PENTESTING IOT DEVICE: SMART DOORLOCK

PENTESTING IOT DEVICE: SMART DOORLOCK

TRABAJO DE FIN DE MÁSTER EN INTERNET DE LAS COSAS
DEPARTAMENTO DE INFORMÁTICA

AUTOR
JESÚS ALBERTO TEJEDOR DORIA

DIRECTOR
JOAQUIN RECAS PIORNO
GUILLERMO BOTELLA JUAN

CONVOCATORIA: SEPTIEMBRE
CALIFICACIÓN: 9

MÁSTER EN INTERNET DE LAS COSAS
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

23DESEPTIEMBREDE2020

DEDICATORIA

A todas las personas que me apoyan día a día, para seguir mejorando tanto en lo personal como en lo profesional.

AGRADECIMIENTOS

A Irene por su infinita paciencia y apoyo incondicional en la realización del master y este trabajo en un año tan complicado como este. A mi familia por su s ánimos y fuerza para elaboración de este trabajo. A mi amigo Mariano por ayudarme con los problemas de la herramienta objection y regalarme su tiempo y talento. A Joaquin por su ayuda en todo el proceso del trabajo y por incentivarme y guiarme para mejorar.

RESUMEN

Pentesting IoT Device: Smart Doorlock

El panorama actual demuestra que cada vez existe una mayor dependencia del uso de los dispositivos IoT en todos los ámbitos. Pero un aspecto que no está siendo gestionado correctamente, es el de la seguridad lógica, encontrándose un gran número de dispositivos vulnerables en el mercado.

Por lo tanto, es necesario establecer estándares y patrones para el diseño de los dispositivos a la par que, realizar auditorías previas a su comercialización, para dotar a los dispositivos de un examen que pueda verificar que como mínimo se ha revisado la seguridad de los mismos. Actualmente existen muchas recomendaciones sobre estándares de diseño por parte de organismos como ISACA o OWASP, pero no existe un estándar mundial para el sector.

En el siguiente trabajo se realiza un estudio que refleja la necesidad de seguir estos estándares y realizar las auditoras. En especial de los dispositivos basados en Bluetooth ya que en el sector de los dispositivos IoT el uso de la tecnología Bluetooth es muy común, por lo que es necesario hacer hincapié en las vulnerabilidades existentes y los distintos tipos de ataques. Con la realización de una auditoría de test de penetración (Pentest) sobre el candado inteligente se demuestra la necesidad de hacer estas auditorías antes de poner este tipo de productos a la venta.

Palabras clave

Bluetooth, BLE, Seguridad, IoT, Hacking, Dispositivos, Pentesting, ISACA, OWASP, Candado

ABSTRACT

PENTESTING IOT DEVICE: SMART DOORLOCK

The current situation shows us that there is an increasing dependence on the use of IoT devices in every environment. However, one aspect that has not been correctly managed is security.

Therefore, it is necessary to establish standards and patterns in the design stage of the devices, as well as conducting ethical hacking audits, to make sure that they are secure to use. Currently there are many recommendations from organizations such as ISACA or OWASP, but there is not a global standard for the sector.

The use of Bluetooth as a technology in IoT devices is very usual, so here is the reason to insist on the actual existing vulnerabilities and attacks. Given the ethical hacking audit did on the smart lock, it demonstrates the need to do these audits before putting these products on sale.

Keywords

Bluetooth, BLE, Seguridad, IoT, Hacking, Dispositivos inteligentes, Pentesting

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. Introducción

Este trabajo pretende demostrar la necesidad de tener en cuenta la seguridad desde las fases más tempranas del diseño. Esto se debe a la creciente amenaza por parte de la ciberdelincuencia, por lo que las empresas están invirtiendo grandes sumas de dinero en la protección de sus activos.

Sin embargo, no se aplica de la misma forma al sector de los dispositivos conectados, pensando que es más importante proteger una base de datos con información de clientes, que un dispositivo, pudiendo ser algo tan crítico como un marcapasos, que si fuera "hackeado" podría tener graves consecuencias, debido a ello, en este proyecto se selecciona un candado inteligente para realizarle una auditoria de hacking ético que es la acción de una persona que utiliza sus conocimientos de informática para encontrar vulnerabilidades en el sistema [1], ya que es un dispositivo destinado a la protección de un activo.

Se demostrará como con las mismas herramientas software y los mismos tipos de ataques que se utilizan en los sistemas convencionales se puede tomar el control del mismo.

El objetivo, es tener una visión a alto nivel del estado actual de los dispositivos IoT, en especial, de los dispositivos basados en tecnologías BLE y conocer cuáles son sus características, riesgos y como mitigarlos. Para una compresión más robusta de la problemática, el trabajo consta de una parte práctica donde se realiza una demostración de una auditoría basada en un laboratorio de la empresa Attify [2].

En el trabajo que se presenta, se contextualiza la problemática expuesta en los anteriores párrafos, para posteriormente realizar una demostración práctica y finalizar con las recomendaciones de seguridad pertinentes.

2. Estado de la cuestión

Mostrar un enfoque sobre la relevancia de los dispositivos IoT en la actualidad, qué se espera de este sector en los próximos años, cuáles son sus principales usos y la importancia del ámbito de la seguridad en los dispositivos. Abordando múltiples puntos de vista como la auditoría de los dispositivos o los riesgos que se pueden hallar.

Análisis del sector IoT

La empresa Gartner, expone que a finales del 2020 se utilizarán 5.8 mil millones de IoT Endpoints empresariales. Esto representará un aumento del 21% frente al año 2019 y a su vez, representará un 21,5% respecto al año 2018. [3]

Las empresas de servicios públicos serán los mayores usuarios de dispositivos IoT con un total de 1.37 mil millones de dispositivos para finales del año 2020. [3] Siendo la mayor parte de estos dispositivos utilizados para la medición inteligente de electricidad, tanto en el uso residencial como comercial, esto impulsará la adopción del IoT entre empresas del sector público que también ofrecen servicios como agua o gas.

Otro sector que se verá fuertemente impactado, es el sector de la seguridad física o la automatización de edificios impulsada por dispositivos de iluminación conectados, este último será el segmento con la mayor tasa de crecimiento en 2020 con un 42%, seguidos por la automoción (31%) y la atención médica (29%) [3]. Tabla 1, se muestra un ejemplo del incremento de dispositivos por sectores a nivel mundial, el dato está reflejado en miles de millones (1.000.000.000) de unidades.

Segmento	2018 (Unidades en miles de millones)	2019 (Unidades en miles de millones)	2020 (Unidades en miles de millones)	% Incremento desde 2018 a 2020
Utilidades	0,98	1,17	1,37	39,80
Gobierno	0,4	0,53	0,7	75,00
Automatización de edificios	0,23	0,31	0,44	91,30
Seguridad Física	0,83	0,95	1,09	31,33
Fabricación y recursos naturales	0,33	0,4	0,49	48,48
Automotor	0,27	0,36	0,47	74,07
Proveedores de servicios de salud	0,21	0,28	0,36	71,43
Comercio minorista y mayorista	0,29	0,36	0,44	51,72
Información	0,37	0,37	0,37	0,00
Transporte	0,06	0,07	0,08	33,33
Total	**3,96**	**4,81**	**5,81**	**46,72**

Tabla 1 - Segmento de dispositivos instalados (Recurso 2018-2020, Worldwide*) [3]*

Es importante entender que los datos globales reflejados en la Tabla 1, sufren variaciones en función de la parte del mundo a la cual hagamos referencia, dado que no es el mismo tipo de inversión la que se realiza en los países desarrollados que en los países en vías de desarrollo.

Mientras que, en 2020, la medición inteligente de electricidad residencial es el principal caso de uso en China y Europa, (representando el 26% y el 12% del total de dispositivos respectivamente), en EEUU se encuentra un caso de uso muy distinto. En este país el principal caso de éxito es la seguridad física, con la implantación de sensores para la detección de intrusos, representando el 8% del total de dispositivos IoT. [3]

Seguridad en dispositivos IoT

Para entender lo crítico que es el aspecto de la seguridad en los dispositivos IoT, se deben contextualizar algunos puntos:

Falta de estándar por parte del sector

Actualmente, no hay estándares de la industria o regulaciones del gobierno a nivel mundial para abordar el aspecto de la seguridad, esto es debido a que los fabricantes se encuentran en una guerra comercial por imponer sus propios estándares y, por lo tanto, no hay ninguna entidad que esté regulando esto a día de hoy.

A su vez, se está incentivando más el diseño del dispositivo a nivel visual y el "time to market", que la seguridad del mismo.

Dado que el firmware es uno de los principales vectores de ataque cuando se intenta realizar un "hackeo" a un dispositivo IoT, la industria debe realizar auditorías del firmware para comprobar que este es seguro, ya que, en caso de ser vulnerable, se podría tomar el control del dispositivo a nivel de hardware.

Principales problemas de seguridad

Para entender hasta qué punto la seguridad de los dispositivos es un problema, no se debe olvidar que no es un problema asociado al dispositivo, sino a todo el entorno que lo rodea, ya que al incluir un elemento inseguro en un ecosistema seguro este compromete a todo el ecosistema-

Este problema, es muy parecido al que ocurrió con los "módems" y posteriormente con los "routers wifis" domésticos, donde las redes estaban abiertas y el tráfico no estaba cifrado, esto hacía muy complejo poder respetar la integridad y la confidencialidad del dato.

Dado que los dispositivos IoT muchas veces son desplegado en entornos complejos, tanto a nivel de infraestructura o por la falta de suministros energéticos, el enfoque de la seguridad física vuelve a convertirse en algo totalmente principal. Es importante poner énfasis en esta cuestión, ya que no estamos hablando de un servidor en el sótano de un edificio, con todas las medidas de protección físicas que se le

presuponen, como circuito cerrado de video vigilancia, puertas, empleados de seguridad, si no que los dispositivos pueden estar al alcance de cualquiera, como por ejemplo una boya de uso marítimo.

Aun así, se siguen encontrando múltiples problemas cuando los dispositivos si pueden ser instalados en un entorno controlado, y es entonces, cuando se deja de lado la seguridad física y pasamos a la seguridad lógica, o más conocida como ciberseguridad.

El estudio realizado por la empresa Paloalto Network y publicado en marzo del 2020, ha mostrado a la comunidad el gran problema que existe con los dispositivos IoT, ya que el estudio está basado en 1.2 millones de dispositivos [4].

Dicho estudio realiza un análisis de estos dispositivos en empresas de tecnologías de la información y atención médica, con el fin de obtener una "foto" de la situación actual y poder lanzar las recomendaciones necesarias, para que las organizaciones puedan tomar medidas para reducir su exposición y por lo tanto el riesgo actual. El dato que más llama la atención es que el 83% de dispositivos de captura de imágenes médicas se ejecutan en sistemas operativos que ya no disponen de soporte, como se puede ver en Ilustración 1. [4]

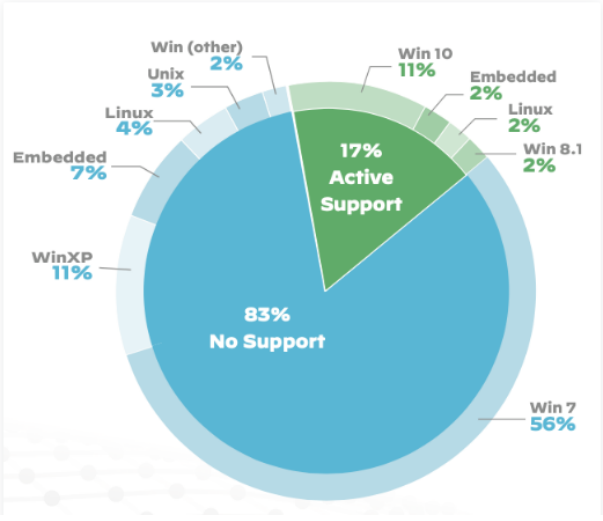

Ilustración 1 - Dispositivos IoT en SO soportados (Gráfico PaloAlto Network) [4]

3. Capitulo Bluetooth Low Energy en dispositivos IoT.

Este capítulo define el origen y refleja las principales características del BLE (Bluetooth Low Energy) y cuál es su relevancia en los dispositivos IoT.

