signing**.

LE Privacy **(Privacidad LE)**

Los dispositivos Bluetooth emiten una serie de mensajes, para anunciar su presencia a los demás dispositivos. Estos mensajes contienen diversos elementos de información, entre ellos a dirección del mismo dispositivo Bluetooth, que lo

identifica de forma unívoca. Esto podría hacer que un dispositivo no autorizado, podría capturar estos paquetes y hacer un seguimiento de este dispositivo.

Esta función, propia de Bluetooth LE, es una salvaguarda que hace que la dirección del dispositivo dentro del mensaje anteriormente mencionado, sea sustituido por un valor aleatorio que va cambiando cada cierto periodo de tiempo. De esta forma, aunque alguien no autorizado se haga con estos mensajes, no podrá relacionarlos con un mismo dispositivo.

Pero, ¿cómo sabe un dispositivo legítimo la dirección real? Para conseguir esto, se hace uso de una clave llamada **IRK** (Identity Resolving Key), que es intercambiada entre los dispositivos durante el proceso de pairing y que permite a un dispositivo traducir ese valor "aleatorio" contenido en el mensaje delo dispositivo emisor. [8]

Data Signing **(Firma de datos)**

Se trata de una característica de seguridad que va a permitir autenticar la información enviada entre dos dispositivos, que hayan sido emparejados de forma correcta anteriormente. Esto es muy útil en aquellos escenarios donde el modo de conexión entre dispositivos no implemente cifrado.

Los mecanismos de seguridad en este tipo de dispositivos, al igual que en Bluetooth LE, también han ido cambiando y mejorando con el tiempo, adaptándose en función de la tecnología. Estos se pueden agrupar en los siguientes:

- LE Legacy Pairing: Mecanismo usado en las versiones 4.0 y 4.1
- LE Secure Connections: Mecanismo utilizado a partir de la versión 4.2.

A continuación, se muestra una figura con un diagrama a modo de resumen de estos mecanismos de seguridad. Para más información, consultar la especificación del core de Bluetooth. [14]

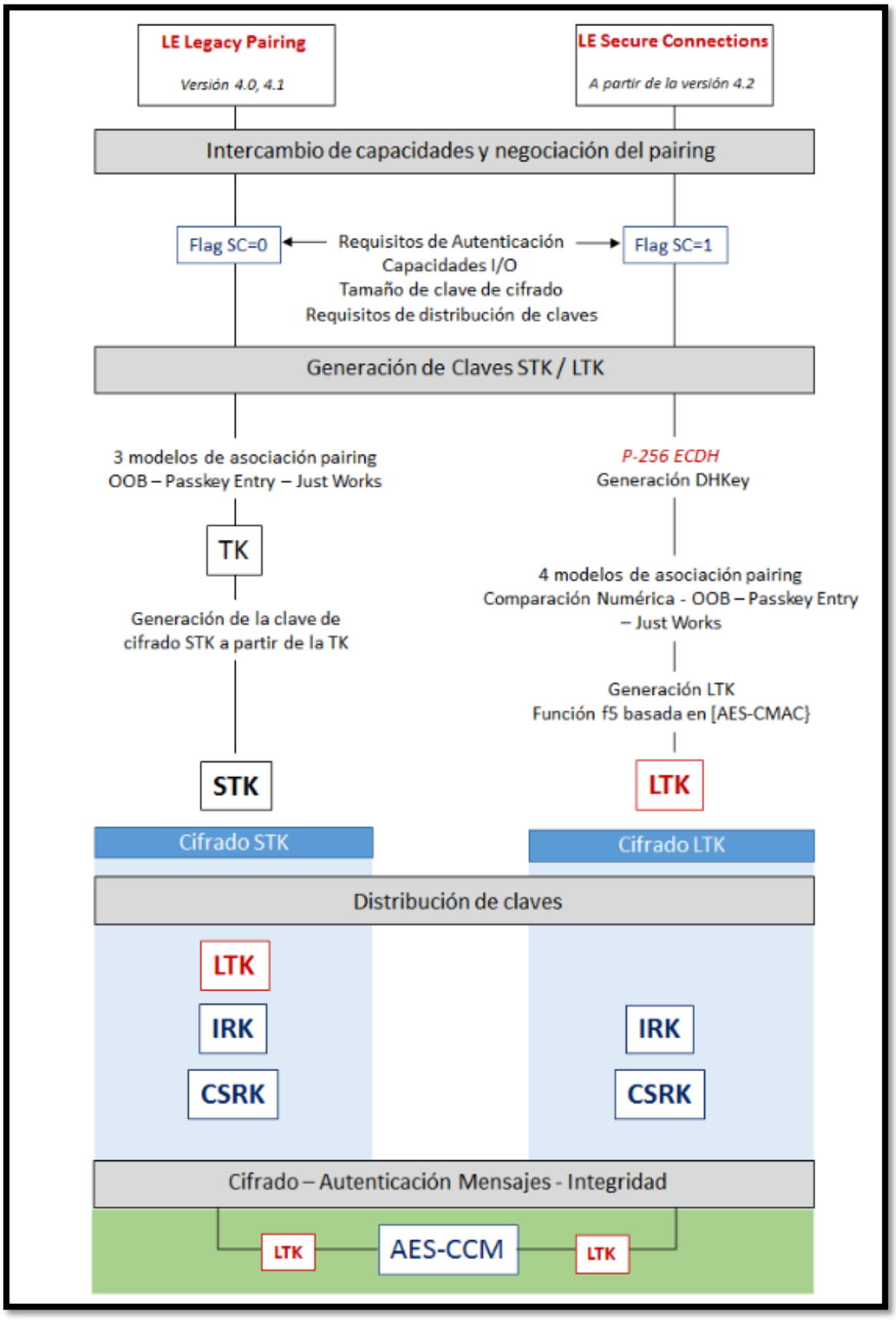


Ilustración 13. Mecanismos de Seguridad Bluetooth LE. [8]

4.9. Vulnerabilidades y Ataques en Bluetooth

4.9.1. Introducción

En este apartado, se comenta de forma breve y a modo de retrospectiva, un histórico de las vulnerabilidades detectadas en las diferentes versiones de Bluetooth y que posteriormente se fueron solucionando. Cuando hablamos de lo vulnerable que es una conexión Bluetooth entre dispositivos, el factor de la versión Bluetooth es probablemente el más importante, puesto que, en la seguridad de las comunicaciones entre dispositivos, dicha comunicación es tan fuerte, como lo es el eslabón más débil. Por tanto, muchas de las vulnerabilidades ya encontradas siguen siendo estando presentes ya que, en la actualidad, muchos dispositivos Bluetooth con versiones antiguas, se siguen utilizando.

Posteriormente, se hará lo mismo, pero comentando una serie de ataques comunes a dispositivos Bluetooth.

4.9.2. Vulnerabilidades Bluetooth en las distintas versiones [3]

Versiones anteriores a Bluetooth 1.2

Las claves de enlace que se utilizan para llevar a cabo el emparejamiento entre dispositivos, se basan en claves de unidad estática y se reutilizan en cada emparejamiento. Esto hace posible que, si un atacante lograra hacerse con dichas claves, podría escuchar las conexiones del dispositivo original y, además, podría suplantar al dispositivo original o cualquiera conectado a él.

Versiones anteriores a Bluetooth 2.1 + EDR

- Se utilizan códigos PIN de longitud insuficiente, que pueden ser fácilmente adivinados por un posible atacante.
- No existe una gestión de códigos PIN.
- El flujo de claves se repite después de 23,3 horas de uso, por lo que si una conexión dura más de este periodo de tiempo, se utiliza exactamente el mismo flujo de claves. Esto permitiría a un atacante, descifrar todos los mensajes en la conexión.

Versiones 2.1 y 3.0

- Los dispositivos, que son capaces de trabajar con un Modo 4 de seguridad, pueden utilizar versiones anteriores cuando se conectan a otros que no son compatibles con ese mismo modo, inclusive el modo de seguridad 1, que no tiene ningún tipo de seguridad. Esto hace que pueda conectarse con cualquiera, pero que sea más vulnerable.
- Hace uso de claves estáticas, por lo que es vulnerable a ataques MITM (Man In The Middle).

Versiones anteriores a 4.0

- El número de solicitudes de impugnación de la autenticación es ilimitado, por lo que un atacante podría recopilar muchas respuestas para obtener información de la clave de acceso.
- La función de cifrado utilizada, se considera débil a día de hoy

Todas las versiones

- Si las claves de enlace no se almacenan como es debido, un posible atacante podría verlas e incluso modificarlas. Esto podría provocar multitud de ataques contra el dispositivo y cualquiera que se conectase a él.
- Existe la posibilidad de que la clave de encriptación que se utilice, sea tan pequeña como 1 byte.
- No existe autenticación de usuario. Solo existe autenticación entre dispositivos.
- Un dispositivo puede permanecer como visible o como detectable de forma indefinida. Esto hace que, si el usuario no se da cuenta de desactivarlo, pueda ser seguido por un atacante.

Para obtener información acerca de todos los **CVE (Common Vulnerabilities and Explosures)** acerca de Bluetooth, puede consultarlo en su página oficial utilizando “Bluetooth” como palabra clave de búsqueda. [15]

Además de esto, la **Guía de Seguridad Bluetooth del NIST** [16] (National Institute of Standards and Technology) proporciona una tabla con estas y otras vulnerabilidades en diferentes versiones Bluetooth.

4.9.3. Ataques comunes en Bluetooth

En este apartado, como se ha comentado anteriormente, se va a comentar de forma breve y a grandes rasgos, algunos de los ataques más comunes y populares que se pueden llevar a cabo contra dispositivos Bluetooth. Cabe destacar, que los problemas de seguridad que siempre ha tenido y tiene Bluetooth, **se deben en gran medida al proceso de emparejamiento** entre dispositivos y muchos de los ataques que vamos a comentar, están diseñados para ser utilizados durante el mismo.

Eavesdropping

Básicamente, se trata de escuchas pasivas del tráfico de información, intercambiado entre dos dispositivos Bluetooth. Por ejemplo, aunque solo se trate de escuchas, podría llevarse un ataque contra unos auriculares Bluetooth de un ejecutivo. Esto nos permitiría obtener las conversaciones en los que sean utilizados y se podría obtener información muy sensible.

MAC Spoofing Attack

Este ataque se lleva a cabo antes de que se establezca el cifrado entre dos dispositivos, y durante la formación de la Piconet, cuando las claves de enlace están siendo generadas. En este punto, los dispositivos se autentican el uno al otro mediante estas claves de enlace. Por tanto, durante el ataque, el atacante puede hacerse pasar por los usuarios legítimos. Este usuario malicioso podría así, comenzar y terminar conexiones, interceptar información, modificarla, etc. [17]

PIN Cracking Attack

Este ataque puede hacerse durante el proceso de emparejamiento y el proceso de autenticación. Para ello, el atacante utiliza un sniffer de frecuencias para obtener el RAND (número aleatorio utilizado para emparejar) y la BD_ADDR del dispositivo que se quiere atacar. Después, el usuario malicioso utiliza un algoritmo de fuerza bruta (Algoritmo E22) que es usado para combinar todas las posibles permutaciones del PIN de emparejamiento con la información que se ha obtenido anteriormente, hasta que ese PIN es encontrado. [17]

Man In The Middle

Este es uno de los ataques más comunes en las comunicaciones inalámbricas en general. En Bluetooth, sucede cuando dos dispositivos quieren emparejarse. Cuando esto va a suceder, no se comparten claves secretas hasta llegado a un punto del proceso de emparejado, esto permite al atacante situarse en medio de ambos dispositivos legítimos y cuando estos creen que la autenticación entre ellos ha sido satisfactoria, en realidad ambos están emparejados con el atacante. En la siguiente imagen, ilustramos cómo sería. [17]

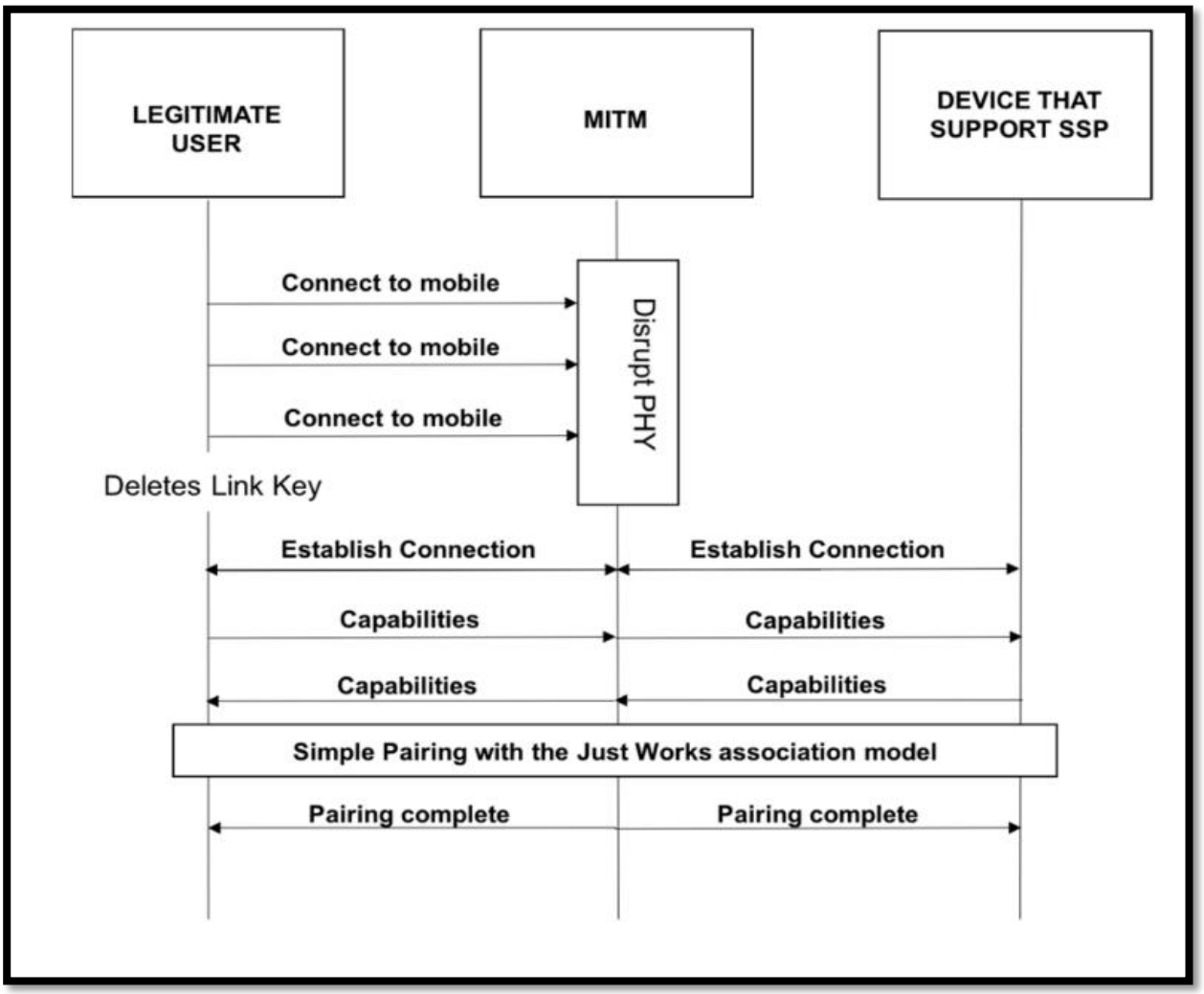


Ilustración 14. Ataque MITM. [17]

Bluejacking

Cuando se realiza este ataque, un usuario malicioso envía mensajes no solicitados a un dispositivo con el Bluetooth abierto, con la finalidad de engañar al usuario para que responda al mensaje o lo guarde en sus contactos. Este ataque se podría asemejar al Phising al spam del correo electrónico. Requiere de cierta colaboración de la víctima, para que tenga éxito. [17]

Bluesnarfing

Con este ataque, lo que se hace básicamente es explotar las vulnerabilidades propias del firmware de los dispositivos antiguos, de tal forma que un atacante puede acceder a datos almacenamos en el dispositivo de la víctima como agenda, calendario, IMEI, etc. [17]

Bluebugging

Al igual que sucede con el ataque anterior, este explota también las vulnerabilidades en el firmware de algunos dispositivos. En cambio, lo que un atacante puede obtener con este ataque es acceder al sistema operativo e interfaz de comandos. Con estos comandos, se podría desde escuchar llamadas, hasta extraer toda la información del dispositivo de la víctima. [17]

Denial of Service

Denial of Service (Denegación de servicio) es un ataque que consiste en el uso de un envío continuo de paquetes por parte de mensajes por parte de un atacante, con el fin de inhabilitar la interfaz de Bluetooth o agotar la batería de la víctima. Estos ataques suelen ser menos comunes que los vistos anteriormente. [17]

Fuzzing Attack

Consiste en enviar datos no estándar o mal formados a la interfaz de radio del dispositivo con el fin de observar el comportamiento de la víctima (si se ralentiza su funcionamiento, o paraliza su operación). Esto puede hacer que el atacante infiera vulnerabilidades en la pila de protocolos. [17]

Blueborne

Este ataque se lleva a cabo explotando un defecto de desbordamiento de búfer de pila. Al dirigir el procesamiento de las respuestas de configuración L2CAP pendientes del cliente, el atacante puede secuestrar las conexiones Bluetooth. Esto les permite controlar el contenido y las funciones integradas de un dispositivo específico. Para que el ataque tenga éxito, sólo se necesita la dirección MAC y un Bluetooth. [18]

Si se tienen dispositivos con el Bluetooth habilitado, simplemente con la información anterior, esos dispositivos serían potencialmente vulnerables al ataque.

Este ataque, fue desarrollado por la empresa de seguridad **Armis** [19] hace relativamente poco tiempo, por lo que muchos dispositivos actuales son vulnerables a este ataque. En nuestro caso práctico, veremos la cantidad de dispositivos que pueden llegar a ser vulnerables a este ataque.

4.9.4. Blueborne

En este apartado, a pesar de que se ha explicado brevemente **Blueborne**, queremos dar más detalles y centrarnos en este vector de ataque, porque en el caso práctico de este proyecto, van a ser identificados aquellos dispositivos que son potencialmente vulnerables a este vector de ataque.

Estamos ante un vector de ataque bastante reciente revelado por la empresa de seguridad **Armis** [19]**,** concretamente por la división **Armis Labs** que se encarga de investigar de forma continua vulnerabilidades de existan en distintas tecnologías. Este vector de ataque, pone en peligro los principales sistemas operativos móviles y de escritorio, incluidos IOS (en versiones anteriores a la 10), Android, Windows, Linux y los dispositivos que los integran. Junto con este vector de ataque, Armis reveló ocho vulnerabilidades **0-day** [20]**,** de las cuales cuatro estaban clasificadas como críticas.

Blueborne permite que un atacante pueda tomar el control absoluto de un dispositivo, con lo que podría acceder a todo tipo de información, redes corporativas, propagar malware, etc. [21]. Este vector de ataque no requiere que el dispositivo

objetivo esté emparejado con el atacante, ni siquiera es necesario que esté detectable.

Pero, ¿cuál es el grado de la amenaza? Según Armis, este vector de ataque puede afectar de forma potencial a todos los dispositivos que hagan uso de Bluetooth, que se estiman en más de 8.200 millones de dispositivos en la actualidad. Los últimos informes según cuenta Armis, muestran más de 2.000 millones con Android, 2.000 millones con Windows y 1.000 millones de dispositivos Apple en uso, y a eso hay que sumarle la cantidad de dispositivos IoT que tienen Bluetooth, dispositivos médicos, etc.

Armis se puso en contacto de forma coordinada con diferentes actores para asegurar una respuesta a las vulnerabilidades identificadas. Estos actores eran Google, Microsoft, Apple, Samsung, Linux…

Antes se ha comentado que los dispositivos de prácticamente todas las plataformas están afectados, pero ahora, veremos exactamente qué vulnerabilidades afectan a cada una de estas plataformas:

- **Android**: Todos los smartphones y weareables (excepto aquellos que solo utilizan Bluetooth Low Energy) en todas sus versiones se ven afectadas por cuatro vulnerabilidades encontradas en el sistema operativo Android. Dos de esas vulnerabilidades permiten ejecutar código remoto (**CVE-2017-0781** y **CVE-2017-0782**), otra, permite pueda robarse información (**CVE-2017-0785**) y la última permite a un atacante llevar a cabo un ataque MITM (**CVE-2017-0783**). Algunos de los dispositivos afectados son: Google Pixel, Samsung Galaxy, Samsung Galaxy Tab, LG Watch Sport.
- **Windows**: Todos los ordenadores desde Windows Vista están afectados por una vulnerabilidad denominada “Bluetooth Pineapple”, la cual permite a un atacante llevar a cabo un MITM (**CVE-2017-8628**).
- **Linux**: Se trata del sistema operativo que hay debajo de una gama muy amplia de dispositivos. Se ven afectados todos los dispositivos Linux que ejecutan la librería BlueZ, de la que hablamos posteriormente. La vulnerabilidad que afecta a este grupo de dispositivos, permite fugas de

información (CVE-2017-1000250). Además, todos los dispositivos Linux desde la versión 2.6.32 (publicada en julio de 2009), hasta la 4.14 están afectados por una vulnerabilidad que permite la ejecución de código remoto (CVE-2017-1000251). Como ejemplo de dispositivos afectados, tenemos el Samsung Gear S3, cuyo sistema operativo es Tizen (que corre sobre Linux) o las Smart TVs de Samsung.