Bluetooth Low Energy

BLE es una tecnología de red de tipo PAN (Personal Area Network), diseñada y comercializada por el consorcio Bluetooth Special Interest Group [5]. Este consorcio fue formado en el año 1998 por las empresas Ericsson, IBM, Intel, Toshiba y Nokia. Adoptaron el nombre de Bluetooth para su especificación, con el paso de los años se fueron añadiendo y saliendo del consorcio diferentes empresas [6], y a su vez se publicaban nuevas versiones de Bluetooth donde en 2010 la versión 4.0 ya incluía la tecnología BLE.

Esta tecnología surgió para ser utilizada en aplicaciones novedosas, sobre todo en el ámbito salud, fitness, seguridad y en el sector de dispositivos para el hogar.

BLE en el ecosistema IoT

En la actualidad, la tecnología BLE forma parte del día a día de los consumidores. Por ejemplo, hace unos años se pusieron de moda los cascos inalámbricos o los famosos "manos libres", pero coloquialmente los consumidores lo llamaron Bluetooth, con frases como "¿Puedo conectar el bluetooth al coche?", o "poner el bluetooth para hablar", "altavoces bluetooth para la playa o la ducha", obviando, por abuso del lenguaje, que esta palabra pudiese hacer referencia a la tecnología que utilizan dichos dispositivos para comunicarse con el exterior y así evitar el uso de cables.

En la actualidad, podemos encontrar esta tecnología en cualquier domicilio, se pueden encontrar múltiples dispositivos que funcionan vía BLE como cafeteras, aspiradoras, bombillas, etc. De la misma forma podemos encontrar las famosas pulseras Fitbit, que se sincronizan con los Smartphone, en la muñeca de cualquier ciudadano, también podemos observar de manera generalizada en muchos hogares el uso de dispositivos más sofisticados que utilizan esta característica para la transferencia de

información, como por ejemplo, los monitores remotos que controlan la apnea del sueño con fines médicos [7].

En la industria, la tecnología BLE, se utiliza principalmente para las comunicaciones entre maquinas, previo a enviar esta información a una puerta de enlace, pudiéndose observar su uso, entre otros muchos, en la gestión de residuos urbanos.

Características BLE

La principal diferencia entre el Bluetooth clásico y el BLE, es que este último está diseñado para tener un bajo consumo de energía y costes muy reducidos, manteniendo ambos básicamente el mismo rango de alcance. En cambio, la forma de operar es muy distinta, ya que el Bluetooth clásico estaba diseñado para la transmisión continua de información, mientras que el BLE permanece en modo suspensión hasta que es necesario, de esta forma las aplicaciones pueden funcionar con una batería muy pequeña durante mucho tiempo.

El Bluetooth clásico opera en el rango de 2400-2483.5 MHz dentro de la banda de frecuencia ISM 2,4GHz. Los datos se dividen en paquetes y se intercambian a través de uno de los 79 canales, donde cada canal dispone de 1MHz de ancho de banda [8].

El BLE también ha conseguido mejorar los tiempos de conexión. Mientras que el BLE solo necesita unos pocos ms para establecer una conexión, el Bluetooth clásico necesita alrededor de 100ms. Esta comparación se observa claramente en la Ilustración 2.

As of DEC, 2016	Bluetooth™		
Specifications	**Classic Bluetooth**	**Bluetooth Low Energy (V 4.2)**	**Bluetooth 5**
Range	100 m	Greater than 100 m	Greater than 400 m
Data Rate	1-3 Mbps	1 Mbps	2 Mbps
Application Throughput	0.7 -2.1 Mbps	0.27 Mbps	—
Frequency	2.4 GHz	2.4 GHz	—
Security	56/128-bit	128-bit AES with Counter Mode CBC-MAC	—
Robustness	Adaptive fast frequency hopping, FEC, fast ASK	24-bit CRC, 32-bit Message Integrity Check	—
Latency	100 ms	6 ms	—
Time Lag	100 ms	3 ms	—
Voice Capable	Yes	No	—
Network Topology	Star	Star	—
Power Consumption	1 W	0.01 to 0.5 W	—
Peak Current Consumption	less than 30 mA	less than 15 mA	—

Ilustración 2 - Comparación versión Bluetooth (origen: everythingrf) [9]

Características técnicas:

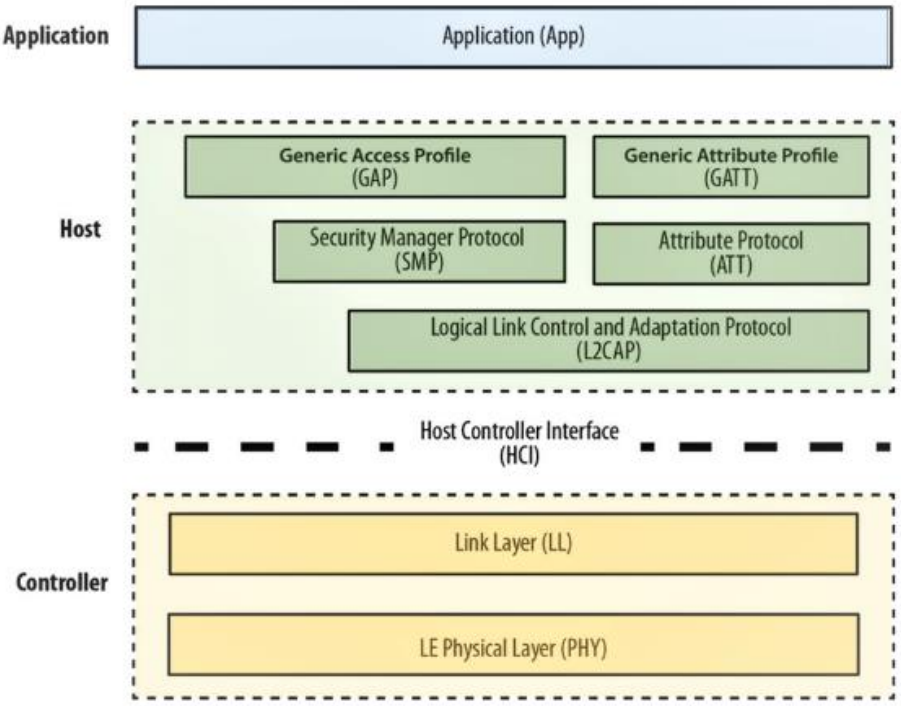

Ilustración 3 - Protocol Stack BLE (recurso: researchgate) [10]

A nivel del HOST (véase Ilustración 3) se encuentran las capa que permiten interactuar entre el controlador y la aplicación:

GAP (Generic Access profile): Controla las conexiones y las publicaciones en Bluetooth. GAP permite que un dispositivo sea visible por el resto, a su vez, establece cómo el dispositivo puede interactuar con el resto.

La capa define varios roles que pueden adquirir los dispositivos, llamados dispositivos periféricos y centrales, (véase Ilustración 4):

- **Periféricos**: Se suele asignar este rol a dispositivos de baja potencia y poco procesamiento, normalmente solo recogen el dato con el objetivo de enviarlo a un dispositivo central más potente para que este lo procese.

- **Centrales**: Suelen ser dispositivos con muchas más capacidades de procesamiento, de manera general sin restricciones energéticas, ya que son un eslabón clave de la red porque deben conectarse a los dispositivos periféricos.

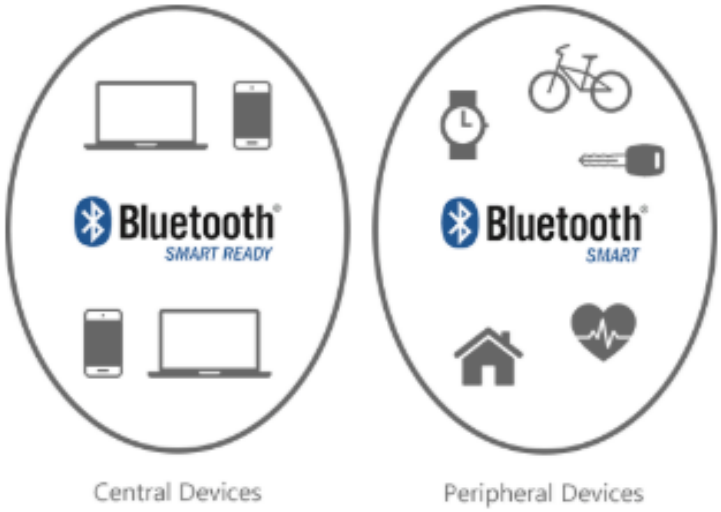

Ilustración 4 - Bluetooth central y periféricos

GATT (Generic Attribute Profile): Este término hace referencia a cómo dos dispositivos Bluetooth transmiten datos de un extremo a otro. Para realizar esta comunicación, utiliza el protocolo de datos genérico **ATT (Atrribute Protocolo),** Estos datos se organizan jerárquicamente en servicios, Las funciones de esta capa, comienzan cuando se establece la conexión entre 2 dispositivos.

Hay que tener en cuenta que cuando se realiza un emparejamiento estas conexiones son exclusivas, por lo que un dispositivo con el rol de periférico solo se puede conectar a un dispositivo con el rol de central.

Para intercambiar datos entre 2 dispositivos con el rol de periférico es necesario pasar por el dispositivo central, (véase Ilustración 5).

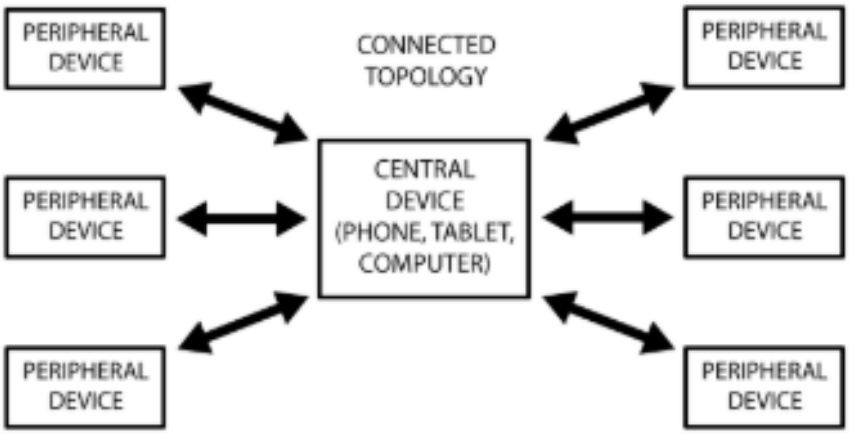

Ilustración 5 - Topología GATT (recurso: learn.adafruit) [11]

Seguridad en BLE

SMP (Security Manager Protocol), es un protocolo que proporciona un entorno para generar y distribuir claves de seguridad entre dos dispositivos IoT.

Cuando la comunicación llega a la capa host e interviene el protocolo SMP, es el momento más vulnerable para un dispositivo BLE, ya que es cuando se realiza el emparejamiento y el intercambio de claves, es decir, el dispositivo central realiza la conexión contra el periférico, pero los dispositivos periféricos no pueden conectarse directamente entre ellos, ni al dispositivo central. (véase Ilustración 6).

Para realizar el emparejamiento se definen 2 roles:

- Inicia: Este rol se corresponde con el rol maestro de la capa de enlace.
- Responde: Este rol corresponde con el rol esclavo de la capa de enlace.

A su vez, el protocolo SMP se apoya en 3 procedimientos para realizar este proceso de intercambio:

- Establecimiento conexión: Se genera una clave de cifrado común y temporal que habilita que se pueda pasar a una conexión cifrada y por lo tanto una conexión segura. Esta clave es de un solo uso por lo que no es necesario almacenarla ni puede ser utilizada posteriormente.

- Emparejamiento: Se crea el emparejamiento de ambos dispositivos y posteriormente se realiza el intercambio de claves de seguridad permanentes. Estas claves se almacenan para poder ser usadas en futuras conexiones.
- Establecimiento cifrado: Define como deben ser utilizadas las claves de cifrado en futuras conexiones (si estas han sido almacenadas), para no tener que volver a pasar por el paso de vinculación. De esta forma se restablece de nuevo una conexión segura.