- **IOS**: Todos los iPhone, iPad y iPod Touch con una versión igual o inferior a iOS 9.3.5 y las AppleTV con una versión de su software igual o inferior a 7.2.2, están afectados por una vulnerabilidad que permite ejecutar código remoto (CVE-2017-14315).
- **Amazon Echo y Google Home**: Estos dispositivos fueron identificados como vulnerables a Blueborne. Puede consultar el impacto de este ataque en los asistentes de voz [22].

Para llevar a cabo **Blueborne**, un atacante tiene que pasar por diversas etapas. En primer lugar, el atacante tendría que localizar conexiones Bluetooth activas que tenga a su alrededor. Los dispositivos, pueden ser identificados incluso si no se encuentran “visibles” (Posteriormente, se expone un hardware que permitiría hacer esto). Una vez hecho esto, el atacante obtiene la dirección del dispositivo (puede ser denominada MAC o BTADRR). Una vez identificada esta dirección, el atacante puede determinar el sistema operativo que está utilizando la víctima y preparar un exploit que se ajuste a eso. El atacante explotará una vulnerabilidad del protocolo para actuar sobre el dispositivo de la víctima. Después, tendrá que determinar el tipo de ataque que realiza dependiendo de sus objetivos (extraer información, tomar el control del dispositivo…). Para ver todos los detalles sobre Blueborne, podemos consultar el **paper** donde se explica con todo detalle. [23]

4.10. Librerías para trabajar con Bluetooth

En este apartado, se habla de las librerías encontradas para poder trabajar con hardware Bluetooth. Esto es importante, puesto que como ya dijo al principio, a la hora de llevar a cabo un desarrollo que utilice Bluetooth, es inviable hacerlo a bajo nivel, por el conocimiento y tiempo de desarrollo que conlleva. Esta búsqueda en el estudio teórico ya que es algo que se ha hecho antes de hacer ningún desarrollo,

aunque posteriormente, se hará uso de estas librerías (todas o algunas) en para desarrollar el software que vamos a utilizar el caso práctico.

BlueZ

Es la pila oficial de protocolos Bluetooth en Linux y un proyecto de código abierto distribuido bajo la Licencia Pública General de GNU (GPL). BlueZ [24] es una librería a bajo nivel (concretamente, en el lenguaje C), programada para poder interactuar con el hardware de Bluetooth. Esta librería proporciona soporte para las diferentes capas y protocolos del mismo. Es flexible, muy eficiente y está implementada de forma modular.

PyBlueZ

Se trata de una librería escrita en Python, que utiliza los recursos Bluetooth con el objetivo de permitir a los desarrolladores que utilicen este lenguaje, el poder crear aplicaciones que hagan uso de Bluetooth de una forma mucho más sencilla y rápida. Podemos hacer uso de esta librería en Windows, Linux y MacOS está disponible de forma gratuita bajo la Licencia Pública General de GNU. [25]

PyGattlib

Esta es una librería, también para Python, que sirve para hacer uso del protocolo GATT, que permite trabajar con dispositivos Bluetooth LE. Básicamente, se trata de una abstracción de la implementación de este protocolo, en la librería BlueZ. [26]

BluePy

Esta no es una librería como tal, sino que se trata de una Interfaz para Python, que nos permite trabajar de una forma mucho más sencilla con dispositivos LE en Linux [27]. Esta es una interfaz que hace uso de las librerías de Python anteriormente mencionadas

Web Bluetooth

Se trata de una API para programación web, que permite descubrir y comunicarse con dispositivos Bluetooth, pero solo aquellos de la versión 4, haciendo uso de GATT. [28]

libblepp

Esta es una implementación de las funciones que se puede llevar a cabo con Bluetooth LE escrita en C++ y que no hace uso de BlueZ, por lo que es totalmente independiente de la misma. [29]

Bluetooth - Manager

Es una librería o más bien framework, que puede ayudar a gestionar diferentes dispositivos y adaptadores Bluetooth y Bluetooth LE. Está diseñado para que el trabajo con el protocolo Bluetooth sea más fácil de realizar y que se haga de una forma más sencilla a más alto nivel. Esta librería está escrita y se utiliza para el lenguaje Java. [30]

4.11. Ubertooth One

4.11.1. ¿Qué es?

Durante nuestra investigación acerca del Bluetooth, se ha realizado una búsqueda de algún hardware que pudiese ser útil a la hora de utilizar la tecnología Bluetooth para obtener información. Así, es como dimos con el **Ubertooth One**.

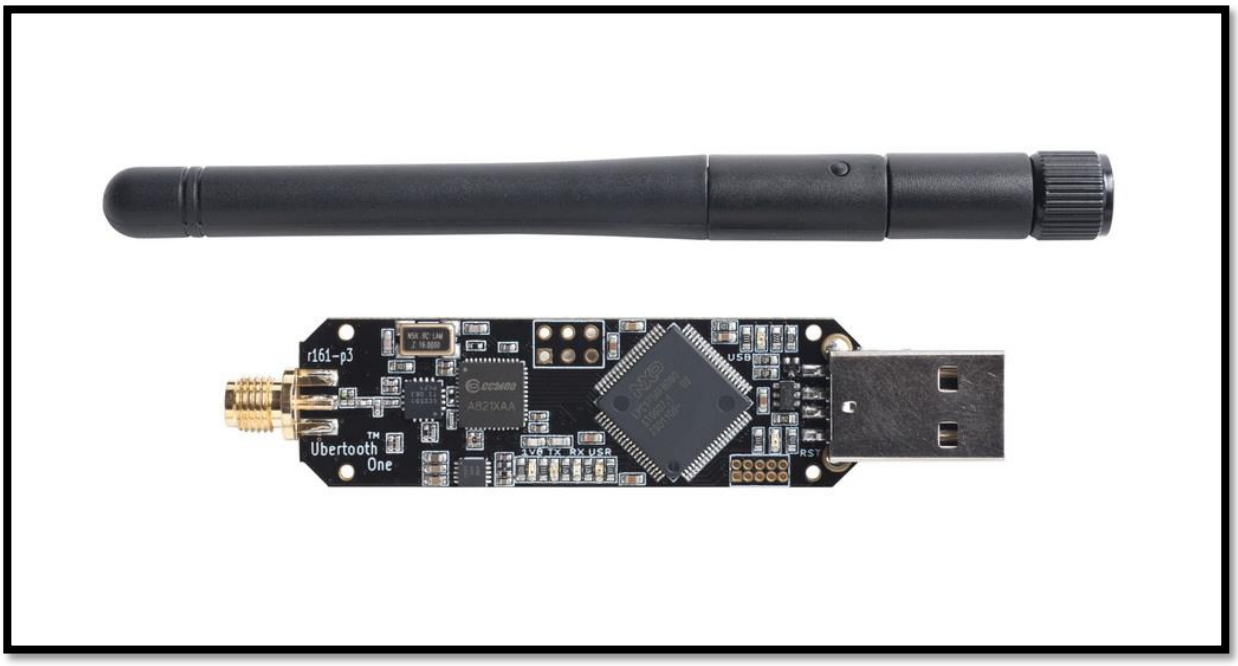


Ilustración 15. Ubertooth One. [36]

Ubertooth One se podría definir como una plataforma de código abierto para tecnología inalámbrica, que trabaja en la frecuencia de 2,4 GHz, y que está pensada para la experimentación con Bluetooth. Anteriormente tuvo un predecesor, el **Ubertooth Zero**. [31]

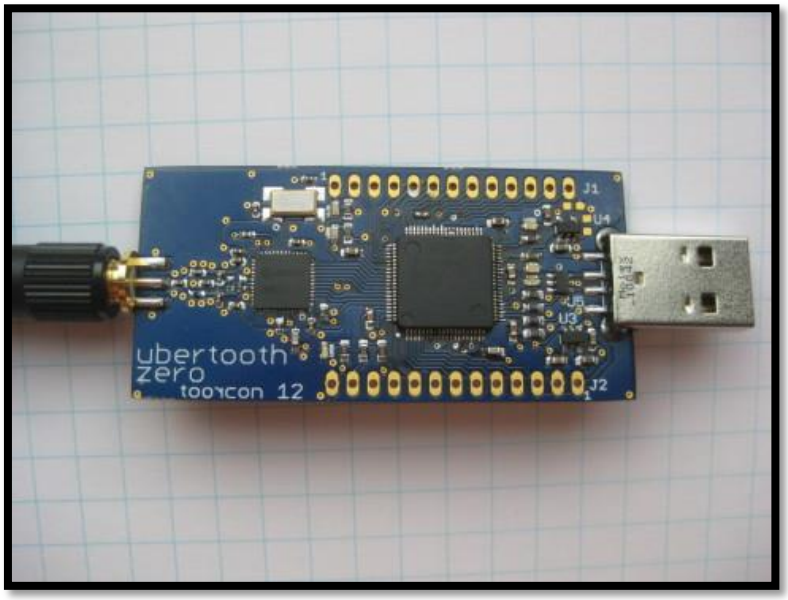


Ilustración 16. Ubertooth Zero. [31]

Los equipos de monitorización con Bluetooth comerciales, suelen ser muy herméticos, puesto que solo los modifica quien los fabrica, y además suelen ser muy caros. **Ubertooth One** fue diseñado para ser una plataforma alternativa asequible la monitorización y desarrollo de nuevas tecnologías BT, BLE, etc…

Este dispositivo, fue creado por **Mike Ossmann** [32] en **Great Scott Gadgets** [33] en 2011 (Ubertooth Zero en 2010). Surgió cuando este no encontró un adaptador Bluetooth que cubriera las necesidades que él tenía. En YouTube, podemos encontrar un video de su charla en ShmooCon 2011, donde cuenta toda la historia. [34]

Se trata de un hardware de código abierto y todas las especificaciones hardware y demás información del producto, la podemos encontrar en su repositorio de GitHub. [35]

Este dispositivo está diseñado principalmente como un **receptor Bluetooth**, pero como un receptor **más avanzado que los tradicionales** que podemos encontrar en cualquier dispositivo de nuestro día a día. Este dispositivo tiene capacidades superiores al de los adaptadores tradicionales, puesto que nos permite hacer uso de él para monitorizar y detectar señales Bluetooth. Aunque actualmente, el firmware solo soporta las características de recepción y transmisión del canal de publicidad (Advertising, del que hemos hablado con anterioridad).

4.11.2. Arquitectura

En este apartado, vamos expondremos de forma breve, los componentes con los que ha sido fabricado este dispositivo.

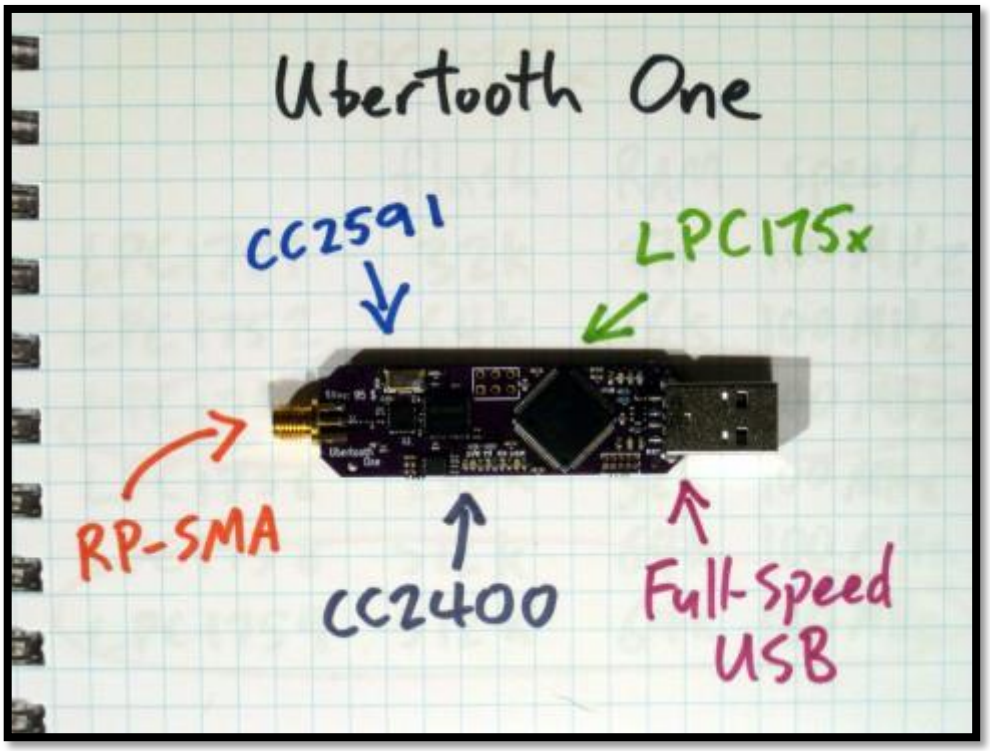


Ilustración 17. Ubertooth One. [36]

- **RP-SMA**: Conector SMA de radiofrecuencia, que podemos utilizar para conectar a un equipo o conectarle una antena (se suele usar principalmente para esto).
- **CC2591**: Se trata de una interfaz de radiofrecuencia de alto rendimiento, para diferentes aplicaciones en la banda de los 2,4 GHz de baja potencia y bajo voltaje

- **CC2400**: Se trata de un transceptor de radiofrecuencia de 2,4 GHz de un solo chip. Este transceptor, está integrado con un módem de banda base que soporta velocidades de datos de hasta 1Mbps.
- **LPC175x**: Se trata de una familia de microcontroladores basados en el Cortex-M3 de ARM para sistemas embebidos, con un alto nivel de integración y bajo consumo de energía. [37]
- **USB**: Conecta el dispositivo a un equipo que va a ejecutar código sobre él.

4.11.3. Características

Las características del **Ubertooth One** son la siguientes [36]:

- Transmisión y recepción en 2,4 GHz.
- Potencia de transmisión y recepción con una sensibilidad comparable a la de un dispositivo Bluetooth Clase 1.
- Conector de depuración de corteza estándar (JTAG de 10 pines y 50 milímetros).
- Conector serial de programación en el sistema (ISP).
- Conector de expansión: destinado a la comunicación interurbanas u otros usos futuros.
- seis LEDs indicadores.

4.11.4. Pines y LEDs

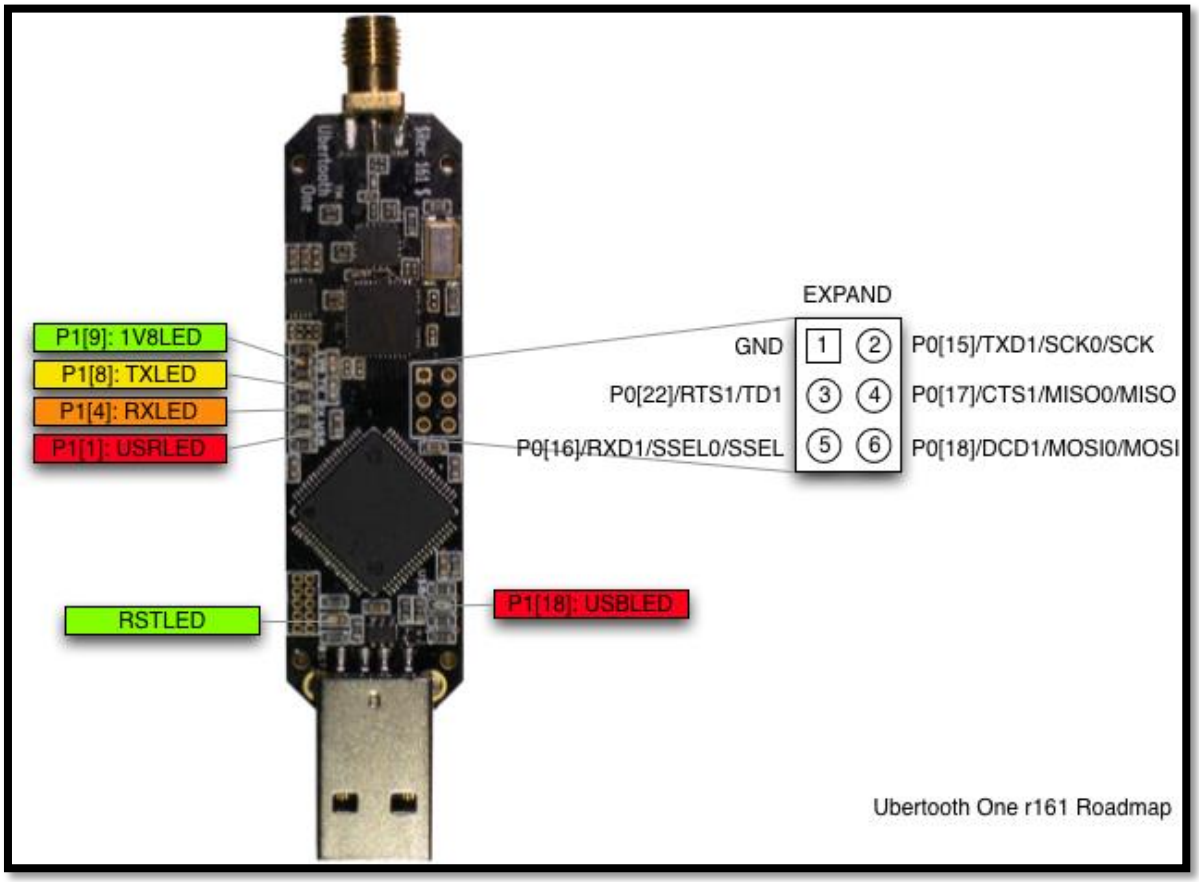


Ilustración 18. Pines y LEDs Ubertooth One. [36]

En este esquema, se muestran la localización de los siguientes LEDs y pines del Ubertooth One. [36]

- **RSTLED**: Indica que el LPC175x se encuentra encendido. Cuando el dispositivo se encuentra en funcionamiento, este LED siempre debe estar activo, excepto durante el restablecimiento completo del LPC175X (por ejemplo, al entrar en modo ISP).
- **1V8LED**: Indica que el CC2400 está siendo alimentado con 1,8 V. El control de este suministro de corriente al transceptor, depende del firmware. Se requiere alimentación de 1,8 V para activar el oscilador de cristal que se requiere para activar el USB.
- **USBLED**: Indica que el USB se ha configurado correctamente.
- **TXLED**: Normalmente indica transmisión de radio. El control de este LED depende del firmware.

- **RXLED**: Normalmente indica la recepción de radio. El control de este LED también depende del firmware.
- **USRLED**: Usado por el usuario según sus necesidades para que indique lo que este quiera. Esto lo hará mediante el firmware.

Los LEDs TX, RX y USR parpadean en un patrón distintivo cuando el bootloader está listo para aceptar comandos USB DFU.

4.11.5. Cómo conseguirlo

Al ser de código abierto, cualquiera que esté interesado podría acceder a construirlo y usarlo, puesto que toda la información la tenemos disponible en internet. [38, 35, 36]

Pero si no alguien no tiene los conocimientos o el tiempo suficiente para construirlo, está disponible para comprar en diversos sitios de internet. [39]

Por desgracia, para nuestro proyecto no se ha podido adquirir este hardware, aunque parecía adecuado destacarlo en este documento, puesto se trataba de un hardware muy interesante, muy relacionado con nuestro proyecto y, en caso de usarse, puede potenciarlo mucho

5. CASO PRÁCTICO

5.1. Introducción

Es en este apartado, donde se comienza a desarrollar el caso práctico que con anterioridad mencionamos brevemente.

Para esto, en primer lugar, se ha tenido que llevar a cabo el desarrollo de un software, haciendo uso de hardware Bluetooth, para poder capturar información de diferentes dispositivos en una o varias zonas, con el fin de analizar esos datos a posteriori, y ver qué conclusiones se pueden obtener sobre esos mismos datos.

5.2. Desarrollo de software

5.2.1. Introducción

En esta memoria no se va a detallar nada referente a la planificación y plazos del desarrollo, puesto que se trata principalmente de un proyecto de investigación que mezcla el estudio teórico de una tecnología (Bluetooth en este caso), con un pequeño caso práctico para lo que no consideramos necesario que se detallen todos estos datos. En el caso de que se tratara de un posible proyecto de más envergadura y centrado principalmente en lo que el software es capaz de hacer, sí que sería necesario y recomendable detallar la planificación y presupuesto aproximado que llevaría a cabo el desarrollo del software.

En definitiva, aquí expondremos se ha desarrollado este software, las tecnologías y medios utilizados, además de las características y funcionalidades del mismo

5.2.2. Tecnologías software y librerías para el desarrollo

En este apartado del proyecto, se hablará de las tecnologías y librerías que se van a utilizar para el desarrollo del software en nuestro proyecto, con las que se pretende que sea más fácil y rápido hacer ese desarrollo.