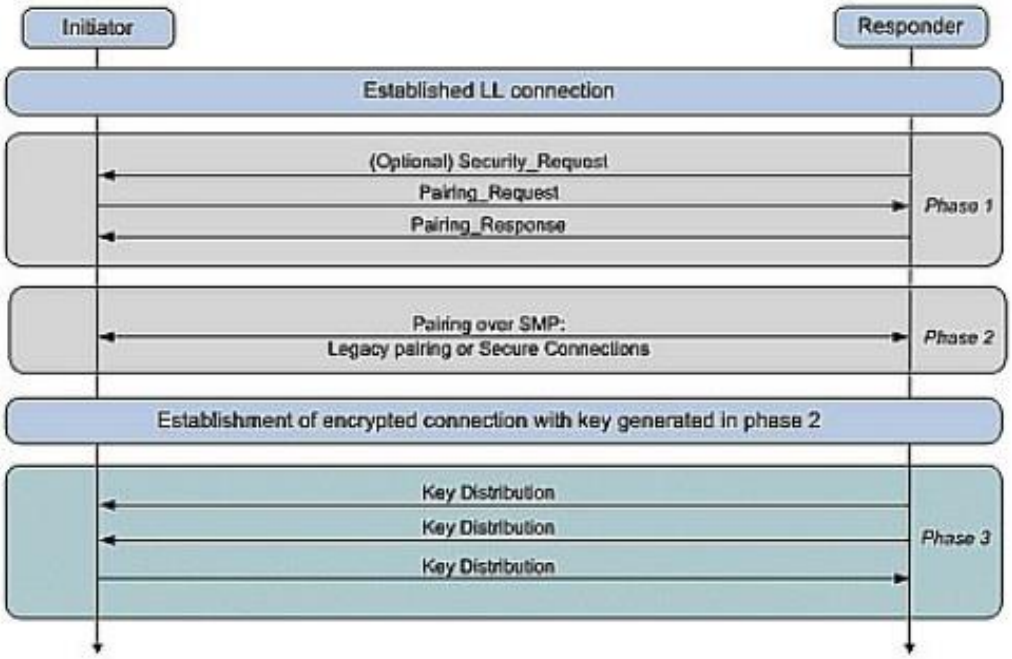

Ilustración 6 - Proceso de emparejamiento BLE (Recurso: microcontrollertips) [12]

En la Tabla 2 se muestran los mecanismos de seguridad para Bluetooth Low Energy.

En la versión 2.1 de BLE se estableció el mecanismo de seguridad Legacy Pairing para las siguientes características:

- Emparejamiento
- Unión
- Autenticación de dispositivo
- Cifrado de mensajes
- Integridad del mensaje

En la versión 4.0 se adoptó exactamente el mismo modelo de seguridad con las siguientes excepciones:

- Sin equivalencia de comparación numérica: Se muestra un número, generalmente de 6 dígitos en la pantalla de cada dispositivo y el usuario comprueba que ambos números coinciden.
- Just Works y Passkey Entry no ofrecen protección para la escucha, ya que ECDH (Elliptic Curve Diffie-Hellman) no se usa en la versión 4.0, esto permite que en la generación de claves se utilice una clave temporal que puede ser capturada con facilidad si es intercambiada con un dispositivo no cifrado.

La principal diferencia entre Legacy Pairing utilizado en la versión 4.0 y 4.1 y Secure Connections que es utilizada a partir de la versión 4.2 es el uso de ECDH para realizar el emparejamiento, esto permite que se utilicen un algoritmo de encriptación para la generación de las claves, generando cada dispositivo que va a participar en el emparejamiento su propias claves públicas y privadas que van a ser compartidas dotando al mecanismo de mayor robustez.

Para el uso de los 2 métodos de seguridad en BLE se necesitan crear las siguientes claves:

LE Legacy Pairing: La clave que genera para el cifrado de la conexión, es una clave de 128 bits llamada STK (Short Term Key). Para poder generarla los dispositivos deben disponer previamente de una clave temporal TK (Temporary Key).

LE Secure Connections: La clave que genera para el cifrado de la conexión, es una clave de 128 bits llamada LTK (Long Term Key), que es almacenada en los dispositivos y es utilizada para conexiones futuras entre ambos dispositivos sin necesidad de llevar acabo nuevamente el proceso de pairing.

	Bluetooth LE	
	LE Legacy Pairing	**LE Secure Connections**
Versiones	4.0 / 4.1	A partir de 4.2
Generación de claves (Pairing)	• Generación de STK. • Uso de TK, clave temporal. • 3 Métodos Pairing (Passkey Entry, OOB, Just Works). • Algoritmos de generación de claves AES-128.	• Generación de LTK. • 4 Métodos Pairing (Comparación Numérica, Passkey Entry, OOB, Just Works). • ECDH Acuerdo de clave (P-256 Elliptic Curve). • Algoritmos de generación de claves basados en AES-CMAC. **basados en AES-CMAC**
Autenticación de Dispositivos	AES-CCM	AES-CCM
Cifrado	AES-CCM	AES-CCM
Integridad	AES-CCM	AES-CCM
Protección frente a eavesdropping	No. No usa ECDH para la generación de la clave de cifrado STK. Esta se genera usando una clave temporal TK, que puede ser capturada al ser intercambiada por un enlace no cifrado. Solo cabe la posibilidad de protección para el caso del método de asociación OOB cuando el canal OOB proporciona esta protección.	Sí. Utiliza algoritmos ECDH (P-256) para la generación de la clave DHKey compartida entre los dispositivos. Esta DHKey se utiliza para generar la clave de cifrado LTK. Dado que la DHKey no es distribuida, es muy difícil averiguar la LTK.

Tabla 2 - Resumen de los mecanismos de seguridad Bluetooth LE (recurso: ccn-cert 2018) [13]

Vulnerabilidades conocidas BLE

Las vulnerabilidades más comunes están basadas en las siguientes fallas de seguridad, normalmente, siempre se dan en la segunda fase del proceso de emparejamiento:

- Visibilidad del dispositivo por defecto.
- Dispositivos sin autenticación de usuario.
- Cuando se realiza el emparejamiento no se almacenan las claves de forma segura.

Los ataques más comunes a los dispositivos basados en BLE:

- Sniffing: Se captura el tráfico con la intención de revisar los paquetes en busca de información, si la conexión no ha sido cifrada previamente los datos viajarían en claro.
- Replay: Se captura el tráfico no cifrado y se reenvía este mismo mensaje al dispositivo con la intención de replicar comandos intentando escribir en sus características.
- Hijacking: Se consigue suplantar la conexión de un dispositivo legitimo generando timeout's, que se envían al dispositivo con la intención desemparejarlo, de esta forma se suplanta la dirección física del dispositivo legitimo para vincularse como si fuera él.

Bluetooth Mesh

El bluetooth mesh es una topología de red basada en malla (mesh) (vease Ilustración 7), que permite realizar comunicaciones de dispositivos de varios a varios (m:m).

Esta topología nace para cubrir las necesidades y los retos que suponen el desarrollo de los mercados emergentes, como los "smart building", "smart industry", "smart cities" y "smart home". Permitiendo la creación de redes a gran escala, para poder realizar la interconexión de cientos o miles de dispositivos entre sí.

Esta topología nace en julio de 2017, y desde su lanzamiento, no ha parado de adaptarse a dispositivos existentes, al igual que los nuevos dispositivos nacen con esta capacidad desde el inicio.

No todos los dispositivos de la red tendrán que ser capaces de reenviar la señal ya que deben seguir con un consumo mínimo. Un ejemplo de uso son los termostatos para regular la temperatura de los edificios. [14]

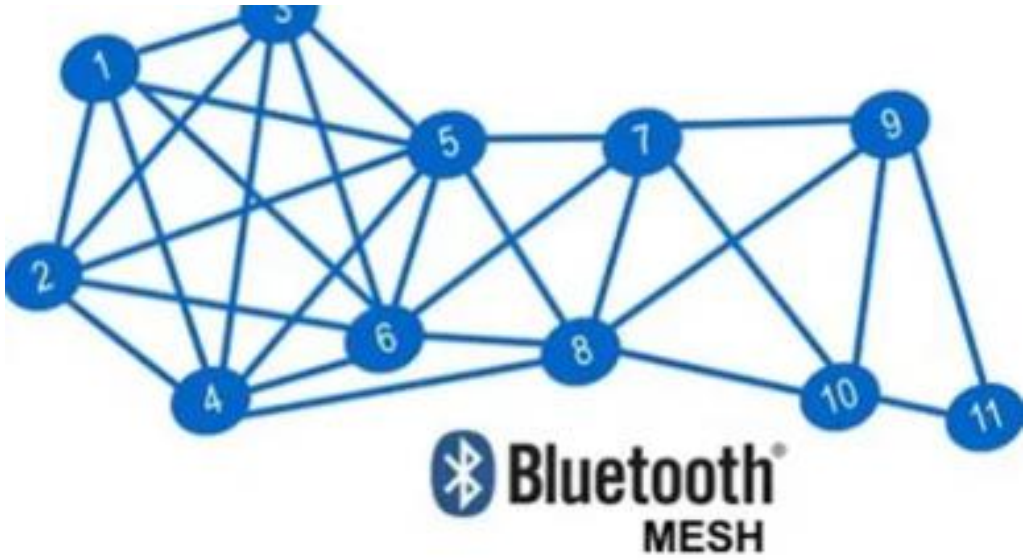

Ilustración 7 - Malla Bluetooth (origen: mobilityarena) [14]

4. Capítulo Enfoque para realizar una auditoría sobre un dispositivo IoT

Este capítulo revisa cómo enfocar una auditoría basada en dispositivos IoT y el ecosistema que los rodea a la hora de realizar una auditoría de seguridad, se debe tener en cuenta que cuando se realiza una auditoria IoT existen los mismos factores que en un ecosistema tradicional.

Tema de la auditoría

Lo más importante es determinar cuál va a ser el tema de la auditoría, es cierto que puede ser fácil en otros ecosistemas, como una auditoría WEB, pero no es trivial cuando se refiere a la auditoría de dispositivos, ya que se puede estar hablando de auditar el hardware, el software y/o las interconexiones.

Para ejemplificarlo, no es lo mismo auditar si el sistema de comunicación está cifrado, y por lo tanto es seguro, que si se pudiese realizar un desbordamiento de buffer.

Objetivo de la auditoría

El objetivo de la auditoría es responder a la pregunta ¿Por qué se es necesario auditar ese dispositivo?

Normalmente, desde el punto de vista de un auditor, todo va a estar basado en los riesgos y las acciones a acometer para eliminarlos o mitigarlos. ISACA [15] categoriza los riesgos en 3 tipologías, Tabla 3.

Categoría Riesgo	Ejemplo
Negocio	• **Regulación y Normativa** • **Privacidad de los usuarios**
Operacional	• **Rendimiento** • **Acceso a funcionalidades**
Técnico	• **Dispositivos vulnerables** • **Dispositivos actualizados** • **Gestión del dispositivo**

Tabla 3 - Riesgos auditoría IoT (Elaboración propia a partir: ISACA) [16]

Dado que la parte práctica del proyecto se centra en las vulnerabilidades lógicas, únicamente se profundiza en el riesgo técnico, que se puede definir como:

- Actualizaciones de software y parches. El tiempo para el lanzamiento de un parche puede ser más largo que el ciclo típico para dispositivos que no son IoT, esto es debido a que el dispositivo muchas veces no tiene la capacidad para poder recibir la actualización del firmware de forma remota, uno de los posibles motivos es el tamaño o la capacidad de procesamiento, y por lo tanto es necesario tener acceso físico al dispositivo.
- Vida útil del hardware. Los dispositivos IoT tienen su propio ciclo de vida, a menudo con obsolescencia incorporada. Los componentes, como las baterías no reemplazables en los dispositivos IoT, requieren procesos de planificación del ciclo de vida y gestión de activos específicos para IoT.
- Normalmente las identificaciones de usuario y las contraseñas para controlar el acceso no existen o no están codificadas y cuando existen puede que no haya una buena gestión de las mismas como falta de caducidad o robustez.
- Algunos dispositivos de IoT pueden piratearse rápidamente, y pueden tardar días o semanas en rectificarse. Las consecuencias más amplias siguen siendo desconocidas porque es difícil saber qué se ha visto, modificado o robado

dado que muchos de ellos no tienen la capacidad de tener una gestión robusta de los logs.