Python

Es el lenguaje de programación seleccionado para el desarrollo del software. Son diversos los motivos por lo que se ha decidido utilizar este lenguaje. En primer lugar, se trata de alto nivel, muy intuitivo, fácil de aprender para aquellas personas que no lo hayan utilizado antes y hasta de instalar en tu equipo.

Ilustración 19. Símbolo Python [40]

Otra razón por la que se ha seleccionado este lenguaje, es porque se puede utilizar prácticamente en cualquier plataforma (Windows, Linux, MacOS) y de una forma muy sencilla y rápida. Esto permite que se pueda hacer uso del software que desarrolle (tanto desarrollo como ejecución) sobre diferentes equipos y diferentes módulos hardware.

También, se trata de un lenguaje muy versátil y es por eso que hoy en día es uno de los lenguajes más utilizados del mundo en prácticamente casi cualquier campo. Además, tiene una comunidad muy grande, y existen multitud de librerías y software desarrollado (algunas se han visto en apartados anteriores) para que no se parta desde la nada a la hora de hacer un software propio.

PyBlueZ [25], PyGattlib [26] y BluePy [27]

Estas son las librerías de Python seleccionadas para trabajar con Bluetooth y van a hacer que sea más sencillo y rápido que trabajar a bajo nivel con el hardware. Estas librerías hacen uso de una librería de más bajo nivel llamada BlueZ, que está programada en C y de las que ya se ha hablado anteriormente. Para saber más, consultar el apartado 4.10 de este documento.

MySQL

Sistema de gestión de bases de datos que se va a utilizar en el supuesto práctico. Se trata de SGBD que se encuentra bajo una licencia doble: Licencia pública general y Licencia comercia por Oracle Corporation. Esta base de datos está considerada como la más popular mundialmente entre las que son de código abierto. Además, también lo es entre las que no lo son junto con Microsoft SQL Server, sobre todo para los entornos de desarrollo web. [41]

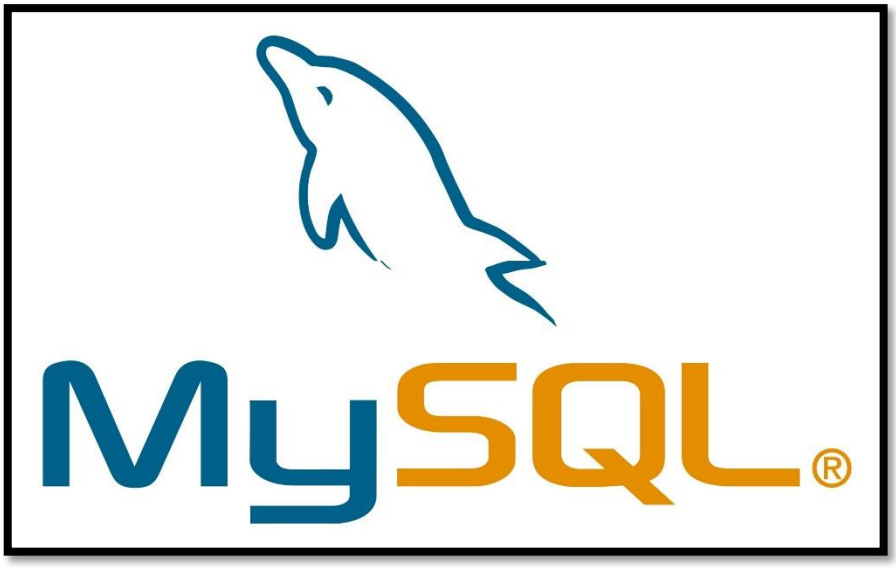


Ilustración 20. MySQL Logo. [42]

Este SGBD, va a servir para almacenar todos los datos que sean capturados haciendo uso del software.

DataGrip

Datagrip es un magnífico IDE desde el que es posible manipular de forma gráfica varias bases de datos desde una misma ventana. Se puede utilizar tanto para bases de datos locales como remotas.

Ilustración 21. DataGrip Logo. [43]

Se trata de un software con multitud de funcionalidades útiles como la ejecución de consultas en distintos modos, además, proporciona un historial en local

con el registro de toda la actividad que se lleva a cabo para así no perder el trabajo que hemos realizado, entre otras muchas.

Este es un software de pago, perteneciente a la empresa **Jetbrains** [44], que desarrolla una serie de IDEs muy recomendables para desarrollar en diferentes lenguajes y de uso profesional entre los que se encuentra Datagrip. Aunque es un software de pago, al ser estudiante se puede adquirir una licencia renovable cada año (si sigues siendo estudiante) para usarlos todos de forma gratuita en su versión completa. [45]

Pycharm

Anteriormente se mencionó que vamos a desarrollar nuestro software en el lenguaje de programación Python, y para ello, se va utilizar del IDE de desarrollo **PyCharm** [46].

Ilustración 22. PyCharm Logo. [46]

Este es el IDE para programar en Python que hace la empresa **JetBrains** [44] de la que ya hemos hablado antes. Se ha elegido esta opción debido a que es un software muy profesional, con un excelente Debugger, porque es fácil de utilizar e instalar en cualquier plataforma y porque se tenía experiencia previa utilizando este software. Además, como se ha comentado antes, por el hecho de ser estudiante, se puede adquirir una licencia de uso gratuita en su versión completa.

5.2.3. Requisitos del software

Se ha buscado que el software se ajuste a nuestras necesidades, es decir, aquellas características que sean necesarias para llevar a cabo el caso práctico

En primer lugar, el software debe ser capaz de estar buscando de forma constante los dispositivos Bluetooth abiertos cercanos, por lo que debe estar trabajando desde el momento que lo ejecutamos, hasta que decidimos pararlo. Además, debe ser capaz de obtener el mayor de información disponible sobre aquellos dispositivos que encuentre (Nombre, dirección Bluetooth o BTADRR, fabricante…) y guardarla en una base de datos.

Además de la información de la que se ha hablado, en este caso práctico se pretende demostrar lo peligroso que puede ser llevar el Bluetooth siempre visible y es por ello, que este software software, además de obtener la información indicada anteriormente, será capaz de identificar aquellos dispositivos que son o pueden ser vulnerables al vector de ataque **Blueborne** del que se habló con anterioridad.

5.2.4. Diseño de la base de datos

Anteriormente, hemos dicho que los datos sobre dispositivos Bluetooth que capturemos con nuestro software, vamos a almacenarlos en una base de datos. En este apartado, queremos explicar el diseño de tablas que va a tener esa base de datos, y detallar qué información exactamente van a almacenar dichas tablas.

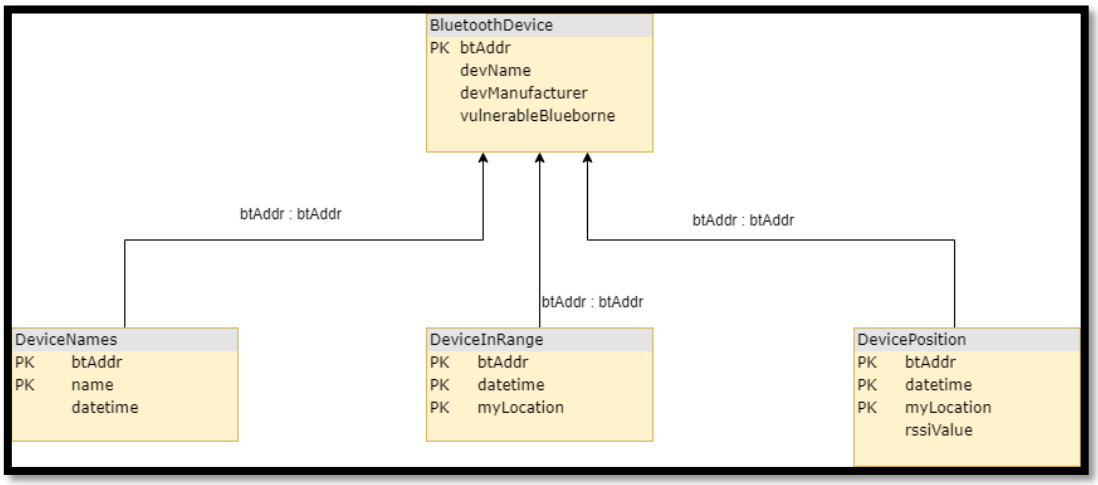


Ilustración 23. Diagrama E/R Bluetooth Database.

En la imagen anterior se puede observar que se trata de una base de datos sencilla pero que cumple con las necesidades y nos permite almacenar los datos que se pretenden utilizar. A continuación, se detalla el contenido de las tablas y explicando cada uno de los atributos.

Tabla BluetoothDevice

Esta tabla representa un dispositivo Bluetooth capturado con el software y cuyos atributos son:

- **btAddr**: Representa la dirección que identifica de manera unívoca a cada dispositivo y, por tanto, este atributo será la Primary Key (Clave Primaria) de esta tabla.
- **devName**: Este atributo de tabla representa el nombre del dispositivo (generalmente establecido por el usuario). En el caso de no localizar ningún nombre, se almacenará la cadena de texto **'UNKNOWN'.** Con este atributo, cabe la posibilidad de que se detecte un dispositivo con un nombre y posteriormente se vuelva a detectar, pero con un nombre diferente (por ejemplo, porque el usuario haya decidido cambiarlo). Cuando esto pasa, en este atributo se almacena el último nombre detectado, aunque esto no significa que se deseche el nombre anterior. Esto se verá en la tabla **DeviceNames**.
- **devManufacturer**: Este atributo almacenará el fabricante (si es que logramos identificarlo). En el caso de no identificar el fabricante, al igual que sucedía con el atributo anterior, se almacenará la cadena de texto **'UNKNOWN'**. El fabricante no se detecta al escanear los dispositivos, sino que lo hacemos utilizando unas listas predeterminadas de fabricantes y la dirección BTADDR. Se verá al explicar el funcionamiento del software.
- **vulnerableBlueborne**: Este es un atributo **Booleano** que va a indicar si el dispositivo Bluetooth es vulnerable o potencialmente vulnerable al vector de ataque **Blueborne** (ya se dijo con anterioridad, que se iba a utilizar en el caso práctico). Esto servirá a posteriori para ver cuantos dispositivos vulnerables se pueden llegar a localizar.

DeviceNames

En esta tabla, se va a almacenar los distintos nombres que puedan tomar los dispositivos. Como ya se dijo antes, con el tiempo, el usuario puede cambiar este valor, y dado esto, decidimos almacenar todos los nombres de cada dispositivo de una tabla de la base de datos. Los atributos de la tabla son los siguientes:

- **btAddr**: Representa la dirección que identifica de manera unívoca a un dispositivo Bluetooth. Este atributo **forma parte de la Primary Key** (Clave Primaria) de esta tabla. A su vez, es **Foreign Key** (Clave Foránea) de esta misma tabla, puesto que hace referencia al atributo de su mismo nombre en la tabla **BluetoothDevice**.
- **name**: Representa un nombre que tiene o ha tenido en algún momento un dispositivo Bluetooth con la dirección btAddr. Este atributo también forma parte de la **Primary Key** ya que, si se localiza el mismo dispositivo con el mismo nombre en dos ocasiones diferentes, solo se almacenará una vez.
- **datetime**: Marca temporal que indica cuándo fue localizado un dispositivo y con un nombre concreto la primera vez.

DeviceInRange

Esta tabla se va a usar para detectar aquellos **dispositivos que se encuentran en rango**, pero posiblemente no estén visibles. Esto solo se podrá hacer cuando previamente se haya capturado la dirección del dispositivo en cuestión y este sea de una versión más antigua y permita esto. La mayoría de dispositivos actuales no permiten hacerlo por razones de seguridad, pero se ha incluido en este proyecto para que, de alguna forma, sume en lo que sea posible a los datos de dispositivos abiertos o visibles.

- **btAddr**: Representa la dirección que identifica de manera unívoca a un dispositivo Bluetooth que se está. Este atributo **forma parte de la Primary Key** (Clave Primaria) de esta tabla. A su vez, es **Foreign Key** (Clave Foránea) de esta misma tabla, puesto que hace referencia al atributo de su mismo nombre en la tabla **BluetoothDevice**.

- **datetime**: Marca temporal de cuando ese dispositivo fue localizado en rango. Este atributo también forma parte de la **Primary Key** de esta tabla.
- **myLocation**: Valor adicional que se ha introducido y puede indicar la localización del dispositivo que escanea. Esto podría ser útil en el caso de que, por ejemplo, se estén escaneando dispositivos de manera continua y se produzca un desplazamiento. En ese caso, podría, no solo interesar la información del dispositivo capturado, sino desde qué lugar se realizó esa captura. En este proyecto no será de mucha utilidad, ya que los escaneos se llevarán a cabo desde posiciones fijas. Este atributo también forma parte de la **Primary Key** de esta tabla.

DevicePosition

Aquí está la tabla que va a contener los datos que más van a ser de utilidad junto con los de la tabla **BluetoothDevice**. Aquí tendremos los datos de los diferentes escaneos de los dispositivos Bluetooth abiertos o visibles.

- **btAddr**: Representa la dirección que identifica de manera unívoca a un dispositivo Bluetooth escaneado. Este atributo **forma parte de la Primary Key** (Clave Primaria) de esta tabla. A su vez, es **Foreign Key** (Clave Foránea) de esta misma tabla, puesto que hace referencia al atributo de su mismo nombre en la tabla **BluetoothDevice**.
- **datetime**: Marca temporal de cuando ese dispositivo fue localizado en rango. Este atributo también forma parte de la **Primary Key** de esta tabla.
- **myLocation**: Valor adicional introducido puede indicar la localización del dispositivo que escanea. Su función sería la misma que en la tabla **DeviceInRange**. Este atributo también forma parte de la **Primary Key** de esta tabla.
- **rssiValue**: Valor de RSSI (Received Signal Strength Indicator o Indicador de intensidad de señal recibida) del dispoistivo Bluetooth respecto del dispositivo que estamos escaneando. **RSSI** [47] es una medida estimada del nivel de potencia que un dispositivo de radiofrecuencia está recibiendo de un punto de acceso.

5.2.5. Elementos y ficheros en la estructura de directorios del software

En este apartado, se expone qué elementos existen en los directorios y ficheros más importantes y de los que se han hecho uso, de la estructura de directorios de nuestro software. Se hace esto, debido a que existen varios ficheros con scripts, datos y pequeños programitas que, en su conjunto, son los que permiten llevar a cabo el escaneo de dispositivos, ver cuáles son vulnerables, identificar fabricantes, insertar en la base de datos, etc… Y es por esto, que se va a detallar el contenido de cada uno de ellos.

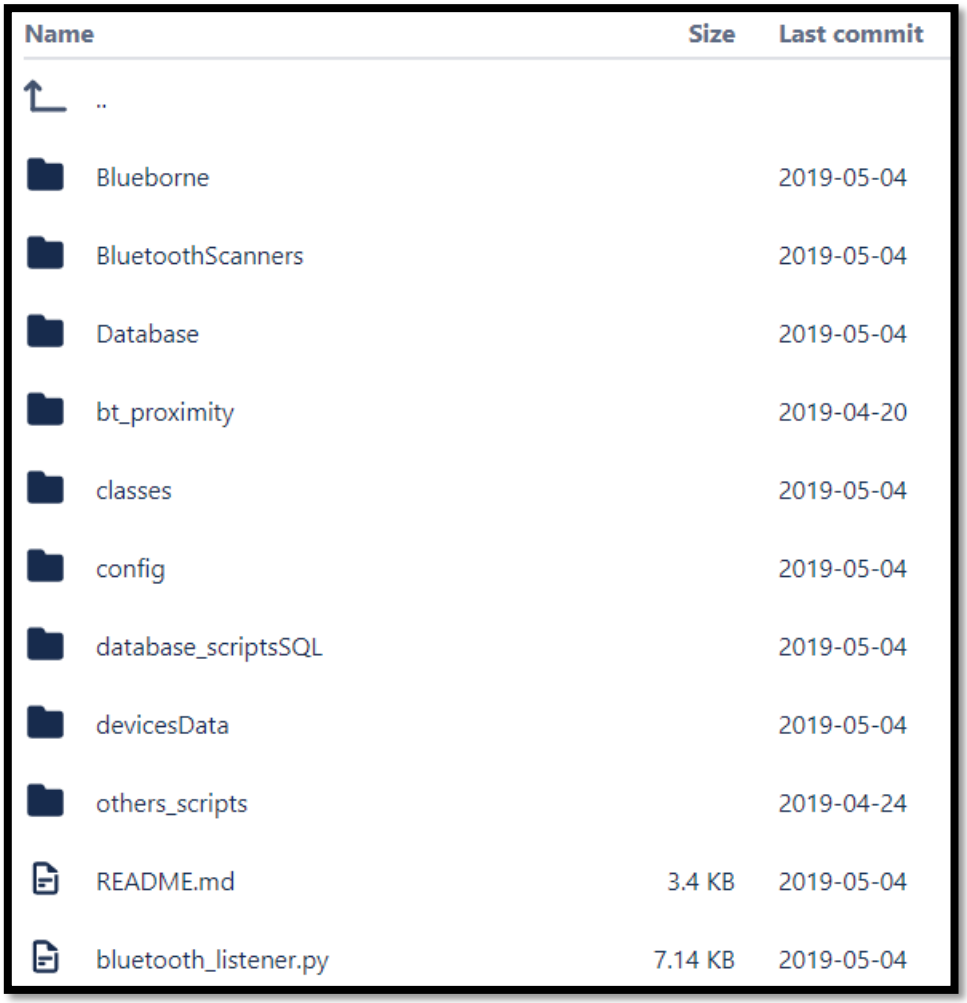


Ilustración 24. Estructura de directorios del software

Directorio “Blueborne”

Este directorio contiene una contiene datos relacionados con el vector de ataque **Blueborne**. Concretamente existe un fichero llamado **devicelist.py**, en el que hay una lista parte de direcciones BTADRR o MAC de los dispositivos que serían potencialmente vulnerables a este vector de ataque. A continuación, se ve una imagen parcial de cómo es esta lista.

```
#LISTADO DE DISPOSITIVOS VULNERABLES A BLUEBORNE
class devices:
    global deviceMACOUIs
    deviceMACOUIs = {}
    deviceMACOUIs["ANDROIDS"] = {}

    deviceMACOUIs["ANDROIDS"]["ALCATEL"]    = { "00:07:72", "00:11:8B", "00:16:4D", "00:17:CC", "00:1A:F0", "00:1C:8E", "00:1D:4C",
    deviceMACOUIs["ANDROIDS"]["APPLE"]      = { "00:03:93", "00:0A:27", "00:0A:95", "00:0D:93", "00:10:FA", "00:11:24", "00:14:51",
    deviceMACOUIs["ANDROIDS"]["ASUS"]       = { "00:0C:6E", "00:0E:A6", "00:11:2F", "00:11:D8", "00:13:D4", "00:15:F2", "00:17:31",
    deviceMACOUIs["ANDROIDS"]["GOOGLE"]     = { "00:1A:11", "08:9E:08", "3C:5A:B4", "48:D6:D5", "54:60:09", "70:3A:CB", "94:95:A0",
    deviceMACOUIs["ANDROIDS"]["HTC"]        = { "00:60:98", "00:09:2D", "00:23:76", "00:EE:BD", "04:C2:3E", "18:87:96", "1C:B0:94",
    deviceMACOUIs["ANDROIDS"]["HUAWEI"]     = { "00:18:82", "00:1E:10", "00:25:68", "00:25:9E", "00:2E:C7", "00:34:FE", "00:46:4B",
    deviceMACOUIs["ANDROIDS"]["LENOVO"]     = { "00:12:FE", "14:36:C6", "14:9F:E8", "50:3C:C4", "60:D9:A0", "6C:5F:1C", "70:72:0D",
    deviceMACOUIs["ANDROIDS"]["LG"]         = { "00:1C:62", "00:1E:75", "00:1F:6B", "00:1F:E3", "00:21:FB", "00:22:A9", "00:24:83",
    deviceMACOUIs["ANDROIDS"]["MEIZU"]      = { "D8:6C:02", "38:BC:1A", "68:3E:34", "90:F0:52" }
    deviceMACOUIs["ANDROIDS"]["MOTOROLA"]   = { "00:0E:C7", "00:20:75", "00:24:37", "00:24:92", "14:1A:A3", "14:30:C6", "1C:56:FE",
    deviceMACOUIs["ANDROIDS"]["ONEPLUS"]    = { "94:65:2D", "C0:EE:FB" }
    deviceMACOUIs["ANDROIDS"]["OPPO"]       = { "00:22:DE" }
    deviceMACOUIs["ANDROIDS"]["SAMSUNG"]    = { "00:02:78", "00:07:AB", "00:12:47", "00:12:FB", "00:13:77", "00:15:99", "00:15:B9",
    deviceMACOUIs["ANDROIDS"]["SONY"]       = { "00:0A:D9", "00:0E:07", "00:0F:DE", "00:12:EE", "00:16:20", "00:16:B8", "00:18:13",
    deviceMACOUIs["ANDROIDS"]["VIVO"]       = { "08:23:B2", "10:F6:81", "18:E2:9F", "1C:DA:27", "20:5D:47", "28:FA:A0", "34:E9:11",
    deviceMACOUIs["ANDROIDS"]["XIAOMI"]     = { "00:9E:C8", "00:EC:0A", "04:B1:67", "0C:1D:AF", "10:2A:B3", "14:F6:5A", "18:59:36",
    deviceMACOUIs["ANDROIDS"]["ZTE"]        = { "00:15:EB", "00:19:C6", "00:1E:73", "00:22:93", "00:25:12", "00:26:ED", "00:4A:77",
```

Ilustración 25. Listado dispositivos vulnerables a Blueborne.

Como se ve en la imagen anterior, esta lista va a ser doblemente útil, puesto que, no solo va a identificar aquellos dispositivos potencialmente vulnerables, sino que va a identificar el fabricante del dispositivo. Esta lista no es de elaboración propia, sino que se ha obtenido de otro desarrollo. [48]

Directorio “BluetoothScanners”

Este directorio contiene diferentes implementaciones software de **Scanners Bluetooth**. Dentro de este directorio tenemos otros dos, donde se separan estas implementaciones de las que se está hablando. Son los siguientes:

- **Bluetooth_LE**: Este directorio contiene el fichero **scanner_bluepy.py**, el cual contiene el código que va a ayudar a escanear dispositivos Low Energy.

- **Bluetooth_No_LE**: Este directorio contiene el código necesario para realizar el escaneo de dispositivos, pero a diferencia del apartado anterior, sirve para escanear **dispositivos Bluetooth tradicionales** (no BLE).

Dentro del directorio **BluetoothScanners**, además de los directorios anteriormente comentados, existe otro fichero llamado **scan.py**. El código de este fichero es un scanner que, haciendo uso del código de los otros directorios, logra escanear dispositivos Bluetooth tradicionales y LE. Al ejecutar la función **scannerBluetooth()** existente en este fichero, se lanza este escáner Bluetooth en un bucle infinito, por lo que siempre se están buscando dispositivos.

Directorio "Database"

Este directorio contiene un único fichero denominado **database_connect.py**. Este fichero contiene código que ayuda a crear conexiones con la base de datos haciendo uso de la librería **mysql connector**.

Para poder hacer esto, se ha creado una clase propia llamada **MysqlConnect**. Lo que se ha querido representar con esta clase, es la información de una base de datos para poder conectar de una forma rápida y sencilla con ella. *En* la siguiente imagen se puede ver cómo está implementada.

```
class MysqlConnect(object):
    '''
    Clase que nos permite crear una conexión a la base de datos
    '''
    def __init__(self, _user = cnf.DATABASE_PARAMS.get('user'),
                       _password = cnf.DATABASE_PARAMS.get('password'),
                       _host = cnf.DATABASE_PARAMS.get('host'),
                       _database = cnf.DATABASE_PARAMS.get('database')
                 ):
        self.user = _user
        self.password = _password
        self.host = _host
        self.database = _database
        self.connect = mysql.connector.connect(user = self.user,
                                password = self.password,
                                host = self.host,
                                database = self.database)
```

Ilustración 26. Clase MysqlConnect

Como se observa, en esta clase hay diversos atributos, que son los necesarios para llevar a cabo una conexión a una base de datos **mysql**.

- **user**: Usuario de la base de datos
- **password**: contraseña para el **user** que va a permitir el acceso a la base de datos.
- **host**: dirección IP del equipo donde se encuentra la base de datos. En este caso, la base de datos es local.
- **database**: nombre de la base de datos.
- **connect**: Objeto de la clase que representa una conexión con la base de datos. Esta conexión se crea en el mismo instante que el objeto, por lo tanto, solo sería necesario invocarlo para realizar cualquier operación con la base de datos.

Además de esta clase, en el fichero se incluyen diversos métodos, que permiten extraer e insertar información sobre los dispositivos Bluetooth capturados en la base de datos. En el fichero, se puede consultar la implementación y los comentarios que detallan qué es lo que hace cada uno de ellos.

Directorio “Classes”

En esta ocasión se ha utilizado este directorio para albergar las clases que se hayan necesitado implementar para el software. En este caso, solo hay un fichero llamado **BluetoothDevice.py**. Dentro de este fichero es donde se define la clase **BluetoothDevice**. Esta clase representa lo que sería un dispositivo Bluetooth para nosotros, o más bien, la información que se va a manipular del mismo y que vamos a almacenar en la base de datos. Como se puede observar en la siguiente imagen, los atributos de esta clase, son muy similares a los de las tablas de la base de datos que comentamos con anterioridad.

```
class BluetoothDevice:
    '''
    Clase que hemos creado para unir las características de un dispositivo bluetooth necesarias
    para almacenar en la base de datos
    '''
    def __init__(self):
        self.btAddr = 0
        self.devName = ''
        self.devManufacturer = ''
        self.myLocation =  ''
        self.vulnerableBlueborne = False
        self.rssiValue = 0
        self.datetime = 0
```

Ilustración 27.Definición Clase BluetoothDevice.

Lo que representan exactamente estos atributos es:

- **btAddr**: Dirección única del dispositivo Bluetooth
- **devName**: Nombre del dispositivo
- **devManufacturer**: Fabricante del dispositivo
- **myLocation**: Localización desde la que se realizará
- **vulnerableBlueborne**: Valor Booleano que insdica si el dispositivo es vulnerable a Blueborne
- **rssiValue**: Valor de RSSI del dispositivo
- **datetime**: Marca temporal que indica cuándo ha sido detectado.

Directorio “config”

En este directorio están aquellos elementos que van a ser modificables en el programa. Es decir, se tratan de diversos parámetros utilizados por los demás ficheros para desarrollar sus funciones. Estos parámetros hacen referencia al identificador Bluetooth utilizado, tiempo de espera para realizar otra captura, parámetros de configuración de la base de datos, etc. Todo esto, se encuentra en un fichero denominado **config_params.py** y a continuación se muestra su contenido.

```
PARAMS = {
    'my_location': 'CARLOS BEDROOM',
    'scanner_device_id':  0,
    'listener_device_id': 0,
    'listener_sleep': 20,

    'absolute_project_path': '/root/Escritorio/tfm_bluetooth/Bluetooth_Carlos/'
         }

DATABASE_PARAMS = {
     'user' : 'Bluetooth_DB_admin',
     'password' : 'password',
     'host' : '127.0.0.1',
     'database': 'Bluetooth_DB'
}
```

Ilustración 28. Fichero de configuración 'config_params'.

Para ver que son exactamente cada uno de los parámetros, recomendamos acceder al mismo fichero donde se encuentran comentados todos y cada uno de ellos.

Directorio "databse_scriptsSQL"

Este directorio, contiene un fichero **script.sql**, que contiene las sentencias SQL necesarias para la creación de la base de datos. Con este fichero, cualquiera podría replicar la estructura de tablas en la base de datos de una forma muy rápida y sencilla.

Directorio "deviceData"

En este directorio hay, por un lado, 3 ficheros **.csv**, que contienen multitud de fabricantes de dispositivos junto con el inicio de diversas direcciones MAC. Estos listados son oficiales, públicos y disponibles para quien quiera descargarlos. [49]

Por otro lado, existe un fichero llamado **dev_information.py**. Se trata de un script que se ha desarrollado, para leer los ficheros **.csv** comentados anteriormente. Además, contiene un pequeño programita, que dada una dirección MAC, proporciona el fabricante del mismo haciendo uso de los listados. Esto será muy útil a la hora de identificar el fabricante de los dispositivos que se escaneen.

bluetooth_listener.py

Este fichero, contiene el que sería el programa principal del software. Es decir, es aquí donde se lleva a cabo un escaneo de forma ininterrumpida, de los dispositivos Bluetooth que se encuentran al alcance. Este escaneo se realiza en un **thread** (hilo) independiente, nada más comenzar la ejecución del programa. Después de iniciarse el hilo del escáner, el programa continúa su ejecución intentado detectar, si los dispositivos que ya están en la base de datos, están a nuestro alcance, aunque ya no se encuentren visibles. Esto último también se hace de forma ininterrumpida. Como ya se comentó con anterioridad, esto solo va a ser posible con los dispositivos más antiguos, ya que, por seguridad, los más modernos no nos lo permiten. En la siguiente imagen, se muestra a grandes rasgos de forma gráfica, el funcionamiento del programa **bluetooth listener**.

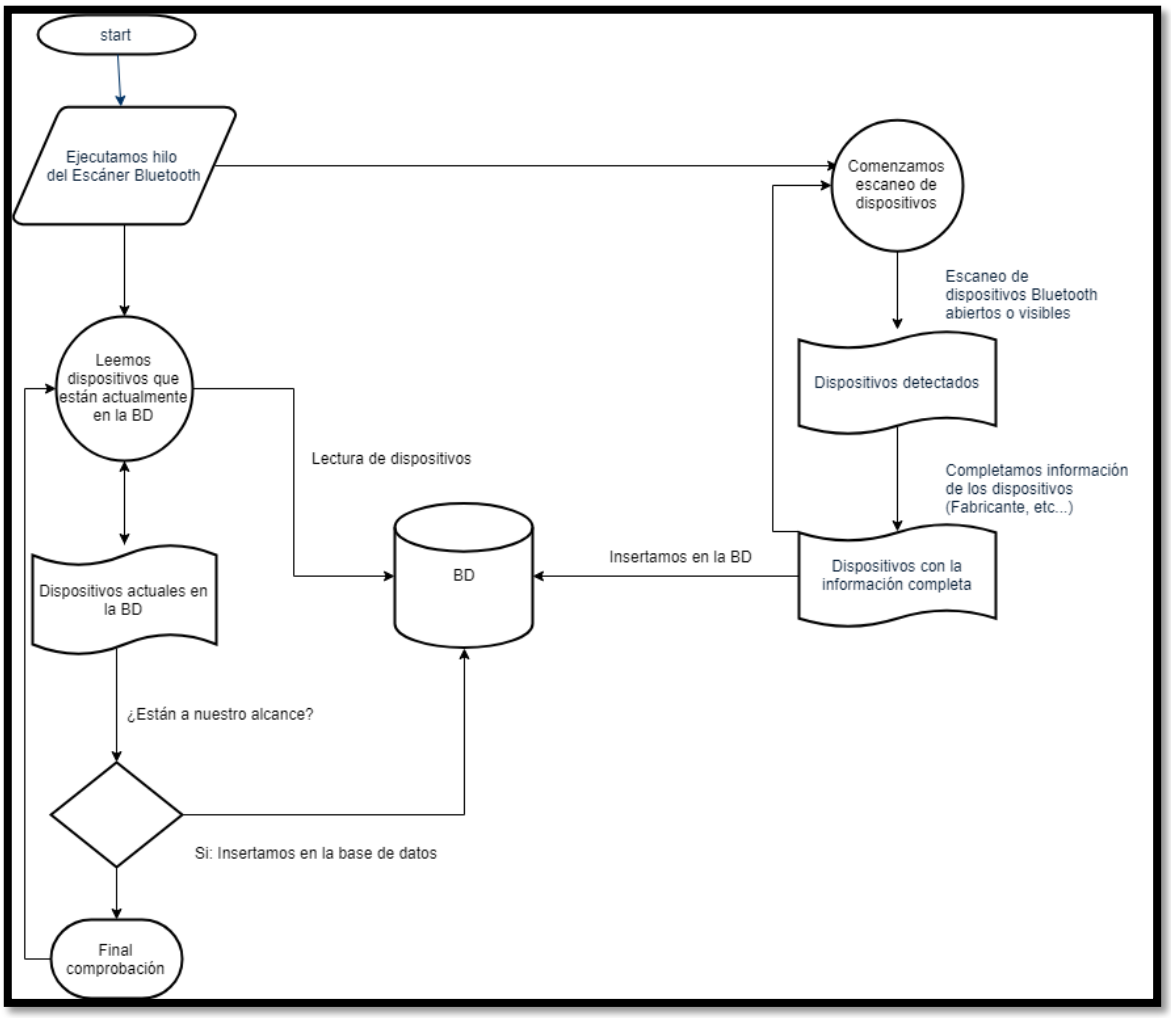


Ilustración 29. Flujo del programa bluetooth_listener.

5.2.6. Hardware y módulos sobre los que vamos a ejecutar el software

Para llevar a cabo el desarrollo del software, se ha utilizando un equipo portátil, haciendo uso de todas las herramientas software que se comentaron anteriormente. Sin embargo, a la hora de realizar el caso práctico como tal, es decir, la captura de dispositivos Bluetooth durante un tiempo prolongado y en diferentes lugares, se ha buscado una alternativa que cumpla una serie de características que son necesarias en este caso.

- Que se trate de un dispositivo pequeño y que sea fácil de transportar.
- Dispositivo que sea capaz de ejecutar sin problemas el software que se ha desarrollado haciendo uso de las diferentes librerías
- Que tenga Bluetooth
- Que sea “Plug and Play”, es decir, que simplemente con pulsar un botón o conectarlo, sea capaz de ejecutar el software y almacenas los datos Bluetooth que capture.

El dispositivo que mejor se ajustaba a todas estas características es la popular **Raspberry Pi**. [50] Concretamente, hemos decidido hacer uso de la **Raspberry Pi 3 Model B+**.

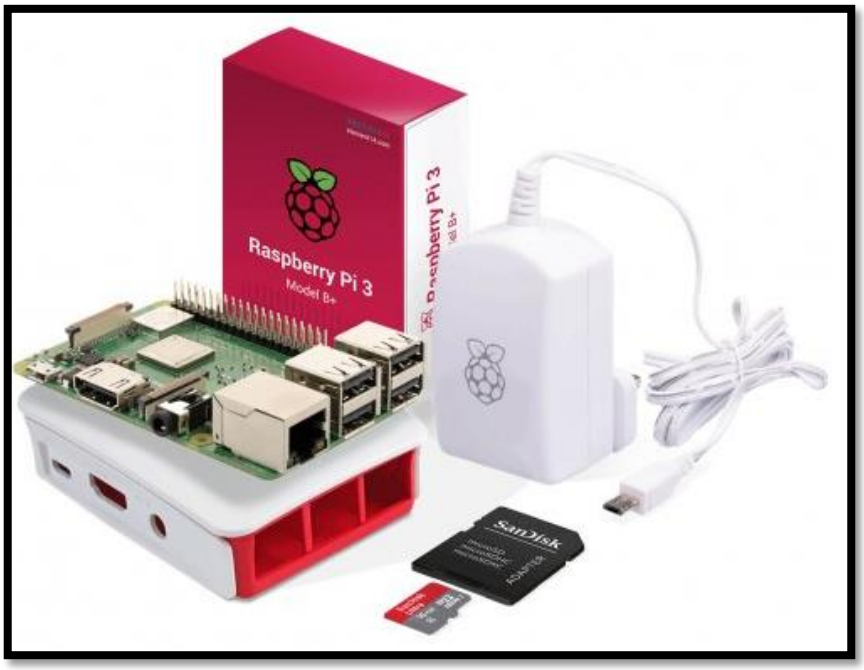


Ilustración 30. Raspberry Pi · Model B+ Started Pack [51]

Pero, ¿qué es una Raspberry Pi? [52]Para quien no la conozca, se trata de un ordenador pequeño, formado por una placa base, un procesador, memoria RAM y un pequeño chip gráfico. Este dispositivo fue lanzado por la **Fundación Raspberry Pi** [50] como un proyecto de educación con el fin de enseñar informática en las escuelas alrededor del mundo. A continuación, vemos los diferentes módulos que tenemos en este hardware.

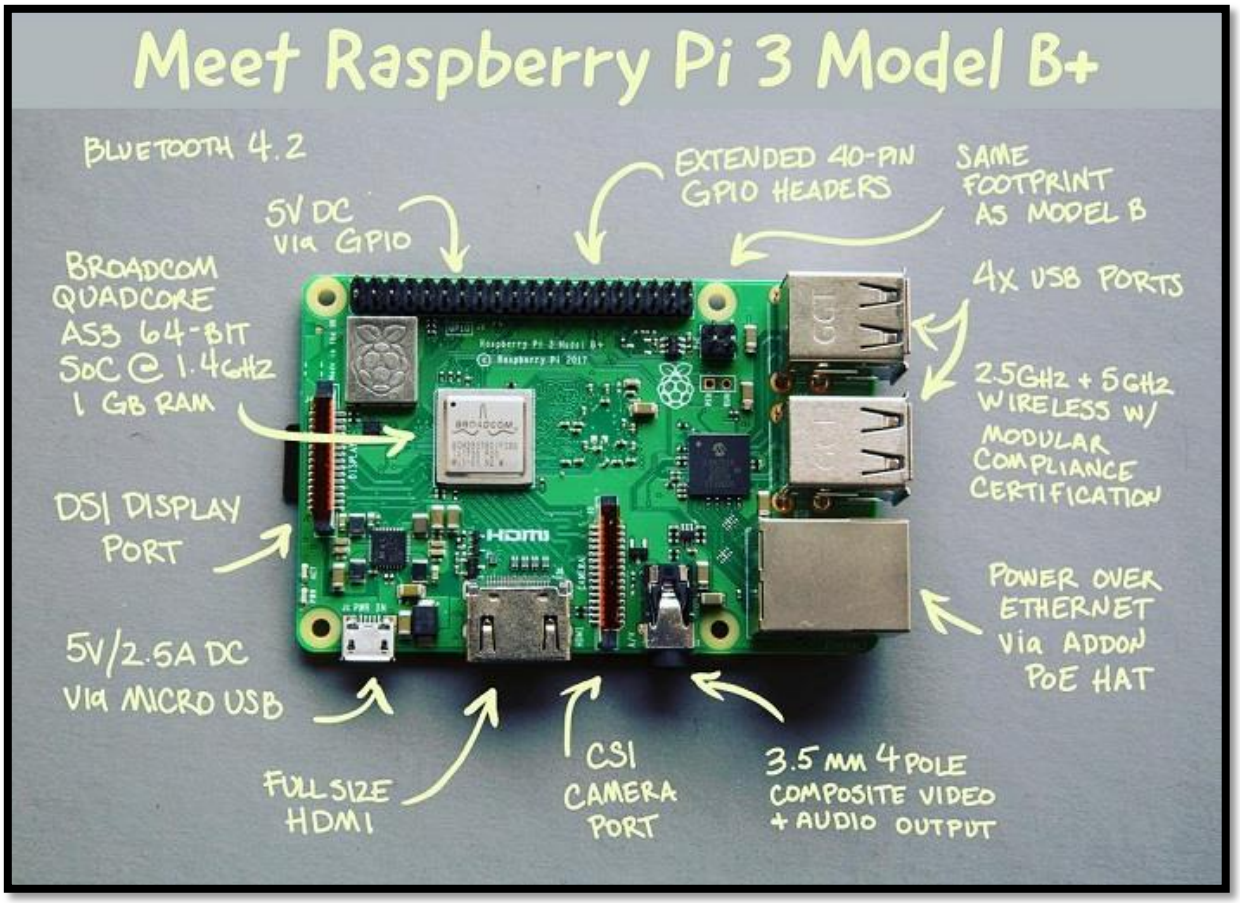


Ilustración 31. Raspberry Pi Model B+ modules. [53]

En el **ANEXO 1** de este documento, tenemos como usar el software que hemos desarrollado, junto con la Raspberry Pi.

5.3. Obtención de datos

5.3.1. Introducción

En este apartado, se va a exponer como se ha llevado a cabo la recopilación de datos sobre dispositivos Bluetooth. Se parte de un hardware con un software embebido. Se trata de una **Raspberry Pi**, **con el software** que se ha desarrollado

instalado en la misma. En concreto, no solo se ha trabajado con una Raspberry Pi, sino que se han usado dos de ellas, lo que va a permitir posicionarlas en diferentes sitios durante un periodo de tiempo. Una vez recopilados los datos capturados por estos dispositivos, se verá qué información y conclusiones se pueden obtener. En la siguiente imagen, se muestran las dos Raspberry Pi usadas para la captura de datos.

Ilustración 32. Raspberry Pi usadas

5.3.2. Localización de los dispositivos para capturar

Para este supuesto práctico y poder capturar dispositivos abiertos, se han situado las Raspberry Pi en un lugar transitado, donde exista la posibilidad de identificar muchos dispositivos y, por lo tanto, hacer este caso práctico más real e interesante. Se ha elegido la **Universidad de Jaén** para situarlas y obtener la mayor cantidad de información posible.

Concretamente, se han situado estos dispositivos de captura, en dos localizaciones diferentes de la primera planta del **Edificio A-3**. En las siguientes imágenes, se ilustra de forma más concreta, la posición donde se han situado estos dispositivos.

Ilustración 33. Plano Campus Las Lagunillas. [66]

En la imagen anterior, se muestra la localización del edificio en cuestión (cuadro rojo) dentro del campus de **Las Lagunillas** perteneciente a la Universidad de Jaén.

A continuación, se expone una imagen del edificio anterior, con la ubicación aproximada de los dispositivos para tener una perspectiva de dicha ubicación.

Ilustración 34. Edificios Universidad de Jaén. [67]

Como se puede ver en la imagen anterior, en las zonas marcadas con un círculo rojo, es donde se han situado las dos Raspberry Pi (dentro del edificio evidentemente) para la posterior captura de datos. Siendo más concretos aún, los dispositivos fueron colocados en las siguientes situaciones.