- Los ciberdelincuentes pueden plantar puertas traseras para futuros ataques automatizados en o desde dispositivos IoT. Los ataques típicos incluyen ataques de denegación de servicio (DDoS) distribuidos por botnet.
- Los hackers pueden usar dispositivos IoT como punto de entrada a las redes de una empresa [16].

Alcance de la auditoría

El concepto del alcance es muy importante a la hora de abordar una auditoría, ya que delimita y establece hasta dónde debe llegar y, por lo tanto, qué elementos o aspectos quedan fuera.

Uno de los factores que añaden complejidad a la hora de definir el alcance de la auditoria de un dispositivo IoT, es que no se debe tener en cuenta únicamente el dispositivo, sino también la infraestructura adicional, las conexiones o la forma de manejar el dato que se transmite.

Revisando la guía de consideraciones de riesgo y valor de ISACA [17], se proponen una serie de preguntas clave, a responder, para que el alcance sea completo en el entorno de los dispositivos IoT:

- ¿Cómo se utiliza el dispositivo desde una perspectiva empresarial?
- ¿Cuáles son los vectores de amenaza para el dispositivo?
- ¿Existen amenazas conocidas y se sabe cómo mitigarlas?
- ¿Quién tiene acceso al dispositivo y cómo se comprobará la identidad?
- ¿Cómo se actualizará el dispositivo? ¿Existe un proceso definido?
- ¿Quién es el responsable de la monitorización frente a nuevos ataques?
- ¿Qué información personal es recopilada?
- ¿Cómo se almacena o procesa la información?
- ¿Se compartirán o divulgarán los datos?

El responder estas preguntas dotará al auditor de una perspectiva completa, para definir un alcance y definir un proceso.

Procedimiento de la auditoría

Para poder trazar un procedimiento, el auditor debe poseer suficiente información para identificar y seleccionar el enfoque adecuado, de esta forma será capaz de definir los pasos a seguir.

Hemos de señalar que no existe un estándar universal a la hora de realizar una auditoría IoT, pero ISACA ofrece un plan basado en 4 controles [15]:

- Controles de línea de base generales: Controles mínimos que deben aplicarse a todos los aspectos de la tecnología.
- Controles relacionados con los datos: Como los controles que se aplican a los datos que forman una parte clave de IoT.
- Análisis y controles relacionados con el aprendizaje: Se aplica para garantizar que el análisis sea ético y permita el uso confiable de los datos y que los resultados del análisis se puedan aplicar a la toma de decisiones empresariales.
- Alineación de negocios y procesos: La evolución y las características de los dispositivos IoT deben aporten valor al negocio y acompañar su implementación con las tendencias y necesidades del mercado.

5. Capítulo Riesgos y medidas de protección y mitigación para dispositivos IoT

Este capítulo abarca los principales riesgos y medidas de protección/mitigación que se encuentran actualmente en los dispositivos IoT.

Con la creciente comercialización de los dispositivos IoT, y bajo la premisa de que ya desde sus inicios primaba el diseño por encima de la seguridad. Se encontraron muchas vulnerabilidades que no solo ponían en riesgo el dispositivo, si no, que también ponían en riesgo la infraestructura a la cual estaban conectados, y los datos que procesaban o almacenaban.

Basándose en esta problemática, múltiples organizaciones comenzaron a publicar sus recomendaciones, análisis y estudios sobre cuáles eran los principales riesgos y en que vulnerabilidades estaban basados.

En este capítulo se revisan dos de estas organizaciones, OWASP que está focalizada en las vulnerabilidades conocidas en los dispositivos y la GSMA que está enfocada en las recomendaciones y mejores prácticas para el diseño correcto de los dispositivos.

OWASP IoT Security Guidance

OWASP es una fundación que trabaja para la mejora de la seguridad del software, a través de una comunidad de código abierto compuesta por miles de miembros y organizaciones.

La repercusión es tan alta que, en la actualidad, son el máximo referente a nivel de estándares sobre vulnerabilidades. Prácticamente toda organización, con una presencia relevante en el sector, basa sus procedimientos teniendo en cuenta las recomendaciones de OWASP.

OWASP es conocido por su listado, que se actualiza cada 2 o 3 años, mostrando cuales son las principales vulnerabilidades encontradas a nivel de desarrollo, cuál es el impacto y el riesgo. Pero, dada la creciente demanda de dispositivos IoT, se han tenido

que adaptar para poder realizar estas recomendaciones para el mundo estos dispositivos.

La última publicación por parte de la comunidad OWASP para IoT fue en 2018 con su Top 10 [18] (véase Tabla 4), con las siguientes recomendaciones en función del nivel de riesgo que representan:

Nivel	Riesgo	Descripción
L1	Contraseñas débiles, fácilmente adivinables o harcodeadas	Uso de credenciales que se obtenga fácilmente por fuerza bruta Credenciales públicas Puertas traseras en el firmware o software que otorga acceso no autorizado.
L2	Servicios de red inseguros	Servicios de red inseguros o corriendo en el propio dispositivo y expuestos a internet Servicios que comprometan la confidencialidad, integridad, disponibilidad o permitan el control remoto no autorizado.
L3	Ecosistemas con interfaces inseguras	Webs inseguras, API con acceso a backend, interfaces móviles en dispositivos que permitan comprometer el mismo o sus componentes. Los problemas comunes incluyen una falta de autenticación/ autorización, falta de cifrado y/o un cifrado muy débil.
L4	Falta de mecanismos de actualización seguros	Falta de capacidad para actualizar el dispositivo. Esto incluye la falta de entrega segura (sin cifrado en tránsito), falta de mecanismos anti retroceso y/o falta de notificaciones de cambios de seguridad.
L5	Uso de componentes inseguros u obsoletos	Uso de componentes/ bibliotecas de software obsoletas. Esto incluye la personalización de los sistemas operativos y el uso de software o hardware de terceros en una cadena comprometida.
L6	Insuficiente protección de la privacidad	La información personal del usuario almacenada en el dispositivo o en el ecosistema se usa de forma incorrecta o sin autorización
L7	Transferencia y almacenamientos de datos inseguros	Falta de encriptación o control de acceso de datos confidenciales en cualquier parte del ecosistema, tanto en tránsito como en reposo.
L8	Falta de gestión de los dispositivos	Falta de soporte en el ámbito de la seguridad en los dispositivos implementados en producción, incluida la gestión de activos, gestión de actualizaciones, desmantelamiento seguro y/o monitorización del sistema.
L9	Configuración por defecto	Los dispositivos o sistemas son enviados con configuraciones predeterminadas de forma insegura, o carecen de la capacidad para poder ser seguros, al restringir las modificaciones de configuración por parte de los operarios.
L10	Falta de hardening físico	Falta de medidas de seguridad física, lo que le permite a un atacante planear potenciales ataques para obtener información sensible que pueda ayudar en un futuro ataque remoto o tomar el control local del dispositivo.

Tabla 4 - Top 10 OWASP IoT 2018 (Elaboración propia a partir de datos: OWASP) [18]

OWASP dispone de más recomendaciones para abordar la problemática del IoT:

- OWASP IoT Top 10 Mapping Project
- OWASP IoTGoat
- OWASP Firmware Analysis Project
- OWASP Firmware Security Testing Methodology
- IoT Security Verification Standard

Todas creadas para las buenas prácticas en el desarrollo e implementación de dispositivos. A su vez, OWASP dispone de múltiples herramientas de código abierto para realizar auditorías y comprobaciones sobre diferentes elementos del ecosistema IoT.

GSMA IoT Security

GSMA (Global System for Mobile Communications) es una organización formada en 1995, a nivel mundial, que defiende los intereses de los operadores de redes móviles.

La organización cuenta con más de 750 operadores móviles y 400 empresas del ámbito de las telecomunicaciones [19].

La GSMA representa a sus miembros a través de programas industriales a gran escala, grupos de trabajo e iniciativas para defender y posicionar a la industria.

Dado el alto impacto que han tenido los dispositivos IoT, la GSMA se vio obligada a lanzar un programa para ayudar a los operadores a aportar valor y acelerar la entrega de nuevos dispositivos y servicios conectados. Para poder alcanzar estos objetivos la GSMA se ha basado en:

- Colaboración de la industria.
- Regulaciones adecuadas.
- Optimización de las redes.
- Desarrollo de habilidades clave para el crecimiento a largo plazo.

LA GSMA abarca muchos temas de forma simultánea, desde ciudades inteligentes, vehículos conectados o cielos conectados mediante drones, pero es a nivel de seguridad donde buscan un estándar común con el que puedan trabajar todos los fabricantes, vendedores e integradores.

La organización considera que, sin seguridad, el IoT dejara de existir, tanto por la falta de confianza de los consumidores, como por la vulnerabilidad de algunos servicios críticos. Por lo que la GSMA junto con la industria ha ideado una gama de recursos para que las empresas asuman la responsabilidad de abordar la seguridad desde el diseño y en cada etapa del ciclo de vida del IoT.

GSMA: Guidelines y Assessment

Se busca promover la mejor directriz y forma de evaluar la seguridad para ayudar a crear un mercado seguro con servicios confiables y escalables [20].

Las pautas de seguridad de la GSMA:

- Incluye 85 recomendaciones detalladas para el diseño, desarrollo e implementación segura de servicios IoT.
- Cubre redes, así como servicios y ecosistemas de punto final.
- Aborda desafíos de seguridad, modelos de ataque y evaluaciones de riesgos.
- Proporciona varios ejemplos trabajados como modelos.

La evaluación de seguridad de la GSMA:

- Se basa en un enfoque estructurado y controles de seguridad concisos.
- Cubre todo el ecosistema.
- Puede encajar en un modelo de cadena de suministro.
- Proporciona un marco flexible que aborda la diversidad del mercado de IoT.

Para poder conseguir todo esto, existen diferentes pautas plasmadas en documentos que abarcan los siguientes aspectos:

- Documento de descripción general.
- Ecosistemas y servicios.
- Ecosistemas de punto final.
- Operadores de red.
- Conjunto completo de documentos.
- Evaluación de seguridad IoT.

6. Capítulo Caso de uso práctico: Pentest Smartlock

En este capítulo se realiza una auditoría de un candado inteligente con tecnología BLE.

La auditoría está basada en un laboratorio pentesting IoT para dispositivos con tecnología BLE de la empresa Atiffy, fundada en 2013 [2].

Para la realización del laboratorio es necesario disponer de un candado inteligente obtenido en aliexpress [21] y de la aplicación OKLOK [22] instalada en un dispositivo Android, (véase Ilustración 8).

Además de un conjunto de herramientas que nos permitirán interactuar con el candado y la aplicación basadas en tecnología BLE.

Ilustración 8 – Aplicación y candado OKLOK

El objetivo es revisar los diferentes procedimientos y técnicas que se pueden utilizar para realizar el "hackeo" del dispositivo. Revisando cuáles son los posibles vectores de ataque, como pueden ser la comunicación entre la aplicación y el candado, o la propia aplicación.

Durante el capítulo, se revisan diferentes partes del ecosistema con el objetivo de encontrar debilidades que nos puedan proporcionar información para realizar la explotación del candado.

Análisis preliminar del dispositivo

El candando dispone de una aplicación para poder interactuar con él, desde la aplicación es posible conectarse al candando mediante el uso de BLE y a su vez puedes registrar la huella dactilar, ya que el candado contiene un lector de huella en su superficie para poder realizar su apertura mediante acceso físico.

Desde la aplicación también se puede controlar el estado del candando, se puede consultar el nivel de batería, si está cerrado o abierto y consultar un histórico de actividad.

Para este laboratorio de Pentesting se utiliza una máquina virtual Ubuntu capaz de interactuar con los periféricos físicos del sistema operativo anfitrión, de esta forma, se puede utilizar el bluetooth del ordenador y también el programa Frida [23] que permite interactuar mediante línea de comandos con el dispositivo móvil Android en el cual está instalada la aplicación OKLOK que controla e candado.

A la hora de realizar un pentesting de un dispositivo IoT, se puede abordar desde un punto de vista físico (hardware) o desde un punto de vista lógico (Software). Para la realización de esta auditoria se plantea un escenario lógico, ya que en internet se pueden encontrar auditorias sobre la parte física del dispositivo.

Dado que está basado en tecnología BLE, se van a utilizar 2 herramientas, Hcitool y Gatttool, para descubrir cuáles son las características y servicios que ofrece el candado:

Hcitool [24]: Es una herramienta CLI (Interfaz de la línea de comandos) permitiendo a los usuarios dar instrucciones a algún programa informático por medio de una línea de texto simple.

La herramienta Hcitool es capaz de detectar y conectarse a dispositivos periféricos, pudiendo enviar comandos para interactuar con ellos. Este se comunica

mediante un puerto HCI, que significa Host Controller Interface, permitiendo la conexión con dispositivos Bluetooth estándar y Bluetooth low energy.

Hcitool permite realizar acciones, como visualizar por pantalla el nombre, la versión y las funciones del dispositivo. A su vez, permite realizar otras acciones más complejas, como mostrar la intensidad de la señal recibida para la conexión establecida, el nivel de potencia de la transmisión o cambiar el rol de la conexión del dispositivo remoto a maestro o esclavo.

Gatttool [25]: Es una herramienta CLI utilizada para acceder a los servicios que se ejecutan en un dispositivo Bluetooth. GATT significa Atributo Genérico y define una estructura de datos para organizar características y atributos de los dispositivos.

Ambas herramientas vienen dentro del paquete Bluez, conocido como el pilar fundamental a la hora de trabajar con Bluetooth, además está integrada con el núcleo Linux por defecto, lo que permite que el despliegue de aplicaciones basadas en blueZ sea más sencillo.

Para comenzar a interactuar con el candado se debe comprobar que la interfaz Bluetooth del sistema operativo esté levantada, esto se realiza mediante el comando **hciconfig**, que nos permite listar las interfaces físicas del ordenador en el sistema operativo Ubuntu virtualizado. Tal y como se muestra en la (Ilustración 9), lista la interfaz hci0, con su dirección física y el status UP RUNNING (corriendo).

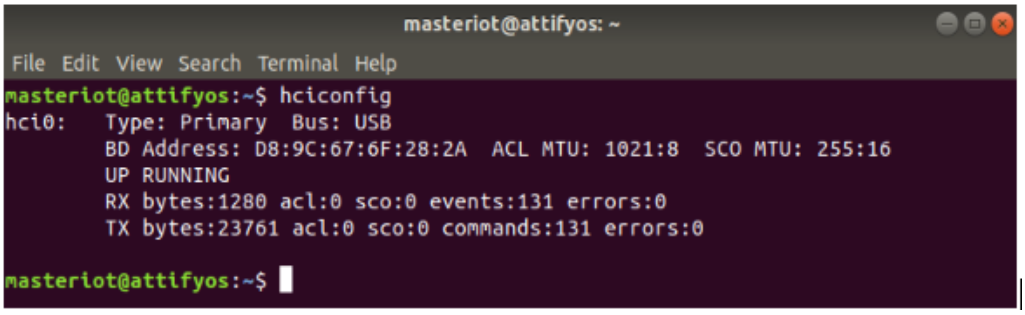

Ilustración 9 - Resultado hciconfig

NOTA: Si se utiliza una máquina virtual es necesario habilitar en la configuración de la misma que el SO nativo comparta la interfaz Bluetooth con el sistema operativo virtual.

Lo principal es descubrir cuál es la dirección física del smart lock, dado que conociendo la dirección física podremos conectarnos a él.

Lanzando el comando **hcitool lescan**, (véase Ilustración 10). Este comando realiza un escaneo de los dispositivos que se encuentran al alcance, donde se puede observar su dirección física y su nombre en caso de ser publicado.

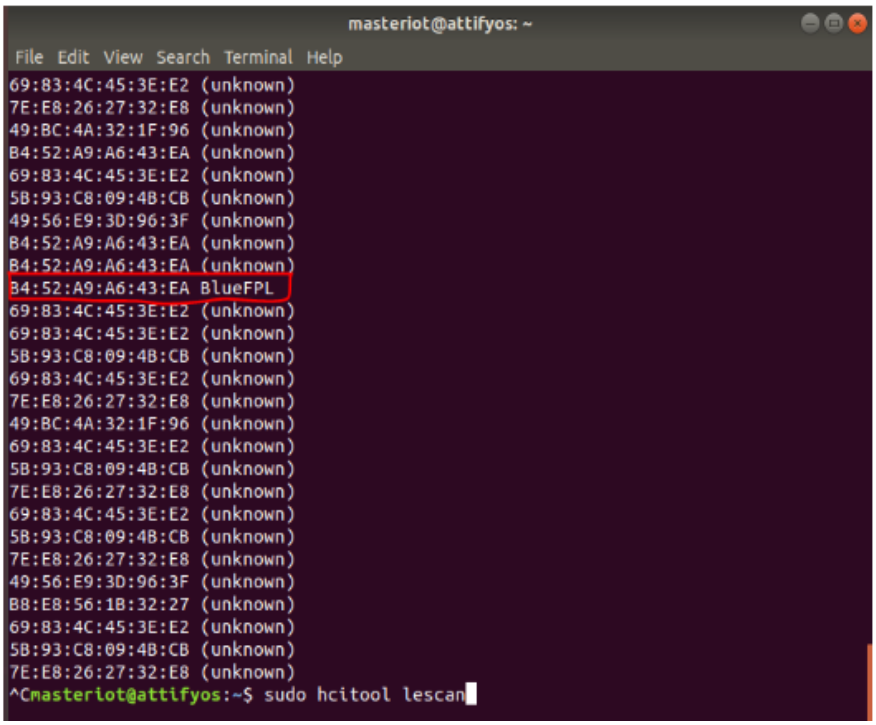

Ilustración 10 - Resultado hcitool lescan

Se muestra la dirección física del candando **B4:52:A9:A6:43:EA** y muestra el id name como BlueFPL. Esto también podría hacerse realizando una lectura del handle 0x9 que ofrece las características de nombre del dispositivo.

Para poder conocer más información acerca del dispositivo, se debe ejecutar el comando **hcitool leinfo,** (véase Ilustración 11). Para obtener esta información del dispositivo no es necesario estar vinculado, se obtiene a partir de los servicios de descubrimiento que ofrecen los dispositivos.

El comando muestra información, como el Handle y Features, resultando esto de suma importancia, ya que todo lo relacionado con los dispositivos Bluetooth se basa en esta información.

Handle: Es un identificador único, formado por 16 bits que establece la relación con cada servicio de la tabla GATT, de forma que no cambiará cuando se realicen transacciones o cuando estén enlazados con otros dispositivos. El número de identificadores disponibles para un servidor GATT son 0xFFF (65535), aunque normalmente los dispositivos suelen contener una media de 20 o 30.

Features: Los Gatt features son procedimientos estrictamente definidos para cada dispositivo, que permiten realizar intercambios de datos basándose en las diferentes operaciones que proporciona ATT. [26]

```
masteriot@attifyos:~$ sudo hcitool leinfo B4:52:A9:A6:43:EA
Requesting information ...
        Handle: 15658 (0x3d2a)
        Features: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
masteriot@attifyos:~$
```

Ilustración 11 - Resultado hcitool leinfo

Para interactuar con el dispositivo se utiliza la herramienta Gatttool, que nos permite acceder a los servicios y características de los dispositivos.

Una vez se realiza el emparejamiento al dispositivo sin necesidad de clave, se ejecuta el comando Primary, (véase Ilustración 12), donde se muestran cuáles son los servicios principales que ofrece el dispositivo. Estos servicios engloban a su vez las características, por ejemplo, el servicio "consulta de estado candado". Dicho servicio incluye las características del mismo, como estado de la batería o candado bloqueado.

```
masteriot@attifyos:~$ sudo gatttool -i hci0 -b B4:52:A9:A6:43:EA -I
[B4:52:A9:A6:43:EA][LE]> connect
Attempting to connect to B4:52:A9:A6:43:EA
Connection successful
[B4:52:A9:A6:43:EA][LE]> primary
attr handle: 0x0001, end grp handle: 0x0008 uuid: 0000fee7-0000-1000-8000-00805
f9b34fb
attr handle: 0x0009, end grp handle: 0x0013 uuid: 00001800-0000-1000-8000-00805
f9b34fb
attr handle: 0x0014, end grp handle: 0x0017 uuid: 00001801-0000-1000-8000-00805
f9b34fb
attr handle: 0x0018, end grp handle: 0xffff uuid: f000ffc0-0451-4000-b000-00000
0000000
[B4:52:A9:A6:43:EA][LE]>
```

Ilustración 12 - Resultado conexión gattool y comando primary

El comando *Characteristics* (véase Ilustración 13) permite ver las características del dispositivo, con sus respectivos identificadores únicos (UUID) que se explicaran en detalle en el apartado Desbloqueando el candado. Del mismo modo que puede identificar un dispositivo por su *device name*, también podría identificarlo por el UUID de su servicio anunciado, ya que esta información suele venir en la documentación del dispositivo.

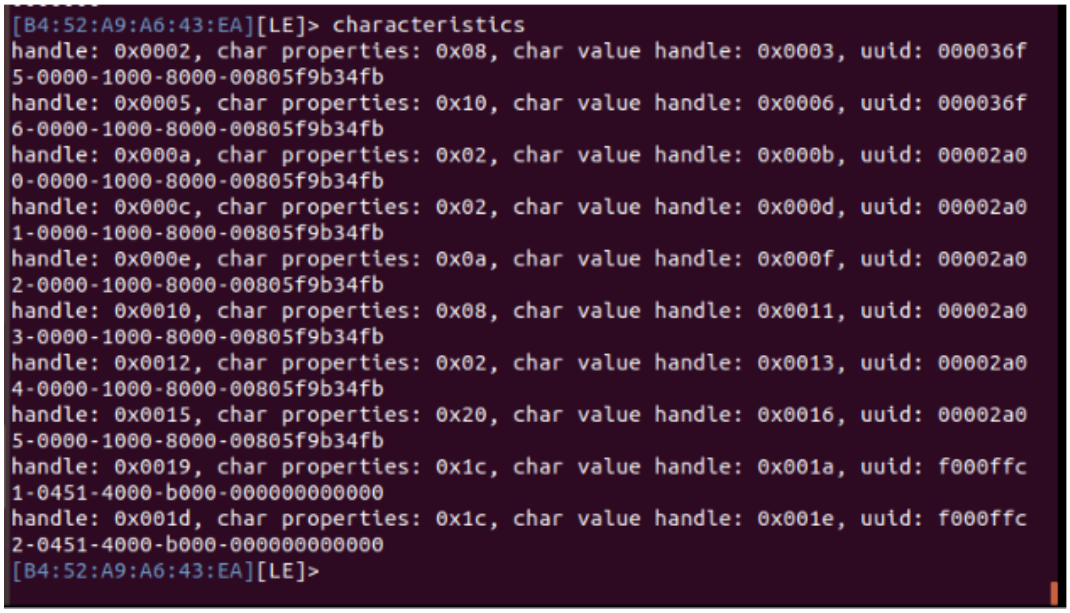

Ilustración 13 - Resultado Characteristics

Los dispositivos IoT basados en BLE están tan extendidos que se puede obtener la misma información que con las herramientas Gatttool y Hcitool. No siendo necesario para ello disponer de una maquina Linux, únicamente con aplicaciones móviles que se pueden encontrar en el market play de Google como **GATTBrowser** [27]. Se trata de una aplicación capaz de conectarse a un dispositivo BLE e interactuar con su tabla GATT, ofreciendo así una interfaz con una experiencia de usuario más amigable que la línea de comandos de Linux.