- **Raspberry Pi 1**: Despacho del tutor del trabajo Manuel Carlos Díaz Galiano. Dependencia A3-114. **(Círculo rojo de la izquierda)**
- **Raspberry Pi 2**: Despacho técnicos EPS. Primera planta del edificio A3. **(Círculo rojo de la derecha)**

5.3.3. Periodo de obtención de los datos

Con el fin de obtener un conjunto de datos representativo, se han querido tener capturas de datos hechas durante un periodo considerable de tiempo. Concretamente, se han dejado **1 mes** estos dispositivos capturando información ininterrumpidamente. Concretamente, desde el **12 de mayo de 2019 al 12 de junio de 2019**. Como se verá en los siguientes apartados, el capturar datos durante un

periodo largo de tiempo, ha hecho que se obtenga un volumen muy importante de dispositivos Bluetooth.

5.4. Análisis de los datos

5.4.1. Introducción

En este apartado, se llevará a cabo el análisis de todos los datos obtenidos de los diferentes dispositivos Bluetooth. Aquí, se parte de que se han dejado las Raspberry Pi escaneando dispositivos en la Universidad de Jaén durante el periodo de tiempo comentado anteriormente.

5.4.2. Volumen de dispositivos obtenidos sin filtrar

Al dejar las Raspberry capturando información sobre Bluetooth, sabíamos que era muy probable obtener una cantidad muy importante de dispositivos Bluetooth, y a pesar de ello, ha sorprendido que el volumen ha sido muy superior al esperado.

Hay que decir, que durante un mes capturando dispositivos, se han obtenido un total de **24.968 dispositivos Bluetooth diferentes**, lo que a priori, parece una barbaridad. A continuación, se muestra una imagen de la salida al consultar la base de datos.

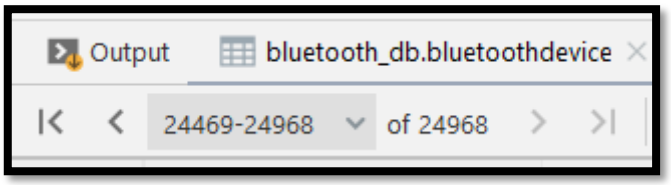


Ilustración 35. Consulta a BD.

A continuación, se muestra una gráfica, que ilustra el número de dispositivos diferentes detectados cada día que se ha estado escaneando.

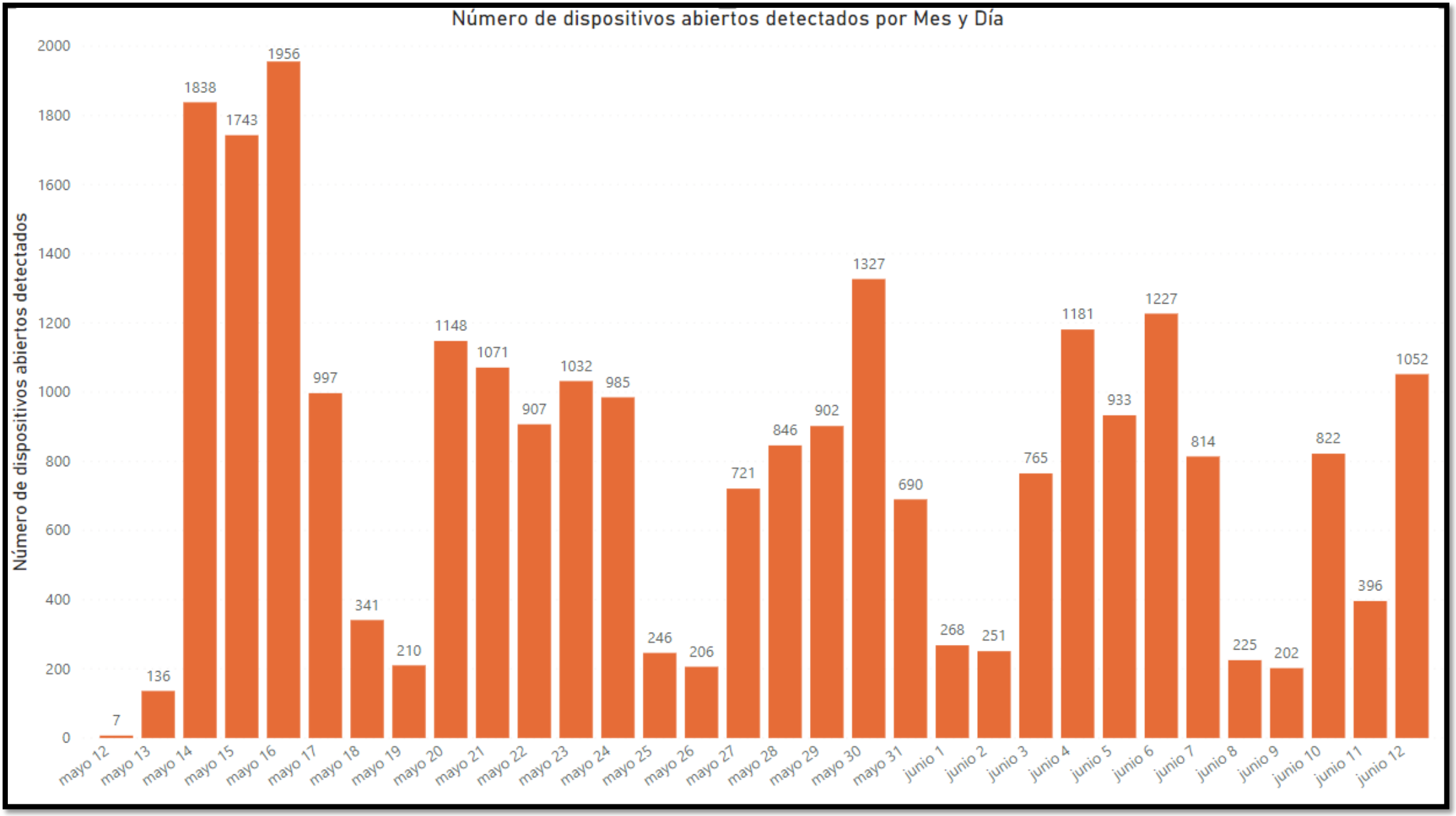


Ilustración 36. Número de dispositivos abiertos detectados por día (datos sin filtrar)

Se observa que se trata, sin duda, de una cantidad extremadamente elevada de dispositivos, sin embargo, existen diferentes motivos por los que ha sucedido esto, por lo que los datos no tienen por qué estar mal o no ser útiles. Las razones por la que posiblemente haya sucedido esto, han podido ser las siguientes.

- Hay que tener en cuenta, que se está capturando información de una cantidad diferentes tipos de dispositivos que incorporen Bluetooth muy elevada y por lo tanto no hay por qué centrarse solo en **smartphones**. Además de estos hay: **pulseras de actividad**, **relojes deportivos**, **televisores, teclados y ratones inalámbricos**, **ordenadores**, **altavoces Bluetooth**, e incluso, se ha logrado capturar **Bluetooth de coches** que pasaban cerca del campus.
- En uno de los apartados de este documento se hablaba de **LE Privacy**. Este método de **seguridad para dispositivos Low Energy**, recordemos que, para impedir que se rastreasen los dispositivos, cambian la MAC que publican cada cierto tiempo. ¿Qué quiere decir esto? Pues simplemente, que, con toda probabilidad, se habrán capturado multitud de direcciones BTADDR o MAC diferentes, pertenecientes a un mismo dispositivo.

Vistas estas situaciones, no parece tan descabellado que se hayan obtenido tantos dispositivos Bluetooth diferentes. Pero para obtener aquellos datos más representativos y que nos sean más útiles, los filtraremos después.

5.4.3. Volumen de dispositivos obtenidos filtrados.

En el apartado anterior, simplemente se ha querido ilustrar la cantidad de información obtenida escaneando, antes de ser manipulada. En este apartado, sin embargo, se eliminarán de este listado de dispositivos aquellos que no vayan a ser útiles y de los que no se puedan obtener información.

Para hacer lo anterior, se aplicarán una serie de acciones a la base de datos, que va a reducir y mucho, la lista de dispositivos Bluetooth y la información sobre ellos. **Es importante, que estos filtros están ideados para estos datos y no se sabe, a priori, si eliminaremos información que no debemos**.

En primer lugar, **se van a eliminar aquellos dispositivos de los cuales ni se conocen su nombre, ni su fabricante**. Estos dispositivos, tienen en la base de datos estos campos con el valor “UNKNOWN”. Estos datos podrían ser útiles en otros casos, sin embargo, para ilustrar los ejemplos de este proyecto, es preferible tener dispositivos que aporten más información.

Nada más que con esto, se ha eliminado un porcentaje elevadísimo de dispositivos. Como se ve en la siguiente imagen, al aplicar este filtro y consultar el número de dispositivos, se hanquedado solo con **1590 dispositivos**.

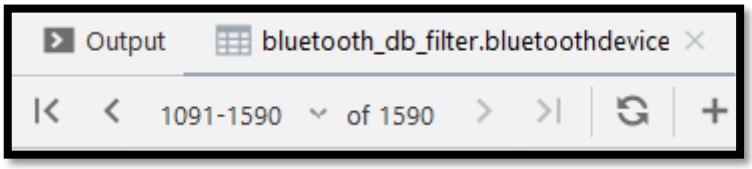


Ilustración 37. Resultado de consulta a BD

A continuación, se exponen un par de gráficos que ilustran elementos interesantes y que después serán comentados.

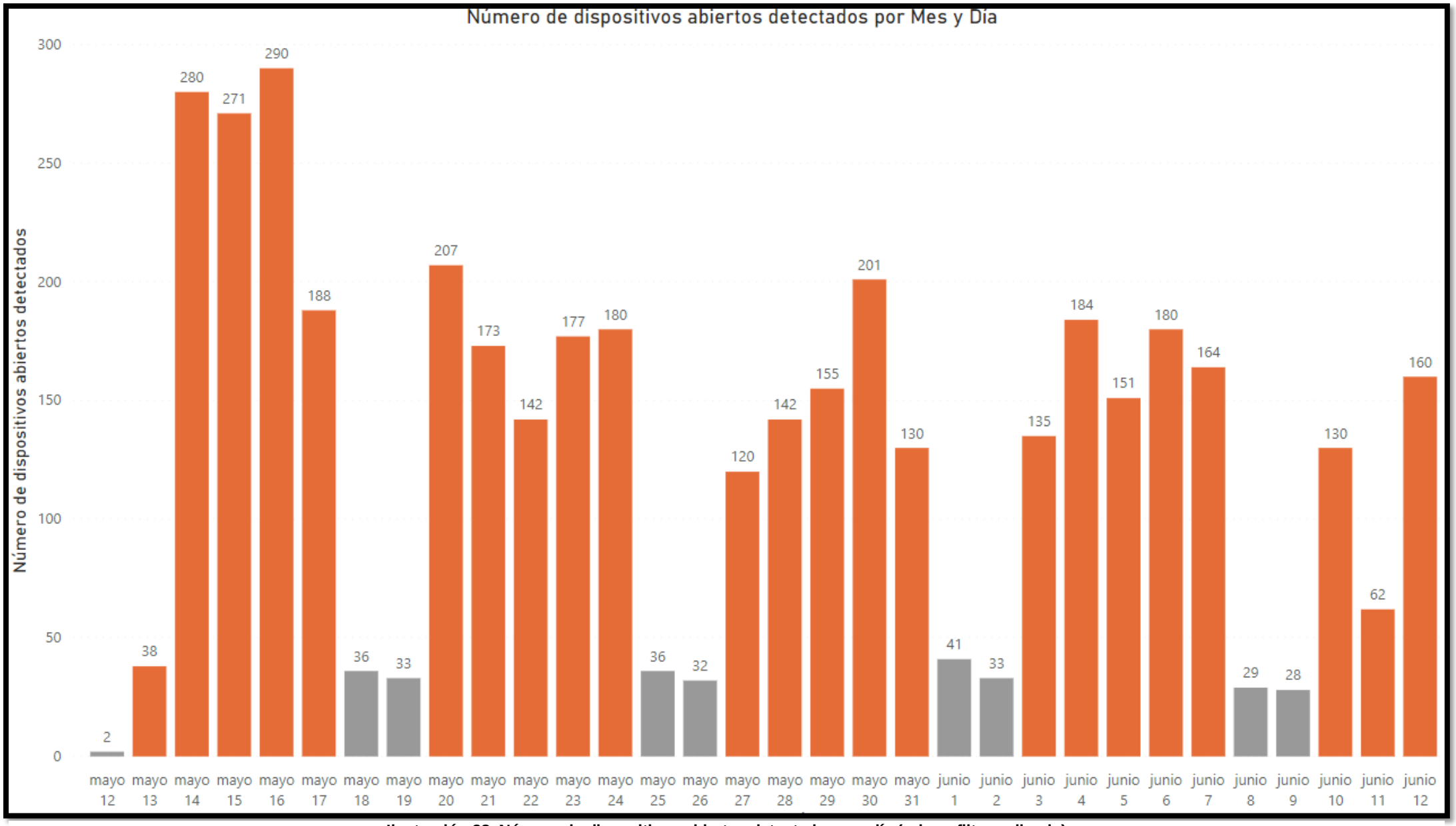


Ilustración 38. Número de dispositivos abiertos detectados por día (primer filtro aplicado)

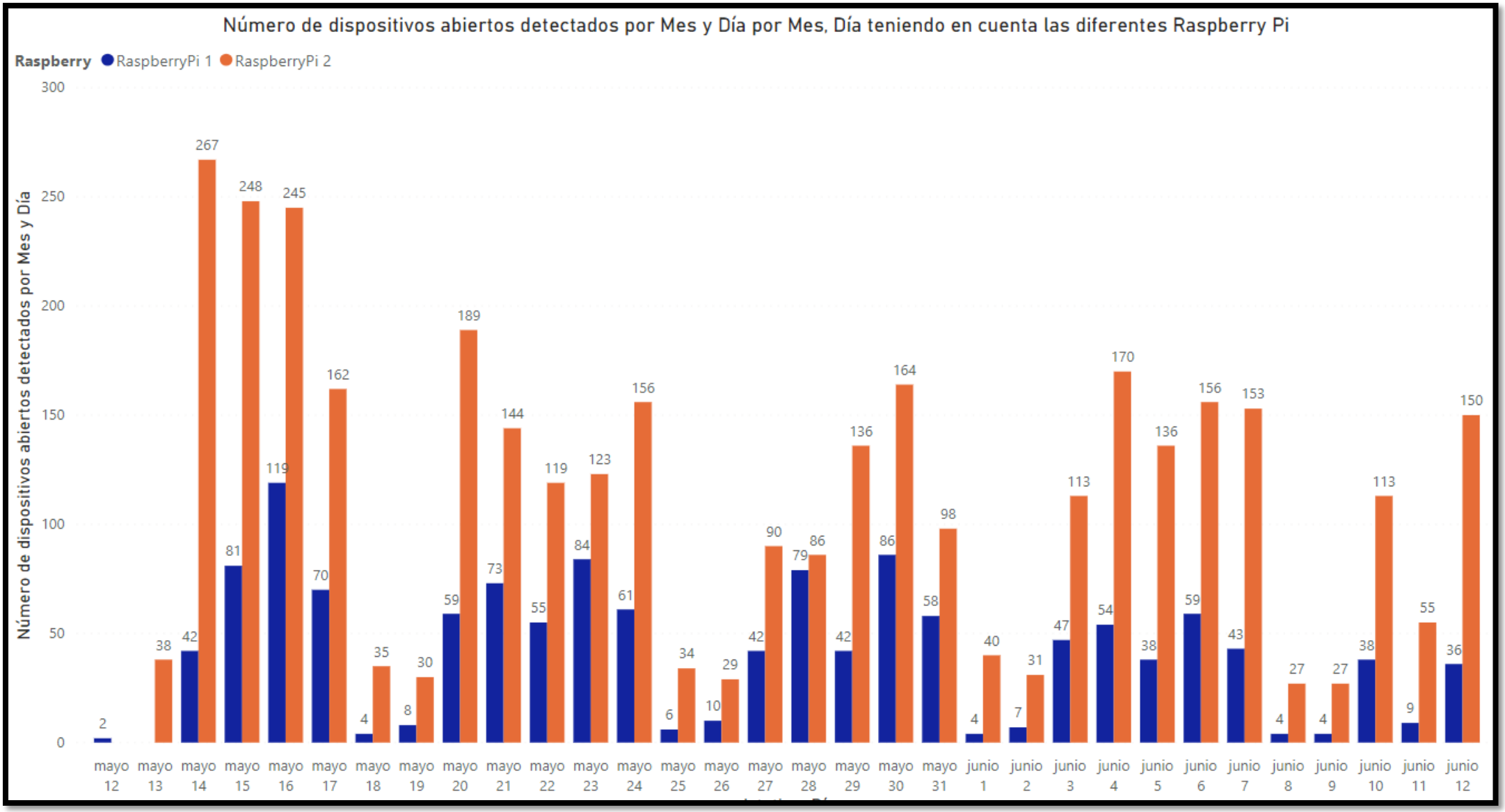


Ilustración 39. Número de dispositivos abiertos detectados por Mes y Día teniendo en cuenta las diferentes Raspberry Pi.

Conclusiones sobre los gráficos (1)

A pesar de solo se observa el número de dispositivos capturados, se observan diferentes elementos, que a un atacante podrían serle de utilidad para planificar una nueva escucha en el futuro.

En primer lugar, se ha visto que al quitar aquellos dispositivos de los que no tenemos información y que, según la información obtenida, no son vulnerables a Blueborne, se reduce mucho el número de dispositivos y, por lo tanto, es posible centrarse en un número más reducido.

En segundo lugar, si no se supiese dónde se ha llevado a cabo el escaneo de los datos, sería posible deducir con bastante facilidad que este se ha hecho en un lugar público con **actividad habitual de lunes a viernes**. Esto se puede ver porque en los **fines de semana**, **marcados en color gris en el gráfico de arriba (Ilustración 38)**, se localizan muchos menos dispositivos que en el resto de días.

Además, en el **segundo gráfico (Ilustración 39)**, un atacante podría visualizar de una pasada, el lugar donde se detectan más dispositivos Bluetooth. Esto le serviría para ver donde puede haber un número mayor de personas y por lo tanto de potenciales víctimas. Parece algo baladí, pero puede mostrar a este atacante dónde dirigir su ataque. En este caso, se observa que la localización donde situamos la **Raspberry Pi 2**, es el lugar donde más dispositivos se localizan prácticamente a diario.

5.4.4. Información sobre dispositivos vulnerables a Blueborne.

En este apartado, se va a centrar en ver qué dispositivos que son **potencialmente vulnerables a Blueborne** y, por lo tanto, aquellos en los que se centraría un atacante, en el caso de hacer lo que aquí se está haciendo.

A continuación, se muestra y explica, información de estos dispositivos, basándose en los datos recabados.

En primer lugar, cabe destacar que, durante el periodo de escaneo, se han detectado un total de **255 dispositivos potencialmente vulnerables a Blueborne**. Como se puede ver, no son pocos los potenciales objetivos.