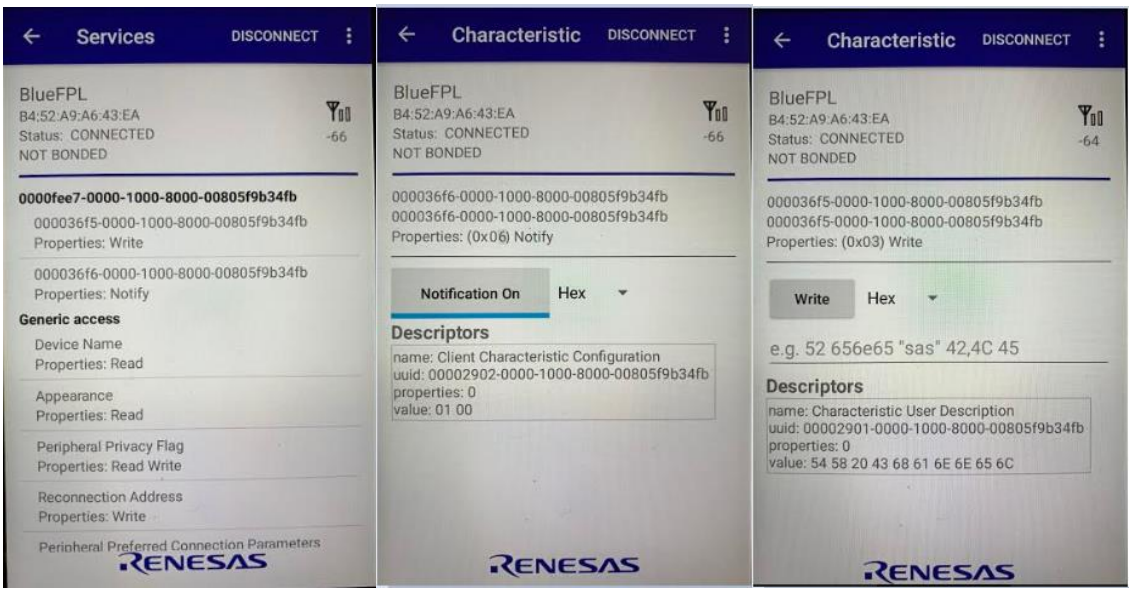

Ilustración 14 - Aplicación GATTBrowser

En la Ilustración 14 se muestra el resultado de lo que sucede al conectarse al candado BlueFPL y su tabla GATT, con los servicios principales que cuelgan del grupo con uuid 0000fee7. Estos servicios contienen dos atributos, uno de escritura y otro de notificación. La ilustración central y la de la derecha muestran cómo se puede interactuar con el dispositivo, activándose las notificaciones o escribiendo directamente en formato hexadecimal.

Esta herramienta nos permite interactuar con el dispositivo sin necesidad de conocer los comandos necesarios para la consola Linux, además nos proporciona mucha información relevante sin tener que navegar hasta la misma como ocurre con las herramientas de líneas de comandos.

Análisis del tráfico BLE

Una vez se ha realizado un reconocimiento a alto nivel del dispositivo, se debe identificar cuáles son los diferentes vectores de ataque.

Uno de los posibles vectores de ataque, en un ecosistema como este, es el tráfico entre la aplicación y el candado. Realizaremos un análisis del tráfico para intentar entender cuáles son las órdenes que recibe.

Para realizar este análisis del tráfico, existen 2 opciones:

1. Análisis del trafico BLE del dispositivo móvil.

2. Sniffing de tráfico mediante un *dongle* BLE como por ejemplo Ubertooth.

La complejidad de esta opción reside en que adicionalmente se deben descifrar las tramas, pero el objetivo del proyecto no es conseguir descifrar las tramas del protocolo BLE, si no tomar el control del candando [28].

En este caso se va a optar por la opción 1, obtener el tráfico del dispositivo Android y posteriormente analizar dicho tráfico con la herramienta Wireshark [29] para intentar entender cuáles son las ordenes enviadas desde la aplicación OKLOK al candado, con la intención de conocer cuáles y como son las ordenes que envía la aplicación para interactuar con el candado y mandarle una acción de apertura.

La Herramienta Wireshark es un analizador de protocolos de red, que permite, entre muchas otras características, la inspección profunda de protocolos y capturar el tráfico en tiempo real. Es compatible con la mayoría de sistemas operativos y permite soporte de descifrado para protocolos como Kerberos, TLS, WEP, WPA, WPA2. [29]. A su vez permite capturar una orden para posteriormente realizar un ataque de repetición con la herramienta Gatttool (Replay Attack).

Para obtener el tráfico, se debe habilitar, en las opciones del sistema operativo Android, el modo desarrollador: **Ajustes → Acerca del teléfono → Pulsar 7 veces en Numero de compilación**. De esta forma en el menú ajustes aparece una opción llamada "**Opciones de desarrollo"**. Dentro de ese menú se debe activar el switch llamado **Registro búsqueda HCI Bluetooth.**

Nota: Para dispositivos iphone se puede activar el perfil "Bluetooth Logging for iOS" e instalarlo siguiendo los pasos que se encuentran en la documentación para google developers [30]

Cuando se activa el modo de registro HCI Bluetooth, se crea un fichero en el dispositivo llamado "btsnoop_hci", donde se encuentran las trazas con la información de los paquetes compartidos entre el candado y el aplicativo.

Una vez se dispone del fichero con la información, este puede ser analizado con la herramienta Wireshark, utilizando el siguiente filtro, (véase Ilustración 15).

btatt.opcode== "Write Request" && btatt.handle==3

El filtro está formado por el atributo btatt (Bluetooth Attribute Protocol) acompañado del opcode, que en este caso representa el tipo de petición Write Request. A su vez, el comando va concatenado con btatt.handle==3 que especifica el atributo del dispositivo. Esta información se obtiene Attify especifica qué *handle* se debe revisar para escribir ordenes al candado. Esta información podría obtenerse con la documentación técnica de Bluetooth que se puede encontrar en su página web [31].

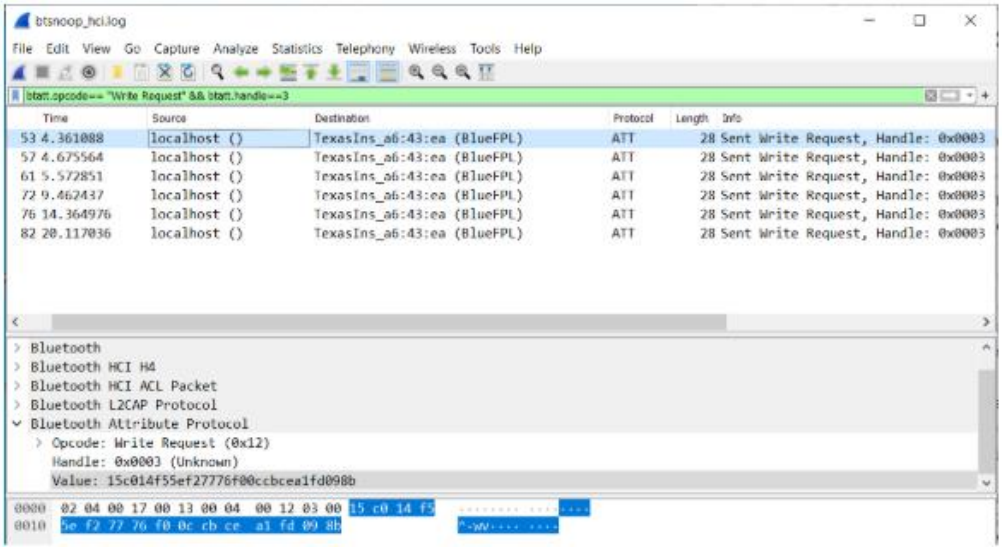

Ilustración 15 - Resultado Wireshark handle 0x03

btatt.opcode== "Handle Value Notification" && btatt.handle==6

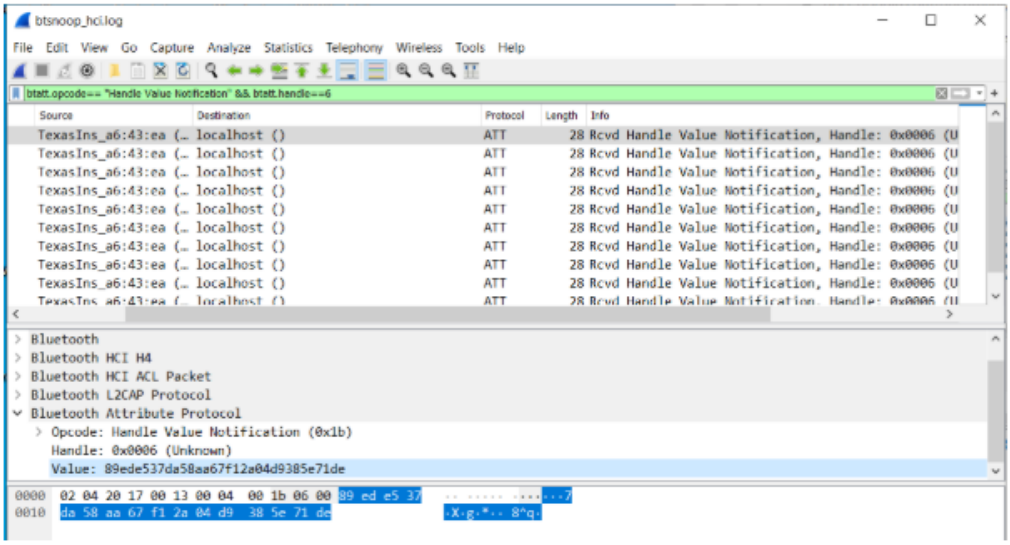

Ilustración 16 - Resultado Wireshark handle 0x06

En la Ilustración 14 se muestra que el servicio con UUID 0xfee7 tiene dos características: 0x36f5 y 0x36f6 con *handle* 0x3 y 0x6, las propiedades correspondientes son WRITE y NOTIFY como se puede visualizar en la Ilustración 14.

Al abrir el archivo de registro *hci snoop* en Wireshark y aplicar un filtro (btatt. Opcode == "Write Request" && btatt. Handle == 3) únicamente se muestran los paquetes Write al *handle* 0x3.

Se observa exactamente lo mismo para el controlado 0x6 en la Ilustración 16, de notificaciones si modificamos el filtro.

En ambos casos se visualiza únicamente que el *payload* contiene una información que no es reconocible, es decir, se podría entender que el valor enviado desde la aplicación al dispositivo y viceversa están cifrados, aun así, se puede probar a replicar un ataque enviado el mismo *payload* capturado en el *handle* 0x03 con la intención de reenviar la misma orden para intentar abrir el candando.

Este ataque es conocido como Reply Attack, lo que se realiza es un intento de escribir ese *payload* capturado en el atributo 0x03 que hace referencia a un servicio con la intención de replicar la orden.

Con la herramienta Gatttool se debe enviar el siguiente comando que realiza una escritura:

char-write-req 0x003 15c014f55ef27776f00ccbcea1fd098b

El resultado que se muestra por pantalla es un "*success*", pero el candado no realiza ninguna acción. Esto quiere decir, que se interactúa con el candado, y que el valor enviado es válido, pero no es correcto o no se ha replicado una orden de apertura del candado.

El problema surge cuando no se conocen los atributos del dispositivo que se deben inspeccionar. Tal y como como se muestra en la Ilustración 13, existen una docena de características y el coste de tiempo para realizar las pruebas sobre un dispositivo sería muy elevado, por lo que la revisión de las trazas y el intento de replicar un ataque de repetición con cada una de ellas supondrían un enorme coste.

En consecuencia, debemos entender cómo se comunican ambos dispositivos, si utilizan algún tipo de cifrado y cómo se negocia entre la aplicación y el candado.

Descifrando el funcionamiento

Como se explica en el apartado anterior, aunque se capture el tráfico y se ejecute un ataque de repetición al lanzar las mismas órdenes, esto no realiza la acción de apertura del candando. Suponiendo, en este caso, que se han replicado todas las ordenes capturadas y entre ellas estaba la orden de apertura.