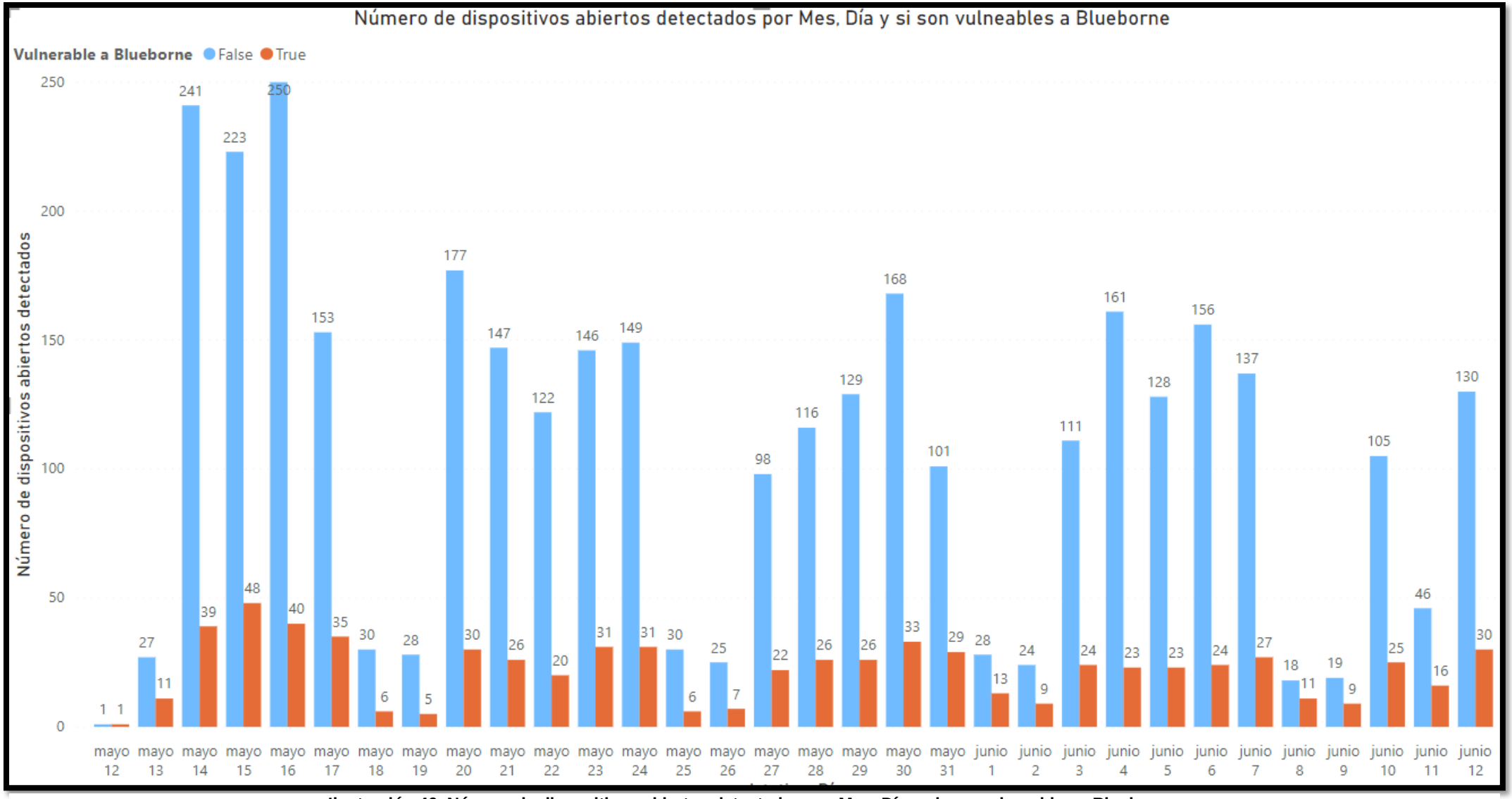


Ilustración 40. Número de dispositivos abiertos detectados por Mes, Día y si son vulnerables a Blueborne.

Conclusiones sobre los gráficos (2)

Como se ha dicho anteriormente, se han detectado un total de **255 dispositivos potencialmente vulnerables a Blueborne**. En la gráfica anterior, se ve la cantidad de dispositivos vulnerables diferentes por día que se han detectado (en color naranja), y también aquellos que no son vulnerables (en color azul).

Como se puede ver, a pesar de solo tener 2 Raspberry Pi, se han detectado una cantidad elevadísima de dispositivos potencialmente atacables explotando Blueborne. Como es normal, se ve claramente que la cantidad de dispositivos **NO vulnerables** detectados, es considerablemente mayor al número de dispositivos **vulnerables**.

Por otro lado, otro parámetro de información acerca de los dispositivos que se ha detectado es el **fabricante** de los mismos. Pero, ¿para qué le puede servir a un atacante saber s existen más dispositivos vulnerables de un fabricante u otro? La respuesta a esto es simple. Hay atacantes, cuyo objetivo es atacar e infectar el mayor número de dispositivos posibles. Esta información les puede servir para preparar un exploit o un ataque dirigido a los dispositivos de ese fabricante y así asegurarse que infectan el mayor número de dispositivos posible.

A continuación, se va a ilustrar y explicar información sobre esto.

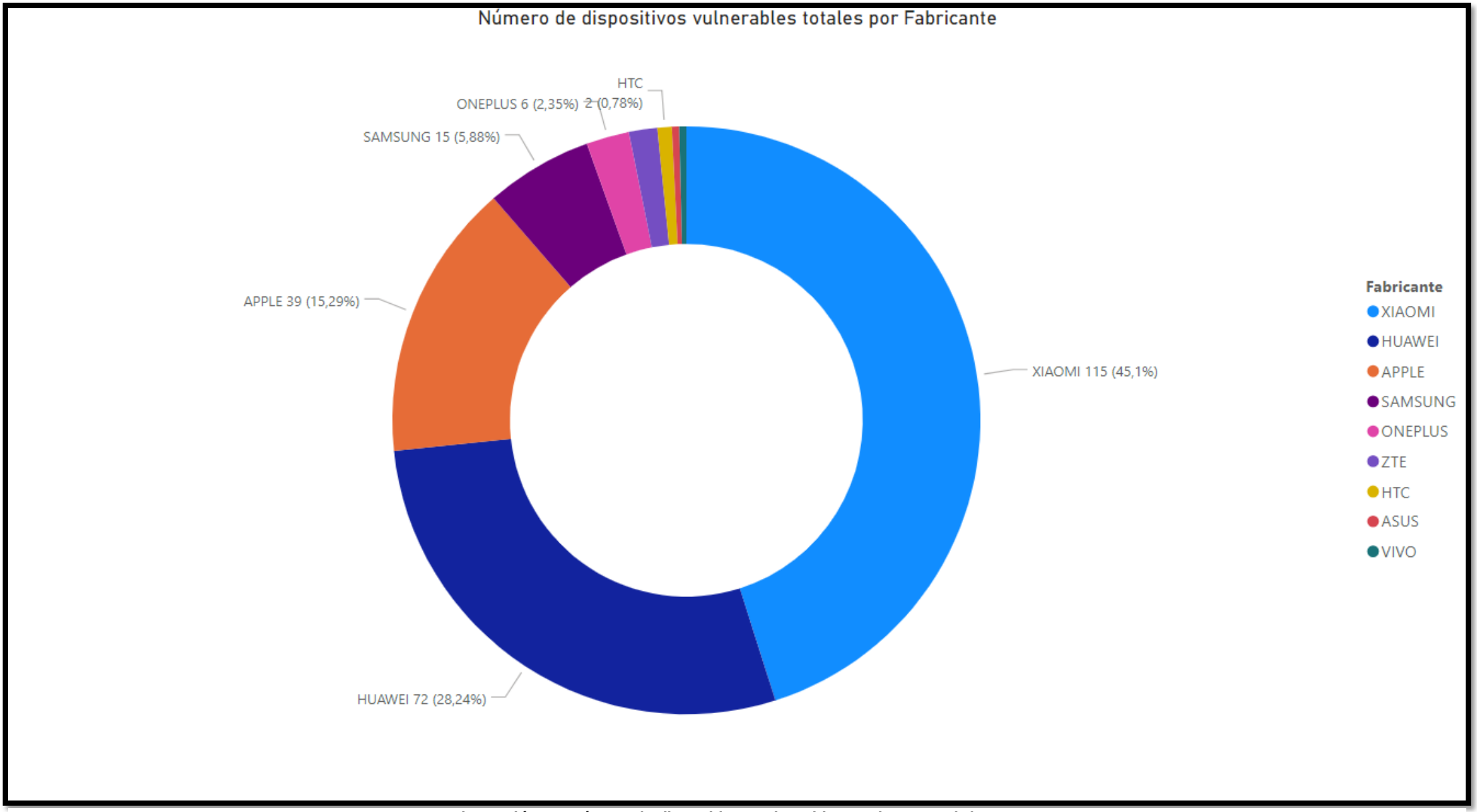


Ilustración 41. Número de dispositivos vulnerables totales por Fabricante

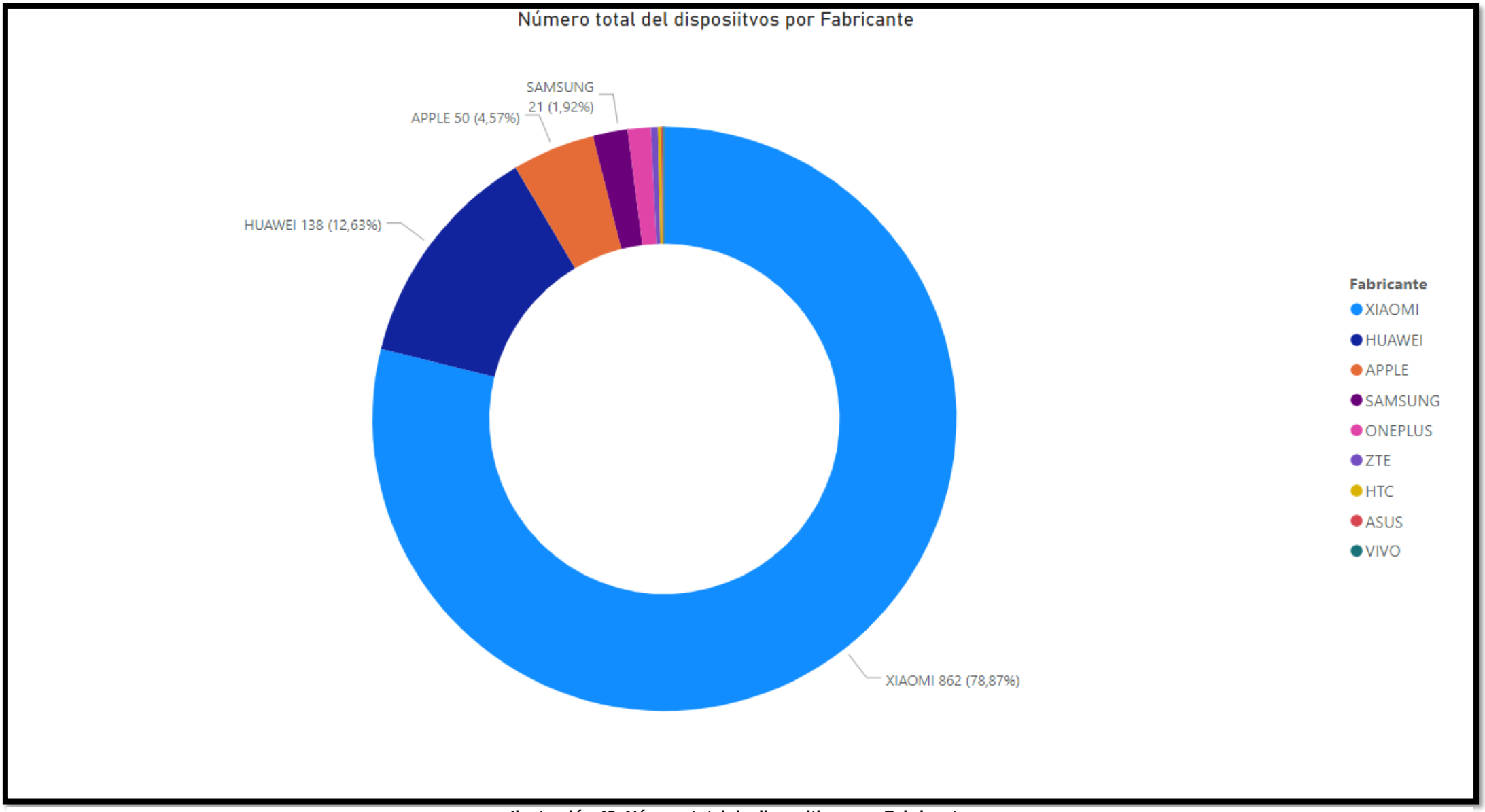


Ilustración 42. Número total de dispositivos por Fabricante.

Conclusiones sobre los gráficos (3)

En el **gráfico de la Ilustración 41**, puede verse que existe una diferencia de abismal en volumen de dispositivos vulnerables de alguna de las marcas que se han escaneado. Por ejemplo, encabezando la lista está al conocido fabricante chino **XIAOMI** con **115 dispositivos potencialmente vulnerables**, seguido por otras dos potencias tecnológicas como son **HUAWEI y APPLE** con **72 y 39 dispositivos potencialmente vulnerables respectivamente**. Pero, ¿quiere decir esto que estas marcas son más vulnerables que otras? La respuesta es sencillamente que NO.

Todas estas empresas son gigantes tecnológicos muy importantes, y la razón por la que se han detectado más dispositivos potencialmente vulnerables se puede ver claramente en el **gráfico de la Ilustración 42**. En este gráfico, se visualiza como no se detectan más dispositivos vulnerables de estas compañías porque son más inseguras que otras, sino porque el volumen de dispositivos que venden es muy superior al de la competencia.

El ejemplo más claro está en el fabricante que encadena la lista: **XIAOMI**. De este fabricante, se ha detectado un volumen muy alto de dispositivos potencialmente vulnerables, concretamente 115 dispositivos, sin embargo, el **número TOTAL** de dispositivos que se ha detectado, **es de 862 dispositivos**, es decir, **un 13,34% de los dispositivos de XIAOMI detectados, son potencialmente vulnerables**. Sin embargo, de otras marcas como ZTE O HTC apenas se detectaron dispositivos, pero todos ellos son potencialmente vulnerables.

Este valor (862 dispositivos), puede parecer un número exageradamente elevado de dispositivos de XIAOMI teniendo en cuenta la diferencia con el resto de fabricantes, pero se ha comprobado, que esto se debe a que existe una gran cantidad de unos de sus dispositivos superventas: La **Xiaomi Mi band** en sus diferentes versiones.

En el **siguiente gráfico (Ilustración 43)**, se muestra de una forma más clara, la relación entre el volumen de dispositivos potencialmente vulnerables de los fabricantes, respecto a los dispositivos totales que se han detectado de los mismos. También se expone una gráfica, con un listado de fabricantes más extenso .

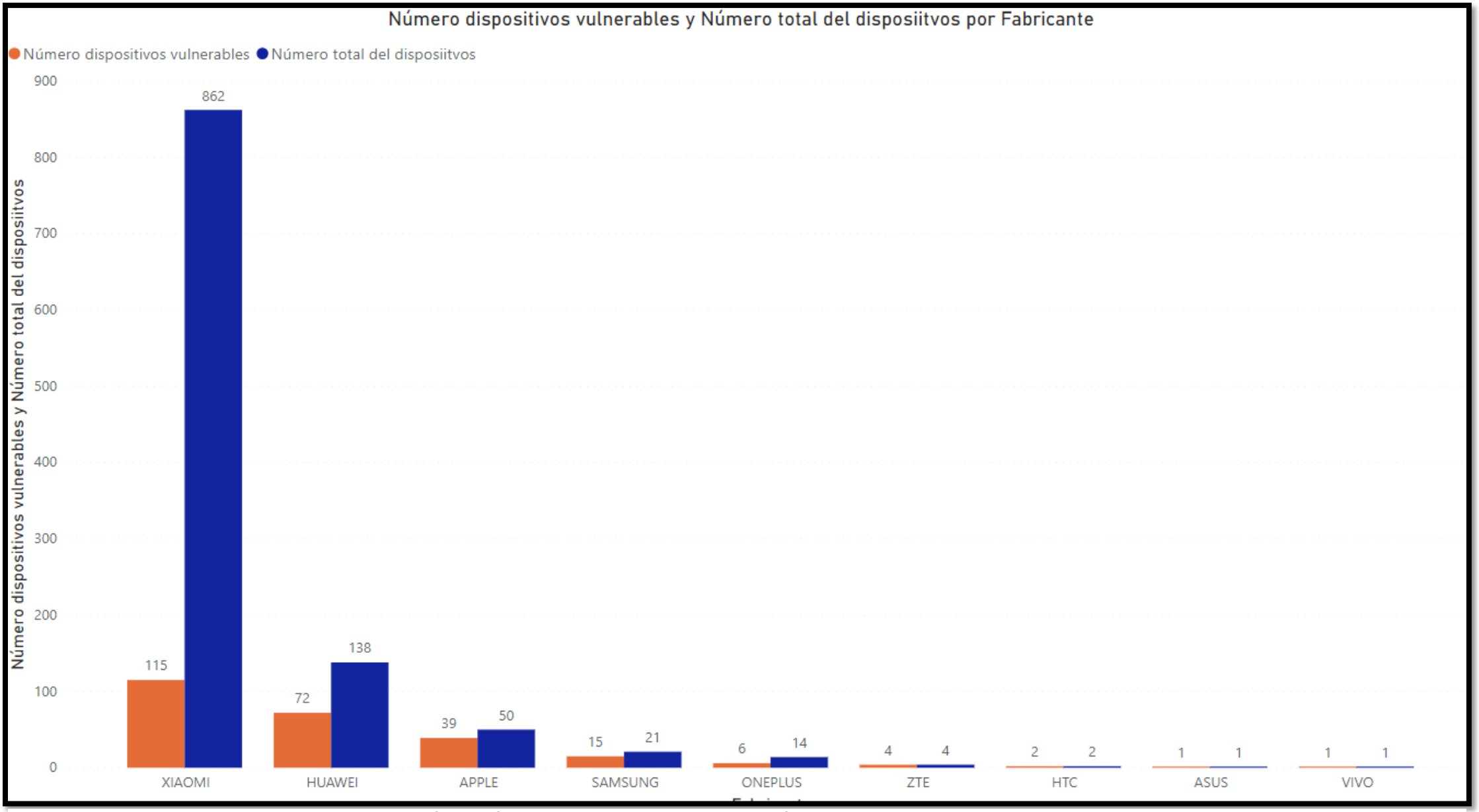


Ilustración 43. Número de dispositivos vulnerables y número total de dispositivos por Fabricante.

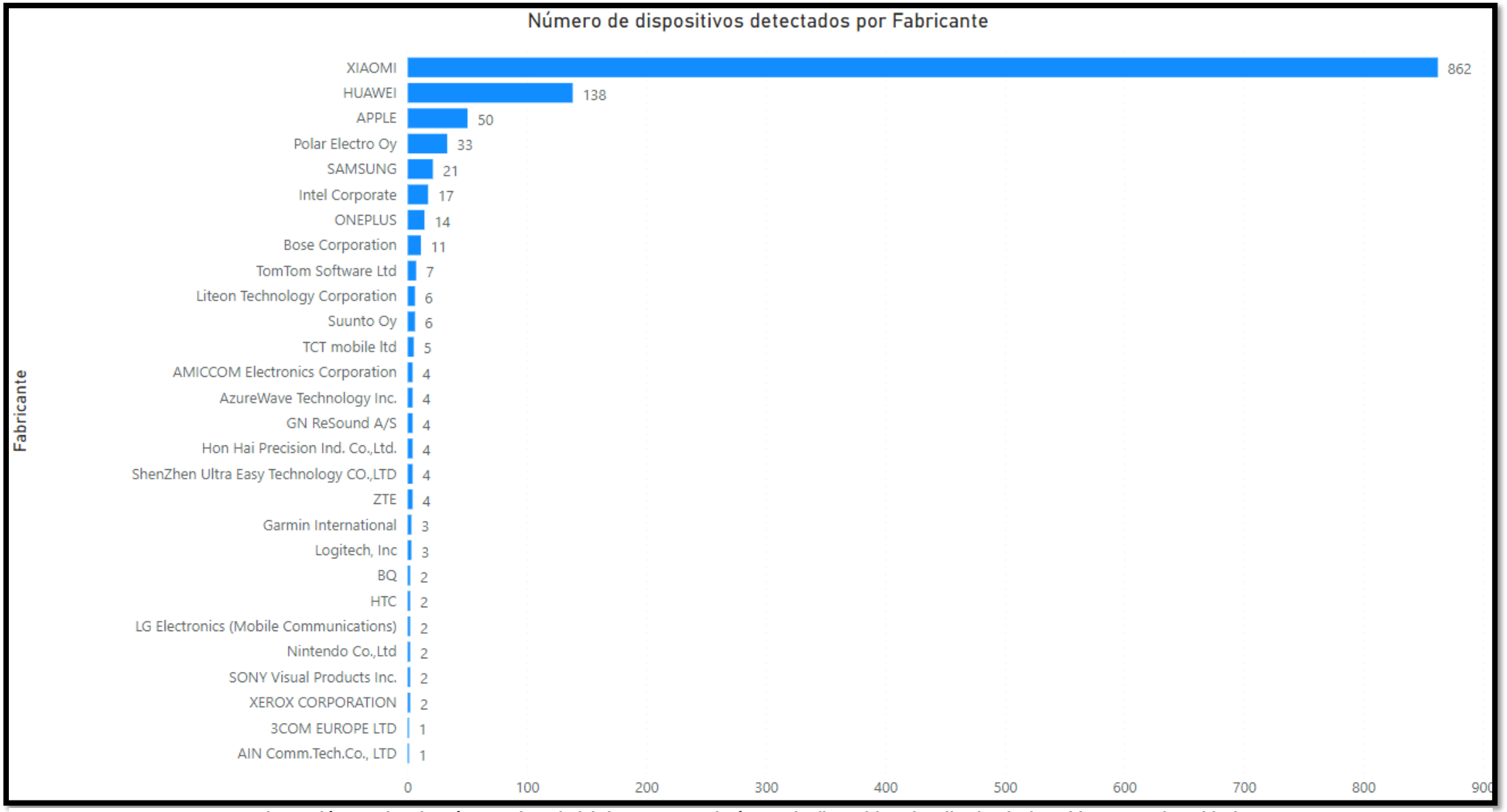


Ilustración 44. Listado más completo de fabricantes, con el número de dispositivos localizados (vulnerables y no vulnerables)

5.4.5. Ataques usando Blueborne

En este apartado, se intentará mostrar que atacar un dispositivo haciendo uso de este vector de ataque es posible. Cabe destacar, que no se ha llevado este ataque de manera práctica principalmente debido a dos motivos fundamentales:

- En primer lugar, la no disposición de un dispositivo personal vulnerable en el que intentar llevar a cabo una demostración del mismo.
- Y, en segundo lugar, aunque se podría haber intentado programar un **exploit** para atacar un dispositivo cuando el software lo localizase, por razones legales y de privacidad de datos, no se podía realizar sin el consentimiento del usuario.

Sin embargo, se sabe que es perfectamente posible debido a que este vector de ataque, como ya se dijo anteriormente, fue descubierto por la empresa de ciberseguridad Armis [19] y esta que ha hecho públicas unas demos de cómo se pueden atacar diferentes sistemas operativos.

- Ataque al Sistema Operativo Android. [54]
- Ataque MITM en Windows. [55]

Evidentemente, un posible atacante no puede replicarlo simplemente con una demostración como las anteriores, sin embargo y con fines didácticos y para la comunidad, **Armis publicó los exploits** que utilizó para realizar estas demostraciones. [56].

El atacante, podría modificar estos exploits para llevar a cabo su ataque. Hay que tener en cuenta que actualmente están diseñados para un tipo de dispositivo concreto.

En el transcurso de nuestra investigación, hemos encontrado, como otras personas explican paso a paso como **modificar estos exploits**, en función de que dispositivo queremos atacar. [57]

5.4.6. Seguimiento de dispositivos

Aquí se verá otro de los problemas a los que se exponen una persona al tener un dispositivo con Bluetooth encendido o visible. Estamos hablando del **seguimiento y rastreo** de alguien a través de su dispositivo con Bluetooth.

En apartados anteriores, se vio como muchos de los dispositivos BLE tenían un mecanismo de seguridad llamado **LE Privacy**, pero esto no es algo que tengan todos los dispositivos Bluetooth actuales. Y aunque así fuese, también se ha visto en este trabajo, que con el hardware adecuado, como **Ubertooth One**, es posible escanear tanto estos dispositivos, como aquellos que se encuentren encendidos y en modo “no visible”.

En este caso, como solo se ha dispuesto de 2 Raspberry Pi, se mostrarán unos pequeños ejemplos de cómo es posible tener situados en un lugar y en un tiempo determinado a una persona por llevar encima un dispositivo con el Bluetooth encendido. Lo que se va a hacer, es mostrar algunos de los dispositivos escaneados, y dependiendo de su localización, han sido detectados por la **Raspberry Pi 1** o por la **Raspberry Pi 2** en diferentes fechas y horas.

NOTA: Hay datos de dispositivos que no se muestran para proteger la privacidad de los usuarios.

Ejemplo 1.

El primer dispositivo que se va a analizar, es el siguiente:

	btAddr	devName	devManufacturer	vulnerableBlueborne
1	04:B1:67:[redacted]	Lau	XIAOMI	1

Ilustración 45. Información de dispositivo de ‘Lau’.

Como se puede observar, el dispositivo anterior nombre de dispositivo es ‘**Lau**’ del que se puede deducir que es un diminutivo de **Laura**, lo que nos permitiría, no solo localizar a este usuario, sino identificarlo con más facilidad.

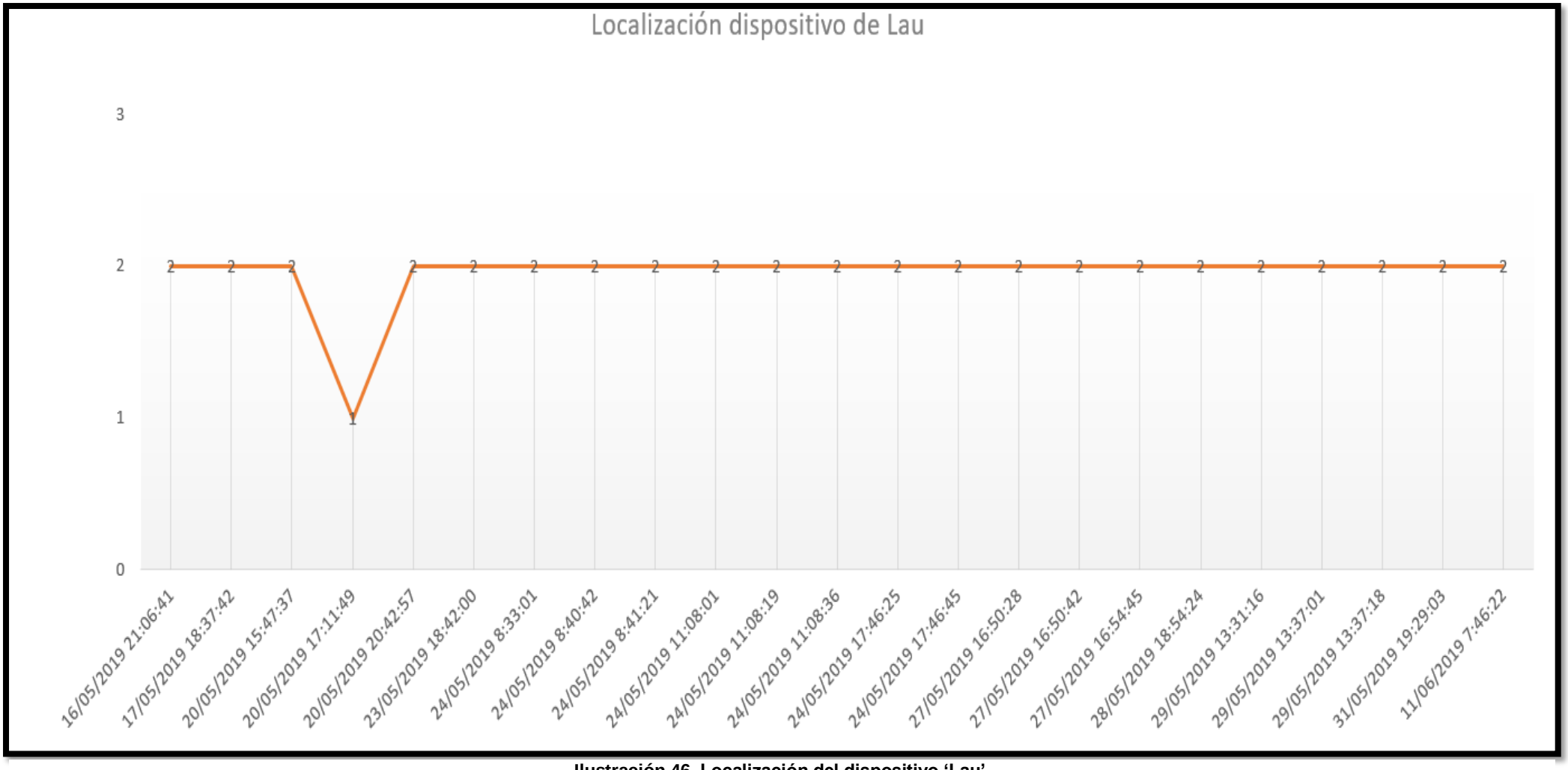


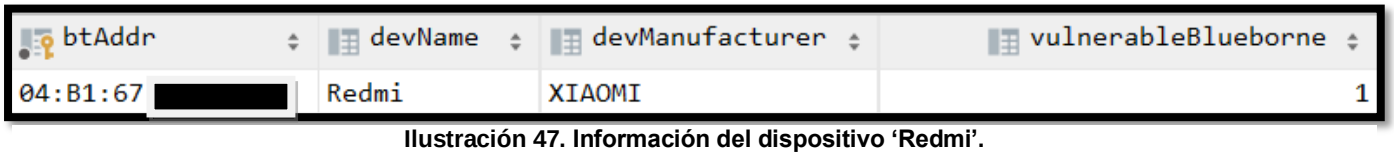
Ilustración 46. Localización del dispositivo 'Lau'.

En el anterior gráfico, se observa como este usuario se ha movido casi todo el tiempo que ha sido localizado, cerca de la **Raspberry Pi 2**, es decir, cerca del **despacho de los técnicos del edificio A3**, por lo que es muy posible que se trate de alguien del alumnado puesto que, en la zona de despachos del profesorado, solo ha sido localizada cerca de la Raspberry Pi 1 en una sola ocasión. Además, investigando un poco, se ve que el nombre de Laura, no figura en el listado de profesorado de la EPS de Jaén.

Además, este dispositivo podría ser atacado puesto que es potencialmente vulnerable.

Ejemplo 2.

En segundo lugar, se va a ilustrar información de presencia del dispositivo que se especifica a continuación.



btAddr	devName	devManufacturer	vulnerableBlueborne
04:B1:67 [redacted]	Redmi	XIAOMI	1

Ilustración 47. Información del dispositivo 'Redmi'.

En la imagen anterior se puede ver que, en este caso, tenemos un dispositivo **Redmi** del fabricante **XIAOMI**, del que no podemos obtener información acerca del dueño del dispositivo, debido a que el nombre que se le ha dejado es el que trae por defecto.

A continuación, se ilustra la información acerca de dónde y cuándo ha sido localizado.

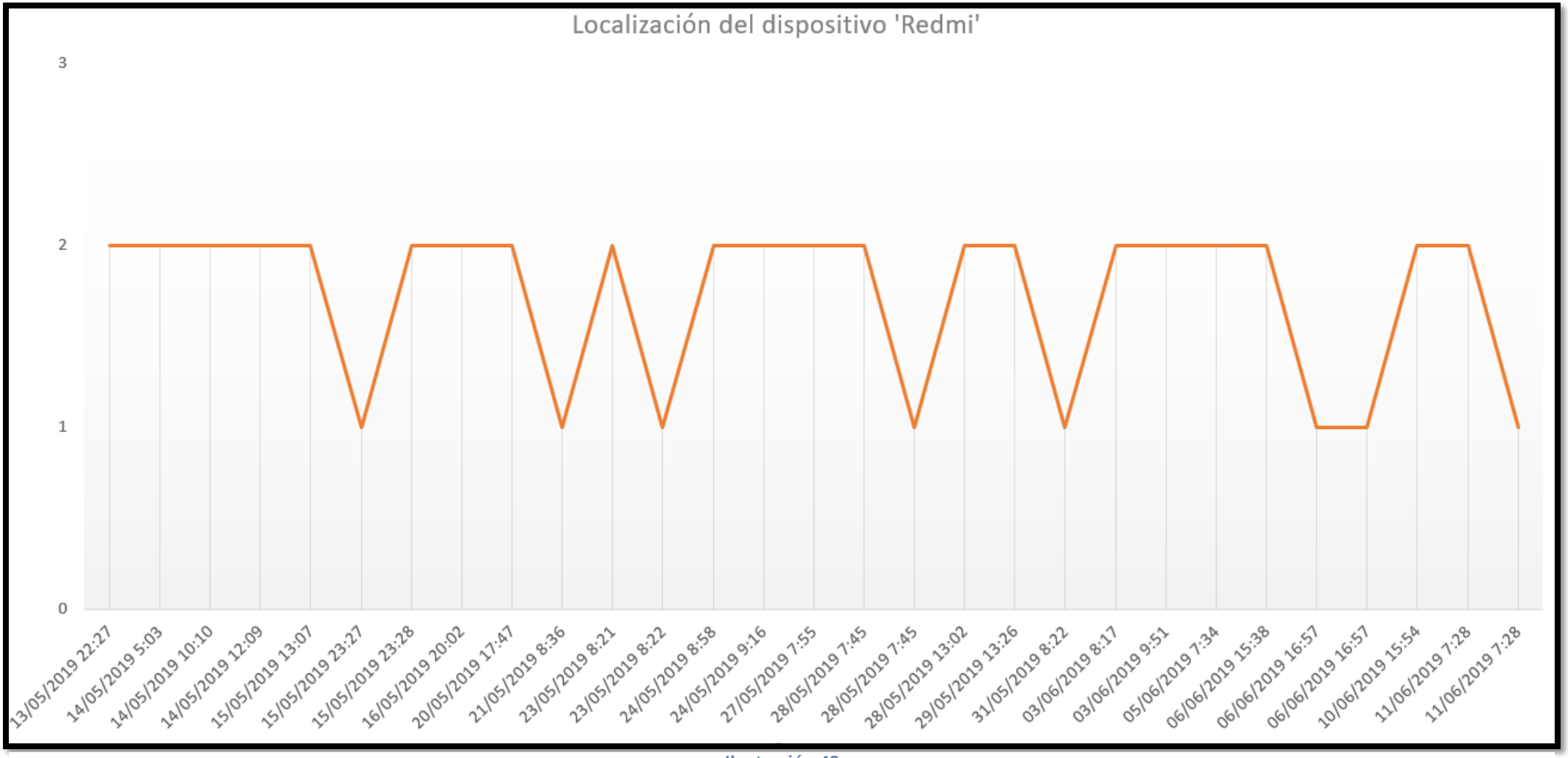


Ilustración 48

Se puede observar, que a diferencia del **Ejemplo 1**, la posición de este dispositivo ha variado de una forma mucho más continua. En este caso, es mucho más difícil conocer quién es este dispositivo, ya que tanto el nombre, como su localización, da lugar a que pueda ser cualquiera.

Observando un poco las horas a las que este dispositivo es localizado, es posible **teorizar** el perfil de la persona a la que pertenece el dispositivo. Esto se obtiene de las siguientes premisas:

- El dispositivo, siempre es localizado en ciertos tramos horarios puntuales (salvo alguna excepción). Estos tramos son: por la mañana muy temprano (5 de la mañana, 7 de la mañana), y por la noche muy tarde (22, 23 de la noche). Estos horarios, hacen que se descarten la mayoría del alumnado y profesorado.
- El dispositivo es localizado en ambos sitios multitud de veces. Además, existen tramos, en los que el dispositivo es localizado en un lugar, y a los pocos minutos es localizado por otro. Esto muestra, que se trata de una persona que se desplaza y cambia de posición con frecuencia.

De las anteriores premisas, se obtiene la conclusión de que, con una probabilidad muy elevada, **este dispositivo pertenece a algún miembro de seguridad perimetral del campus**. Esto acotaría mucho la búsqueda de su dueño, en el caso de querer localizarlo

6. GUÍA DE RECOMENDACIÓN Y USO DE BLUETOOTH

En este documento, se ha intentado exponer y argumentar, lo expuestos que están todos los usuarios que tengan el dispositivo Bluetooth abierto o activado de forma continua. Ha quedado evidenciado, que un atacante podría comprometer a esos usuarios de una forma muy grave en aspectos de privacidad, económicos, etc…Esto se agrava mucho más, si estos dispositivos vulnerables están adheridos a una red corporativa o algo similar. Es por eso que, se ha querido incluir en este documento, unas directrices o consejos de uso del Bluetooth en un entorno empresarial u organización con ayuda de las fuentes citadas. Sin embargo, todo lo que se incluya aquí, sería extrapolable al uso personal del dispositivo, no siendo

necesario que esté conectado a ninguna red corporativa. Estas directrices se exponen a continuación.

1. **NO DEJAR EL BLUETOOTH ENCENDIDO CUANDO NO SE ESTÉ USANDO.** [58]

   Esta, aunque es la primera medida y probablemente la más obvia, es también la más importante. Muchos dispositivos que tienen la tecnología Bluetooth son configurables en cuanto al tiempo que el Bluetooth está encendido. En el caso de que no sea así, habrá que hacer esto de forma manual. Teniendo en cuenta el grado de amenaza, es recomendable cumplir con esta recomendación.

2. **CADA VEZ QUE UN DISPOSITIVO INTENTE CONECTARSE, SE DEBE SOLICITAR AUTORIZACIÓN.** [58]

   Cuando dos dispositivos van a establecer una conexión, primero es necesario llevar a cabo una asociación o emparejamiento como se ha comentado en apartados anteriores a este documento. En diferentes versiones Bluetooth, se puede hacer esta asociación de forma directa o requiriendo una clave. Aunque en la mayoría de dispositivos actuales suele estar ya esta última opción por defecto, se recomienda comprobarlo para mayor seguridad.

3. **PONER BLUETOOTH EN MODO "NO VISIBLE" CUANDO NO SE ESTÉ LLEVANDO A CABO UN EMPAREJAMIENTO ENTRE DISPOSITIVOS**. [59]

   Solo se debe tener el Bluetooth visible cuando se va a llevar a cabo el emparejamiento de dispositivos. Esto, no nos hace estar muy seguros, pero se lo pone mucho más difícil a un potencial atacante.

4. **NO PERMIIR CONEXIONES A UN DISPOSITIVO BLUETOOTH DE ORIGEN DESCONOCIDO.** [58, 59]

Aquí entra tanto envío de información, como emparejamientos. Esto debemos tenerlo en cuenta porque existe la posibilidad de que te conectes a un dispositivo desconocido y este se comporte de forma maliciosa con tu dispositivo. Muchos de estos dispositivos esperan que otros se conecten a él y propagan virus de forma automatizada. Es por eso que es recomendable, solo conectarse a dispositivos de los cuales se conozca su origen. Si se acepta una conexión desconocida, se está dando permiso a que ese dispositivo se conecte con otro.

5. **NO REALIZAR EMPAREJAMIENTOS DE DISPOSITIVOS EN LUGARES PÚBLICOS.** [58]

   Cómo se ha dicho en apartados anteriores, el emparejamiento entre dispositivos es probablemente el momento donde los estos son más vulnerables. Es en el emparejamiento cuando dos dispositivos Bluetooth que quieren conectarse, intercambian las claves que van a permitir esa conexión. Es en este momento, donde un usuario malintencionado podría interceptar dichas claves, y conectarse a otro dispositivo haciéndose pasar por un dispositivo de confianza. Esto le daría la posibilidad de infectarlo con malware, robar información personal, etc.

6. **DESHABILITAR PERMISOS QUE NO SE USAN.** [59]

   Esto, solo se podrá hacer en aquellos sistemas que permitan a los usuarios especificar qué servicios están habilitados o deshabilitados. Por ejemplo, en algunos dispositivos móviles quizá sea posible habilitar el uso de audio y deshabilitar la transferencia de archivos.

7. **USAR CLAVES ALFANUMÉRICAS DE DOCE DÍGITOS O MÁS U OCHO CARACTERES ALFANÚMERICOS CON CARACTERES ESPECIALES.** [59]

   El hacer esto durante el proceso de emparejamiento, hace que sea muchísimo más difícil que un atacante obtenga las claves por fuerza bruta. Para lograr obtenerlas, tendría que utilizar técnicas muchos más avanzadas.

8. **ESTABLECER LOS DISPOSITIVOS BLUETOOTH EN EL MODO DE SEGURIDAD 2, 3 O 4, YA QUE REQUIEREN AUTENTICACIÓN Y CIFRADO PARA LA COMUNICACIÓN.** [59]

   Esto solo será posible hacerlo en aquellos dispositivos que nos lo permitan (los teléfonos móviles no suelen permitirlo). Esto va a ayudar a evitar la conexión de dispositivos no autorizados.

9. **ELIMINAR DISPOSITIVOS ASOCIADOS QUE YA NO UTILICEMOS O NO RECONOZCAMOS.** [59]

   Esto eliminará la clave de enlace compartida cuando los dispositivos fueron emparejados. Dado que el emparejamiento se realiza solo una vez, puede ayudar a que los dispositivos en los que se había confiado previamente, no puedan tener acceso a nuestro dispositivo sin notificar al usuario.

Además de las recomendaciones anteriores, el **NIST** (National Institute of Standards and Technology) [60] proporciona una guía muy completa para la seguridad en Bluetooth, donde podemos encontrar un Checklist de diversas acciones, algunas mencionadas anteriormente y otras adicionales, para incrementar dicha seguridad en el uso esta tecnología. [16]

7. CONCLUSIÓN FINAL

No cabe duda, de que la tecnología Bluetooth ha sido y es un estándar de tecnologías inalámbricas más importantes, versátiles y con un crecimiento increíble en los últimos 20 años, además de una solución fácil a la hora de interconectar cada vez más dispositivos. Sin embargo, se ha visto que las diferentes versiones Bluetooth utilizadas a día de hoy, tienen muchos problemas de seguridad. Al ser una tecnología que se encuentra cada vez más en el día a día de los usuarios, es importantísimo concienciar a los usuarios sobre esta tecnología y que entiendan los riesgos que conlleva su uso. Es por eso, que en este trabajo se pretendía evidenciar cómo, de una forma sencilla, se puede sacar provecho de esta tecnología para llevar a cabo acciones maliciosas y, por lo tanto, conseguir que los usuarios que hacen

uso de esta tecnología, tenga adquiera y afronte la responsabilidad de hacer un buen uso de la misma. Siendo responsables y sabiendo a los riesgos que se enfrentan los usuarios, todos pueden hacer uso de esta tecnología útil y maravillosa que tenemos tienen a su alcance.

ANEXO 1. Instalación de librerías e integración del software en la Raspberry Pi.

En este anexo, se va a ver cómo integrar el software de este proyecto junto con la Raspberry Pi. Esta integración le va a dar mucho más valor al mismo debido a que va a ser mucho más sencillo de utilizar.

Opción 1. Instalación del software en imagen de Raspbian limpia.

El punto de partida va a ser suponer que hay instalada ya una imagen limpia del sistema operativo **Raspbian** [61] (sistema operativo oficial de Raspberry Pi) en la Raspberry Pi.