Un buen vector de ataque en estos casos, es la aplicación OKLOK encargada de la gestión del dispositivo, ya que está disponible públicamente y se puede realizar una ingeniería inversa para entender cómo es su funcionamiento.

Para la realización de la ingeniería puede utilizarse la herramienta JadX [32] que permite, mediante línea de comandos y herramientas GUI (*Graphical User Interfaze*), descompilar el código de archivos tipo Dex y Apk, que son los archivos compilados de una aplicación Android. El archivo APK es la aplicación en sí y el archivo Dex consta de un código optimizado encargado de acelerar el proceso de carga de la aplicación, y convertirlos a lenguaje Java, además de otras funciones como resaltar las sintaxis, realizar una navegación entre clases.

Con esta funcionalidad se puede navegar por él código e intentar comprender como opera la aplicación a partir de sus clases y métodos.

Tras realizar las pruebas de captura de tráfico en varias ocasiones se observa que mandando las mismas órdenes desde la aplicación OKLOK, en la orden de apertura del candado el *payload* es distinto. Esto sugiere que el *payload* está formado por algún dato aleatorio, identificador de sesión, etc.

En la Ilustración 17 se muestra el código fuente de la Apk de Android como si se tratara de código java tras la descompilación usando JadX .

Una vez se dispone de toda la estructura de la aplicación proporcionada por JadX, se debe realizar una ingeniería inversa y entender cómo está estructurada la aplicación, para ello una de las herramientas más útiles que proporciona JadX es el buscador. El buscador permite realizar búsquedas por palabras, como key, AES, Cipher, token, que nos pueden ayudar a encontrar métodos de cifrado/descifrado.

Es de suma importancia entender la cantidad de clases y métodos que contiene una aplicación, junto con todas sus librerías y dependencias. Es por ello que realizar esta ingeniería inversa de forma manual, y además sin disponer de la estructura de Java, supondría una tarea muy compleja.

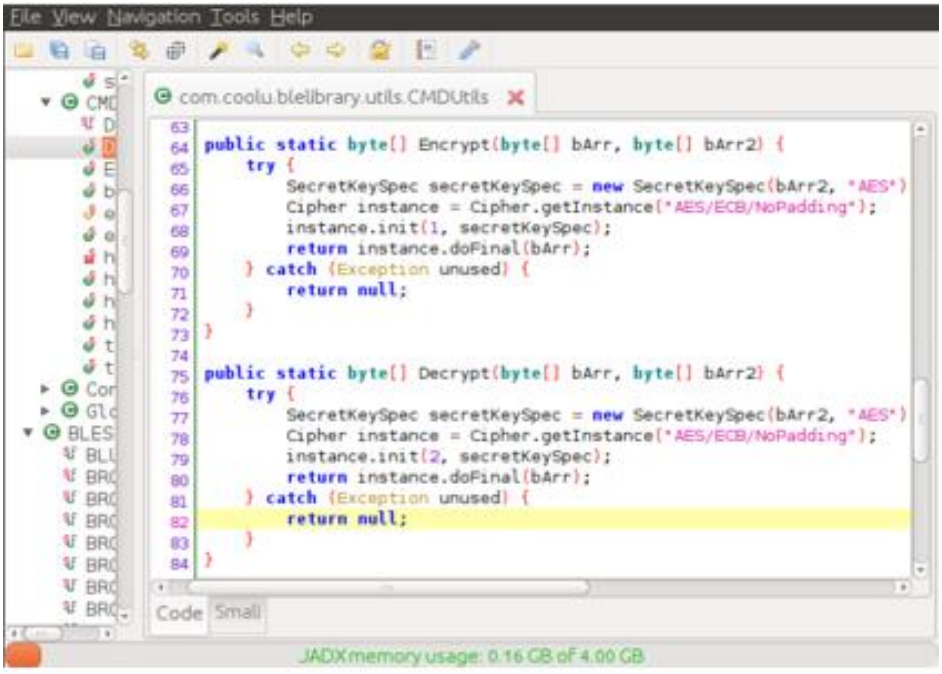

Ilustración 17 - Resultado descompilación JADX

Una vez con el código descompilado y tras la búsqueda de las cadenas Encrypt y Decrypt que se encuentran en la clase **CMDUtils** dentro del paquete **com.coolu.blelibrary.utils,** (Ver Ilustración 17), dos métodos que son interesantes desde el punto de vista del análisis de seguridad.

La aplicación muestra 2 funciones, Encrypt y Decrypt que toman dos conjuntos de bytes como parámetros. El primer parámetro contiene el texto sin formato y texto cifrado, la segunda contiene la clave.

Estas dos funciones, y las referencias a los algoritmos de cifrado, dan algunas pistas sobre el comportamiento de la aplicación respecto al candado.

Es importante revisar el método de cifrado para buscar si existe alguna vulnerabilidad conocida.

El modo de cifrado que utiliza el candado es ECB (Electronic Code Book), este se considera inseguro ya que proporciona una baja confidencialidad debido a que ECB produce la misma salida para la misma entrada cada vez. Si se hubiese realizado una auditoria de pentesting previa a la venta del candado y esta hubiese revisado el Top 10 de OWASP para IoT, se hubieran dado cuenta que corresponde a una vulnerabilidad de nivel 6 (Configuración incorrecta de la seguridad) [33].

Uno de los objetivos principales de los dispositivos IoT, es que puedan tener un consumo energético bajo para ganar la mayor autonomía posible, por lo que, la capacidad de cómputo suele ser un problema importante a la hora de realizar el diseño. Muchas veces este intento de ahorro de consumo penaliza a la parte de seguridad, ya que los algoritmos de cifrado consumen bastante procesamiento.

En la Ilustración 18 se muestra una comparativa del uso del hardware, utilizando el algoritmo que usa el candado AES ECB contra el AES CBC. Se puede observar claramente que, haciendo uso de claves de 128K, el rendimiento del CBC es más del doble en MB/s. A su vez, se considera el tipo CBC unos de los modelos más estables y robustos mientras que el AES ECB está totalmente desaconsejado.

Por lo que para este candando se recomienda el uso del algoritmo de cifrado CBC, aunque para la generación de claves robustas su consumo de hardware sea mayor.

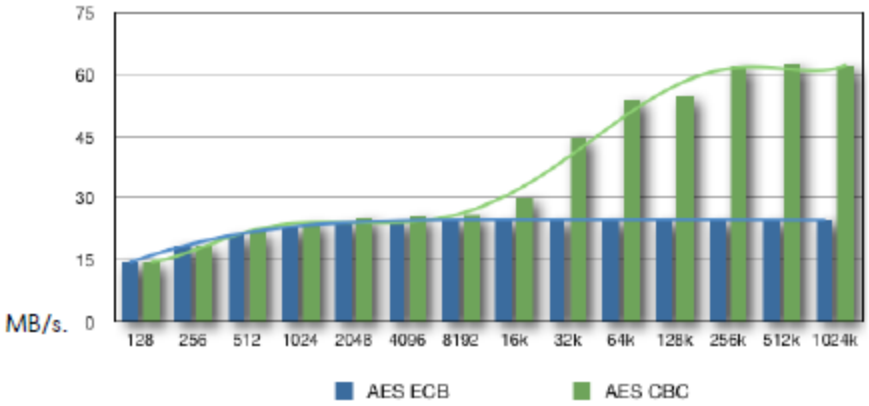

Ilustración 18 - Comparativa Consumo Hardware (origen: spouliot) [33]

Para obtener la clave de cifrado que manejan las funciones Encrypt y Decrypt sin necesidad de realizar una reingeniería completa de toda la aplicación, se utiliza la herramienta de instrumentación dinámica llamada **Frida** [23] que es capaz de conectarse a las funciones Encrypt y Decrypt de la aplicación OKLOK mientras la aplicación se está ejecutando en el dispositivo Android e imprime los valores de las variables.

Frida funciona en modo cliente-servidor, actuando como cliente la máquina virtual Linux y como servidor el dispositivo móvil que tendrá que estar conectado a la máquina virtual, es importante que el dispositivo móvil este rooteado para poder acceder a las características del sistema operativo Android como súper usuario y de esta forma Frida tenga permisos de lectura, escritura y ejecución sobre la aplicación OKLOK.

El núcleo de Frida está escrito en C e inyecta el motor V8 de Google [35] que es el encargado de traducir código Java Script a leguaje máquina para que pueda ser entendido por la CPU en los procesos de destino, donde el código JS de Frida se ejecuta con acceso completo a la memoria, pudiendo acceder de esta forma a las funciones.

El conjunto de herramientas que nos ofrece Frida tiene 4 características:

- **Programable**: Es posible inyectar scripts propios para realizar procesos de caja negra, hookear cualquier función, espiar APIs criptográficas o rastrear el código de aplicación sin necesidad de tener el código fuente. Se puede editar, guardar y ver los resultados al instante, sin compilaciones o reinicios.
- **Portable**: Puede inyectarse en procesos de ejecución de Windows, MacOS, GNU/Linux, iOS, Android, y QNX. Permite además la interacción con Node.js, Python, Swift, .Net, Qt/Qml y C.
- **Gratuita**: Ha sido desarrollada como software libre para que la comunidad pueda contribuir y de esta forma mejorar la herramienta.
- **Confiable**: Tiene el respaldo de importantes empresas, y una suite de pruebas completa.

Para poder utilizar Frida con un dispositivo Android que no esté rooteado, y por lo tanto no tengamos permisos sobre el esquema del SO, será necesario parchear la aplicación OKLOK volviendo a reconstruirla añadiéndole un plugin de Frida en su interior que posteriormente nos permita conectar el Frida del ordenador con el Frida del móvil conectado, de esta forma podremos acceder a la aplicación en tiempo de ejecución. Para realizar el parcheado de la aplicación OKLOK se puede utilizar la herramienta Objection [36], esta herramienta fue creada por los creadores de Frida.

Es muy importante construir la apk con el plugin de Frida para la arquitectura correcta del SO operativo Android, en este caso arm (véase Ilustración 19). Normalmente todas las arquitecturas de los dispositivos móviles suelen ser arm.

objection patchapk --source OKLOK_v1.3.0.apk --architecture armeabi

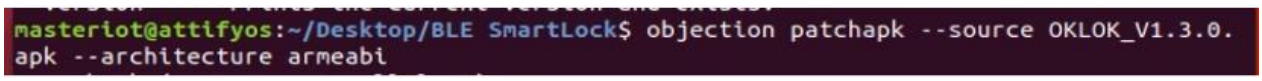

Ilustración 19 - Comando para crear apk con objection

Se debe instalar la nueva apk en el dispositivo, para esto se debe conectar el dispositivo móvil al ordenador mediante el cable USB, para posteriormente realizar la instalación desde la consola con el comando adb (véase Ilustración 20).

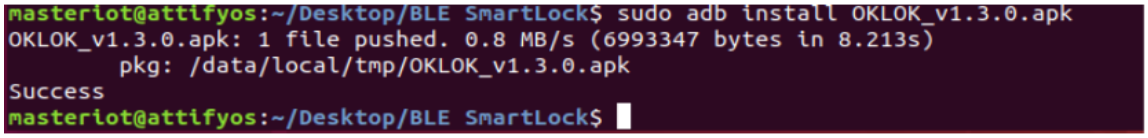

Ilustración 20 - Instalación APK con el comando adb

A su vez, se debe instalar el servidor Frida en el dispositivo móvil y darle permisos de ejecución, (véase Ilustración 21).

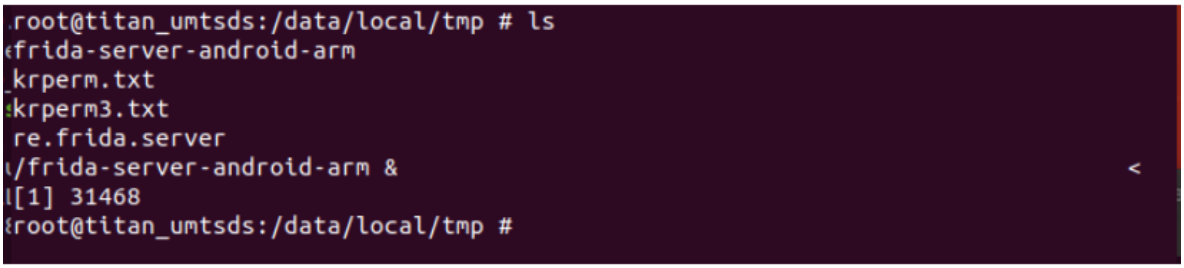

Ilustración 21 - Copia del servidor Frida al Android

Para comprobar que el servidor Frida está funcionando, se puede ejecutar **Frida-ps -U** para listar los procesos del dispositivo, (véase Ilustración 22).