Cuando instalas Raspbian, ya tendrá instalado **Python**, con lo que esto no tendríamos que hacerlo. Sin embargo, es necesario que instalar todas las librerías necesarias para que el software funcione e instalar **MySQL** (en Raspbian suele estar instalado) y crear la base de datos necesaria para guardar los datos de nuestro proyecto.

Instalación de dependencias para Python.

1. En primer lugar, como es evidente, se va a clonar, descargar o copiar el software (según la preferencia del usuario) en una ubicación de la Raspberry Pi.

   ```
   git clone https://bitbucket.org/cjsm0004/tfm_bluetooth.git
   ```

   En este caso, se va a suponer que se ha clonado el proyecto en la ruta **/root/Escritorio**. Esta ruta, será necesaria para configuraciones posteriores.

2. Instalación de la librería para desarrollar con Bluetooth.

   ```
   sudo apt-get install bluetooth libbluetooth-dev
   ```

3. Instalación de la librería PyBluez.

   ```
   sudo python –m pip install pybluez
   ```

4. Instalación de pygattlib

```
sudo python –m pip install pygattlib
```

5. Si se produce un error cuando se haga el paso anterior, se tendrán que instalar otras dependencias que **pygattlib** necesita.

```
sudo apt-get install pkg-config
sudo apt-get install libboost-python-dev
sudo apt-get install libboost-thread-dev
sudo apt-get install libglib2.0-dev
sudo apt-get install python-dev
```

6. Una vez instaladas estas dependencias, ya se podría volver a instalar **pygattlib (Paso 3)**.
7. Instalación **bluepy**

```
sudo python –m pip install bluepy
```

8. Otra librería necesaria, es la librería de Python, que permite conectar de una forma sencilla, con la base de datos MySQL. Esta librería es **mysql-connector**.

```
sudo python –m pip install mysql-connector-python
```

Instalación de MySQL y creación de la base de datos.

1. Instalción MySQL

```
sudo apt install mysql-server php-mysql
```

2. Una vez instalado, accedemos como **root**.

```
sudo mysql –user=root
```

3. Una vez se accede como root, se elimina el usuario **root**.

```
DROP USER ‘root’@’localhost’;
```

4. Después, se vuelve a crear un nuevo usuario **root** pero con la contraseña que nosotros queramos.

```
CREATE USER ´root´@´localhost´ IDENTIFIED ´my_new_password´;
```

Siendo el valor **my_new_password** la nueva contraseña elegida por el usuario.

5. Para que realmente sea como el usuario **root**, es necesario darle todos los permisos en el gestor de bases de datos de MySQL. Para eso, ejecutamos los siguientes comandos.

```
GRANT ALL PRIVILEGES ON *.* TO ´root´@´localhost´ WITH GRANT OPTION;

FLUSH PRIVILEGES;
```

Ahora, ya habría un usuario root creado por el usuario, con un control total sobre MySQL. Cabe destacar, que solo será acceder con este usuario desde la misma Raspberry Pi. No es posible acceder a ella ni siquiera desde otro equipo de la red local.

6. A continuación, se creará la base de datos de nuestro proyecto, a la que vamos a llamar **Bluetooth_DB**.

```
CREATE DATABASE Bluetooth_DB;
```

7. A continuación, se van a crear las tablas de la base de datos de una forma rápida y sencilla, haciendo uso del script que hay en la carpeta del proyecto: **tfm_Bluetooth > Bluetooth_Carlos > database_scriptsSQL > script.sql**. Para ejecutarlo desde el terminar, hay que salor de mysql y ejecutar el siguiente comando

```
mysql –u root –p Bluetooth_DB < “ruta_script_sql”
```

Una vez hecho esto, se puede entrar como **root** para seguir con la configuración.

8. Ahora, para no hacer uso del usuario **root** a la hora de acceder a la base de datos **Bluetooth_DB**, se creará un usuario que tenga únicamente permisos a esta base de datos. A este usuario, se le va a llamar **Bluetooth_DB_admin** y su contraseña va a ser **password**. Este usuario se va a crear para que sea accesible desde cualquier IP. Esto ayudará al usuario, a acceder a la base de datos desde cualquier gestor de bases de datos en la propia red local.

```
CREATE USER ´Bluetooth_DB_admin´@´%´ IDENTIFIED BY ´password´;
```

9. Se otorgan permisos al usuario **Bluetooth_DB_admin** para que tenga acceso y control sobre la base de datos **Bluetooth_BD**.

```
GRANT ALL PRIVILEGES ON Bluetooth_DB.* TO ´Bluetooth_DB_admin´@´%´
            IDENTIFIED BY ´password´ WITH GRANT OPTION;

                        FLUSH PRIVILEGES;
```

10. Por último, para que se pueda acceder desde la red local a la base de datos, es necesario comentar la línea **skip-external-locking** en el apartado **mysqld** en el fichero de configuración de mysql.

Configuración y ejecución del software

Una vez hemos instalado todas las dependencias y creada la base de datos, se verán cuáles son los parámetros de configuración del propio software, que hay que que cambiar para poder ejecutarlo.

Dentro de la carpeta del proyecto, tenemos un directorio llamado **config**, en el que van a estar todos los ficheros que se usen para la configuración. En este caso, solo hay un fichero de configuración en esta carpeta denominado **config_params.py**. Los parámetros a cambiar son los siguientes:

- **my_location**: Este parámetro, es un string utilizado para indicarla posición del dispositivo de escaneo posición cuando realizamos el mismo. Lo que ponga en este parámetro se va a almacenar en la base de datos.
- **scanner_device_id**: Este parámetro es el ID del dispositivo Bluetooth que se va a utilizrar. Normalmente, este valor va a ser el que hay por defecto debido a que lo normal es tener solo un adaptador.
- **absolute_project_path**: Aquí, hay que poner la ruta absoluta donde se localizan las diferentes carpetas de nuestro proyecto. En el caso de este ejemplo es **/root/Escritorio/tfm_bluetooth/Bluetooth_Carlos/**

Los siguientes parámetros, hacen referencia a la configuración de la base de datos. Esta configuración está establecida para la base de datos local dentro de la Raspberry Pi, sin embargo, el usuario podría poner una base de datos MySQL cualquiera, **incluso una base de datos remota**.

- **user**: Nombre de usuario para acceder a la base de datos.
- **password**: Contraseña de acceso de ese usuario para la base de datos.
- **host**: dirección IP donde se localiza la base de datos
- **database**: Nombre de la base de datos

Ya está todo listo para la ejecución de nuestro software. Para eso, hay que ejecutar el fichero **bluetooth_listener.py** en un terminal.

```
sudo python /root/Escritorio/tfm_bluetooth/Bluetooth_Carlos/bluetooth_listener.py
```

Ejecución del software al iniciar Raspberry Pi

Como se ha dicho anteriormente, uno de los objetivos y una de las razones por las que se ha seleccionado una Raspberry Pi, era tener un dispositivo **Plug & Play** que funcionase y capturase información de dispositivos Bluetooth nada más conectarlo.

Anteriormente, se vio cómo ejecutar el software de forma manual, sin embargo, ahora, se verá cómo hacer que el software se inicie al conectar la Raspberry Pi a la corriente.

Para hacer esto, se hará uso de **Crontab** [62]. Se trata de una herramienta para sistemas Linux, que sirve para automatizar tareas en intervalos regulares de tiempo. Puede ser muy útil para programar tareas rutinarias como chequeos de equipos, llevar a cabo backups, etc… Para hacer que el software se ejecute en cuanto se conecte la Raspberry, es necesario hacer lo siguiente.

1. En primer lugar, abrimos un terminal y escribimos lo siguiente:

```
crontab -e
```

   Este comando, nos va a permitir añadir una nueva tarea.

2. En una nueva línea del fichero que se nos abre, escribimos lo que vemos a continuación:

```
@reboot sudo python /root/Escritorio/tfm_bluetooth/Bluetooth_Carlos/bluetooth_listener.py
```

   La notación **@reboot**, indica que, lo que se encuentra a continuación, se va a ejecutar cuando se inicie la Raspberry, una vez escrito esto y guardado el fichero, ya estaría lista la Raspberry Pi para escanear dispositivos nada más conectarla a la corriente.

Opción 2. Instalación de una imagen ya preparada en la Raspberry Pi.

Esta opción es bastante más sencilla y rápida de llevar a cabo, puesto que se ha creado una **imagen de Raspbian** preparada para funcionar nada más sea instalada.

Una vez instalada esta imagen, solo habría que cambiar los parámetros de configuración comentados anteriormente, en el caso de que querer hacerlo, o hacer tareas concretas, como por ejemplo conectar el dispositivo a una red wifi o algo similar.

En la página de Raspberry, se puede ver cómo podemos instalar una imagen **.img** [63].

NOTA: **Al tratarse de un Trabajo Fin de Máster, esta imagen no será pública, al menos hasta la defensa y posterior calificación de este trabajo.**

Bibliografía

[1] «Bluetooth Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Bluetooth. [Último acceso: 15 Julio 2019].

[2] «Bluetooth,» [En línea]. Available: https://www.bluetooth.com/. [Último acceso: 15 Julio 2019].

[3] P. Cope, J. Campbell y T. Hayajneh, «An investigation of Bluetooth security vulnerabilities,» [En línea]. Available: https://ieeexplore.ieee.org/document/7868416. [Último acceso: 17 Junio 2019].

[4] «Harald Blåtand,» [En línea]. Available: https://es.wikipedia.org/wiki/Harald_Bl%C3%A5tand.

[5] «Etimoligía y logo,» [En línea]. Available: https://es.wikipedia.org/wiki/Bluetooth#Etimolog%C3%ADa_y_logo. [Último acceso: 17 Junio 2019].

[6] «BLUETOOTH: DE DÓNDE VIENE Y QUÉ SIGNIFICA SU CONTROVERSIAL LOGO,» [En línea]. Available: https://www.infotechnology.com/online/Bluetooth-de-donde-viene-y-que-significa-su-controversial-logo-20161013-0002.html. [Último acceso: 17 Junio 2019].

[7] «Bluetooth, clases y versiones desde v1.0 hasta v5.0,» [En línea]. Available: https://blog.330ohms.com/2017/02/02/bluetooth-clases-y-versiones-desde-v1-0-hasta-v5-0/. [Último acceso: 15 Julio 2019].

[8] «ENS. Seguridad en Bluetooth,» [En línea]. Available: https://www.ccn-cert.cni.es/series-ccn-stic/800-guia-esquema-nacional-de-seguridad/2707-ccn-stic-837-ens-seguridad-en-bluetooth/file.html. [Último acceso: 15 Julio 2019].

[9] «Profiles make Bluetooth technology interoperable,» [En línea]. Available: https://www.bluetooth.com/specifications/profiles-overview/. [Último acceso: 15 Julio 2019].

[10] «Bluetooth Basics,» [En línea]. Available: https://learn.sparkfun.com/tutorials/bluetooth-basics#common-versions. [Último acceso: 15 Julio 2019].

[11] «Red ad hoc inalámbrica,» [En línea]. Available: https://es.wikipedia.org/wiki/Red_ad_hoc_inal%C3%A1mbrica. [Último acceso: 17 06 2019].

[12] «Criptografía con curvas elípticas,» [En línea]. Available: http://www.criptored.upm.es/crypt4you/temas/ECC/leccion1/leccion1.html. [Último

acceso: 15 Julio 2019].

[13] «Advanced Encryption Standard,» [En línea]. Available: https://es.wikipedia.org/wiki/Advanced_Encryption_Standard. [Último acceso: 15 Julio 2019].

[14] «Bluetooth Core Specifications,» [En línea]. Available: https://www.bluetooth.com/specifications/bluetooth-core-specification/. [Último acceso: 15 Julio 2019].

[15] «CVE Bluetooth,» [En línea]. Available: https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=bluetooth. [Último acceso: 15 Julio 2019].

[16] J. Padgette, J. Bahr, M. Batra , M. Holtmann , R. Smithbey, L. Chen y K. Scarfone, «Guide to Bluetooth Security,» [En línea]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-121r2.pdf. [Último acceso: 15 07 2019].

[17] «Security Vulnerabilities in Bluetooth Technology as Used in IoT,» [En línea]. Available: https://www.mdpi.com/2224-2708/7/3/28/pdf. [Último acceso: 15 Julio 2019].

[18] «Blueborne,» [En línea]. Available: https://www.armis.com/blueborne/. [Último acceso: 15 Julio 2019].

[19] «Armis,» [En línea]. Available: https://armis.com/. [Último acceso: 15 Julio 2019].

[20] «¿Qué es un 0-day?,» [En línea]. Available: https://www.welivesecurity.com/la-es/2015/02/25/que-es-un-0-day/. [Último acceso: 15 Julio 2019].

[21] «Armis - BlueBorne Explained,» [En línea]. Available: https://www.youtube.com/watch?v=LLNtZKpL0P8. [Último acceso: 15 Julio 2019].

[22] «BlueBorne Vulnerabilities Impact Amazon Echo and Google Home.,» [En línea]. Available: https://www.armis.com/resources/iot-security-blog/blueborne-cyber-threat-impacts-amazon-echo-google-home/. [Último acceso: 15 Julio 2019].

[23] «Blueborne Technical Paper,» [En línea]. Available: https://go.armis.com/blueborne-technical-paper. [Último acceso: 15 Julio 2019].

[24] «Bluez,» [En línea]. Available: http://www.bluez.org/. [Último acceso: 15 07 2019].

[25] «PyBluez,» [En línea]. Available: https://github.com/karulis/pybluez. [Último acceso: 15 07 2019].

[26] «PyGattlib,» [En línea]. Available: https://bitbucket.org/OscarAcena/pygattlib/src/default/. [Último acceso: 15 07 2019].

[27] «BluePy,» [En línea]. Available: https://github.com/IanHarvey/bluepy. [Último acceso:

15 Julio 2019].

[28] «Web-Bluetooth,» [En línea]. Available: https://github.com/WebBluetoothCG/web-bluetooth. [Último acceso: 15 07 2019].

[29] «Libblepp,» [En línea]. Available: https://github.com/edrosten/libblepp. [Último acceso: 15 07 2019].

[30] «Bluetooth - Manager,» [En línea]. Available: https://github.com/sputnikdev/bluetooth-manager. [Último acceso: 15 07 2019].

[31] «Ubertooth Zero,» [En línea]. Available: http://ubertooth.sourceforge.net/hardware/zero/. [Último acceso: 15 07 2019].

[32] «Ossmann Webpage,» [En línea]. Available: http://www.ossmann.com/mike/. [Último acceso: 15 Julio 2019].

[33] «Great Scott Gadgets,» [En línea]. Available: https://greatscottgadgets.com/. [Último acceso: 15 Julio 2019].

[34] «ShmooCon 2011: Project Ubertooth: Building a Better Bluetooth Adapter,» [En línea]. Available: https://www.youtube.com/watch?v=KSd_1FE6z4Y. [Último acceso: 22 07 2019].

[35] «Ubertooth,» [En línea]. Available: https://github.com/greatscottgadgets/ubertooth. [Último acceso: 15 07 2019].

[36] «Ubertooth One,» [En línea]. Available: http://ubertooth.sourceforge.net/hardware/one/. [Último acceso: 15 07 2019].

[37] «LPC17x,» [En línea]. Available: https://www.nxp.com/docs/en/data-sheet/LPC1759_58_56_54_52_51.pdf. [Último acceso: 15 07 2019].

[38] «Schematic Ubertooth,» [En línea]. Available: https://cdn.sparkfun.com/datasheets/Dev/ARM/SchematicUbertooth.pdf. [Último acceso: 15 07 2019].

[39] «Ubertooth One Amazon,» [En línea]. Available: https://www.amazon.es/Ubertooth-One-Antena-color-negro/dp/B007R9UPHA. [Último acceso: 15 07 2019].

[40] «Python,» [En línea]. Available: https://www.python.org/. [Último acceso: 15 Julio 2019].

[41] «MySQL,» [En línea]. Available: https://es.wikipedia.org/wiki/MySQL. [Último acceso: 15 07 2019].

[42] «MySQL,» [En línea]. Available: https://www.mysql.com/. [Último acceso: 15 Julio 2019].

[43] «DataGrip. Many databases, one tool,» [En línea]. Available: https://www.jetbrains.com/datagrip/. [Último acceso: 15 Julio 2019].

[44] «JetBrains,» [En línea]. Available: https://www.jetbrains.com. [Último acceso: 15 Julio 2019].

[45] «Free individual licenses for students and faculty members,» [En línea]. Available: https://www.jetbrains.com/student/. [Último acceso: 15 Julio 2019].

[46] «Pycharm. The Python IDE for Professional Developers,» [En línea]. Available: https://www.jetbrains.com/pycharm/. [Último acceso: 15 Julio 2019].

[47] «What is RSSI and its acceptable signal strength?,» [En línea]. Available: https://helpcenter.engeniustech.com/hc/en-us/articles/234761008-What-is-RSSI-and-its-acceptable-signal-strength-. [Último acceso: 15 Julio 2019].

[48] «Blueborne Device List,» [En línea]. Available: https://github.com/hook-s3c/blueborne-scanner/blob/master/classes/deviceslist.py. [Último acceso: 15 Julio 2019].

[49] «IEEE Registration Authority: Assignments,» [En línea]. Available: https://regauth.standards.ieee.org/standards-ra-web/pub/view.html#registries. [Último acceso: 15 Julio 2019].

[50] «Raspberry Pi,» [En línea]. Available: https://www.raspberrypi.org/. [Último acceso: 15 Julio 2019].

[51] «Raspberry Pi Model 3 B+ Starter Pack,» [En línea]. Available: https://shop.bigclown.com/raspberry-pi-model-3-b-plus-starter-pack/. [Último acceso: 15 Julio 2019].

[52] «Dos millones de razones para saber qué es exactamente Raspberry Pi,» [En línea]. Available: https://www.elconfidencial.com/tecnologia/2013-11-22/dos-millones-de-razones-para-saber-que-es-exactamente-raspberry-pi_56003/. [Último acceso: 15 Julio 2019].

[53] «Raspberry Pi 3 Model B+ Review – What's New?,» [En línea]. Available: https://makeradvisor.com/raspberry-pi-3-model-b-plus-review/. [Último acceso: 15 Julio 2019].

[54] «Blueborne - Android Take Over Demo,» [En línea]. Available: https://www.youtube.com/watch?v=Az-l90RCns8. [Último acceso: 15 Julio 2019].

[55] «BlueBorne - Windows MiTM Demo,» [En línea]. Available: https://www.youtube.com/watch?v=QrHbZPO9Rnc. [Último acceso: 15 Julio 2019].

[56] «Armis - Blueborne - GitHub,» [En línea]. Available: https://github.com/ArmisSecurity/blueborne. [Último acceso: 15 Julio 2019].

[57] «BlueBorne RCE on Android 6.0.1 (CVE-2017-0781) [English],» [En línea]. Available: https://jesux.es/exploiting/blueborne-android-6.0.1-english/. [Último acceso: 15 Julio 2019].

[58] «Guía para proteger y usar de forma segura su móvil,» [En línea]. Available: https://www.incibe.es/extfrontinteco/img/File/intecocert/Proteccion/usoseguromoviles.pdf. [Último acceso: 15 07 2019].

[59] J. P. Dunning, «Bluetooth Threat Taxonomy,» [En línea]. Available: https://vtechworks.lib.vt.edu/bitstream/handle/10919/76883/etd-10242010-163002_Dunning_JP_T_2010.pdf?sequence%20=1&isAllowed=y. [Último acceso: 15 Julio 2019].

[60] «National Institute of Standards and Technology,» [En línea]. Available: https://www.nist.gov/. [Último acceso: 15 Julio 2019].

[61] «Raspbian,» [En línea]. Available: https://www.raspberrypi.org/downloads/raspbian/. [Último acceso: 15 07 2019].

[62] «Crontab in Linux with 20 Useful Examples to Schedule Jobs,» [En línea]. Available: https://tecadmin.net/crontab-in-linux-with-20-examples-of-cron-schedule/. [Último acceso: 15 Julio 2019].

[63] «Installing operating system images,» [En línea]. Available: https://www.raspberrypi.org/documentation/installation/installing-images/README.md. [Último acceso: 15 Julio 2019].

[64] «Indicador de fuerza de la señal recibida (RSSI),» [En línea]. Available: https://es.wikipedia.org/wiki/Indicador_de_fuerza_de_la_se%C3%B1al_recibida. [Último acceso: 15 Julio 2019].

[65] «Draw.io,» [En línea]. Available: https://www.draw.io/. [Último acceso: 15 Julio 2019].

[66] «Universidad de Jaén. Cómo llegar,» [En línea]. Available: http://150.214.178.183/comollegar.html. [Último acceso: 15 Julio 2019].

[67] «Imagen,» [En línea]. Available: https://www.google.com/url?sa=i&source=images&cd=&ved=2ahUKEwjXvruhsPPkAhUJ3OAKHcPJDG4QjRx6BAgBEAQ&url=http%3A%2F%2Fgeomaticaes.com%2Festudia-geomatica%2Fescuelas%2Fescuela-politecnica-superior-de-jaen-departamento-de-ingenieria-cartografica-geodesica-y-f. [Último acceso: 15 Julio 2019].