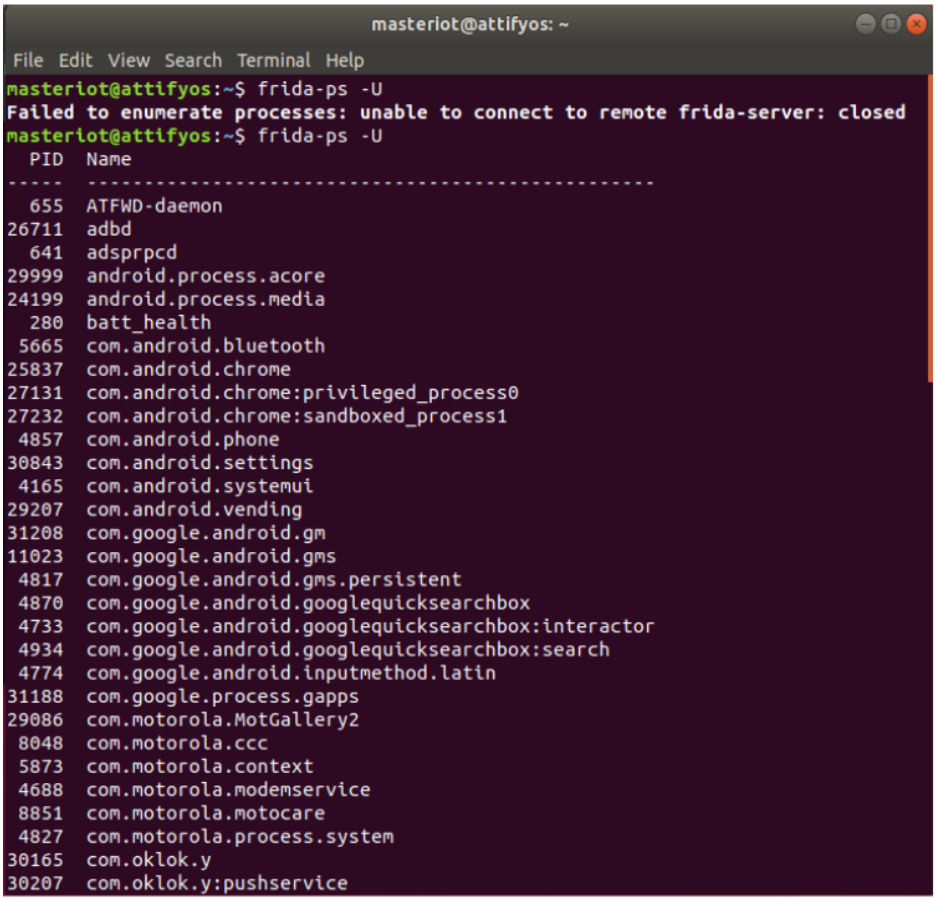

Ilustración 22 - Resultado comando Frida-ps -U

Con la aplicación parcheada e instalada, se puede ejecutar el oklok1.js script (véase Ilustración 23), este script lo proporciona la empresa Attify para poder realizar las pruebas. Este script se conecta a la clase Java CMDUtils y muestra por pantalla el payload y la key usada para el manejo de la información recibida y enviada al candado, utilizando las funciones localizadas con anterioridad a partir del JadX.

El script está adaptado a la versión 3 de la aplicación OKLOK, para trabajar con una versión actual de la aplicación, se debería de adaptar a la estructura de la nueva aplicación. El Script se conecta a las funciones de encriptado y desencriptado.

```
Java.perform(function ()
{
    var CMDUtils = Java.use('com.coolu.blelibrary.utils.CMDUtils');

    log_byte_array = function (arr) {
        var result = "";
        var buffer = Java.array('byte', arr);
        for(var i = 0; i < buffer.length; ++i)  {
            hexb = (buffer[i] & 0xFF).toString(16);
            if (hexb.length == 1) hexb = '0' + hexb;
            result += hexb;
        }
        console.log(result);
    };

    CMDUtils.Encrypt.implementation = function (pt, key) {
        console.log('[+] Inside Encrypt()  ======');
        var ct = this.Encrypt(pt, key);
        console.log('Pt:')
        log_byte_array(pt)
        console.log('key:')
        log_byte_array(key);
        return ct;
    };

    CMDUtils.Decrypt.implementation = function (ct, key) {
        console.log('[+] Inside Decrypt()  ======');
        var pt = this.Decrypt(ct, key);
        console.log('Pt:')
        log_byte_array(pt)
        console.log('Key:')
        log_byte_array(key);
        return pt;
    };
});
```

Ilustración 23 - Script para vincular a Frida

Una vez está la aplicación parcheada instalada en el Android, se procede a su ejecución y automáticamente se muestra como la aplicación se queda esperando, ya que el plugin de Frida que lleva en el interior está esperando a la ejecución del cliente Frida en la consola linux, (véase Ilustración 24).

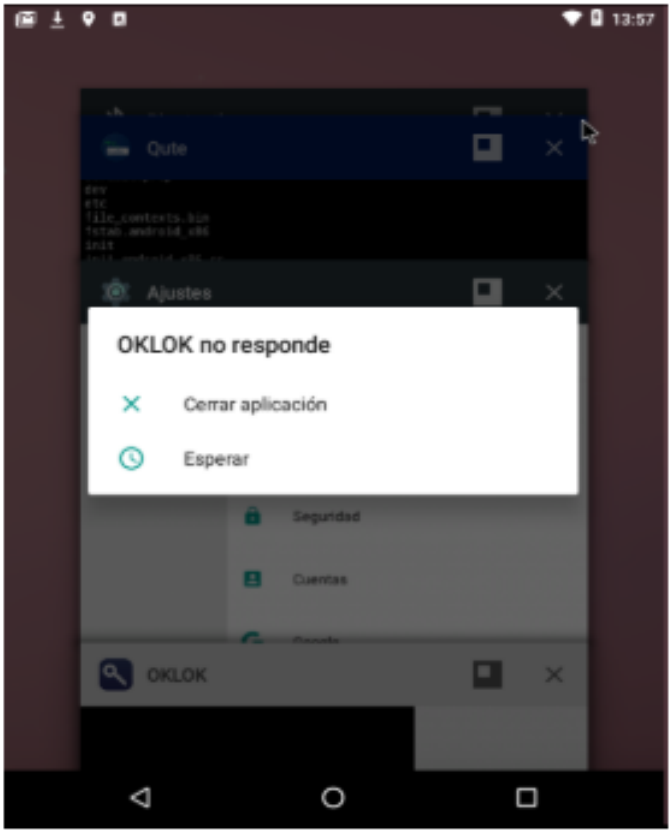

Ilustración 24 - Plugin Frida esperando

Mientras el aplicativo espera, se ejecuta el cliente Frida en la máquina virtual y se envían las órdenes desde la aplicación hacia el candado.

Como se menciona en el apartado de la reconstrucción de la apk con objection, la aplicación tiene que estar creada para la arquitectura correcta de Android, si no la aplicación no se ejecuta correctamente y no espera a interactuar con Frida, por lo que el sistema operativo Android devuelve un error indicando que la aplicación se ha cerrado de forma inesperada.

Desde la máquina virtual Ubuntu se puede ejecutar Frida pasando los parámetros U para el dispositivo conectado por USB (teléfono inteligente Android), -l para el script que se utilizará y finalmente el proceso al que se conecta.

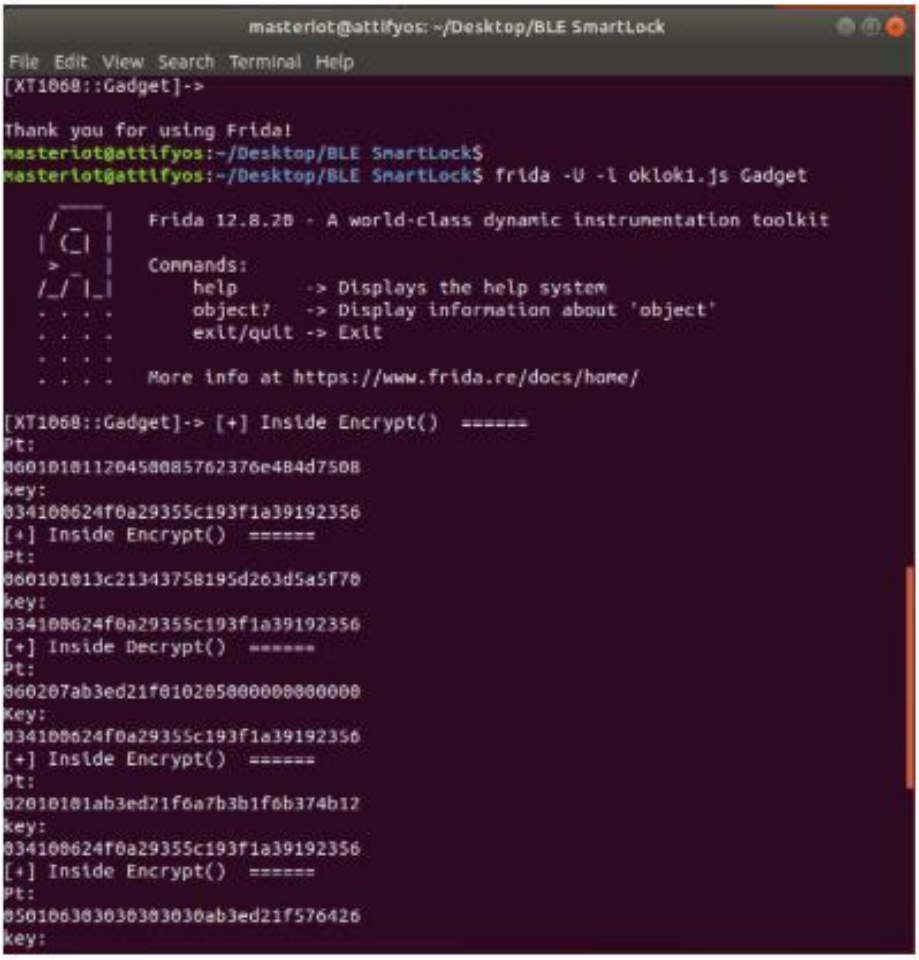

Ilustración 25 - Script Frida mostrando Key

En la Ilustración 25, se muestra como ejecutando la nueva aplicación OKLOK y enviándole ordenes al candado, Frida es capaz de visualizar cada orden que se intercambia entre el dispositivo y la aplicación y además conocer cuál es la clave de cifrado.

Mostrándose la clave AES con valor **034100624f0a29355c193f1a39192356**.

Sus 16 bytes de tamaño indican que la criptografía es AES-128. Con esta clave se puede descifrar el payload capturados en Wireshark. Usando el script oklok-descifrar. Py (véase Ilustración 26).

Si se utilizan las claves para descifrar los paquetes capturados anteriormente, se obtiene cada orden, para ello se utiliza el siguiente Script.

```
import sys
from Crypto.Cipher import AES

key= '034100624f0a29355c193f1a39192356'
#ct = sys.argv[1].decode('hex')
ct = 'fcabad014f285db655d3e0adcd0b32f6'.decode('hex')
aes = AES.new(key, AES.MODE_ECB)
print aes.decrypt(ct).encode('hex')
```

Ilustración 26 - Script desencriptar paquetes

Si se realiza un análisis de los payload obtenidos en cada orden, se muestran los valores de la Ilustración 27.

```
Pt:
060101015c264308225d09602c44752c
key:
034100624f0a29355c193f1a39192356
[+] Inside Encrypt()  ======
Pt:
0601010115767c055401744435175f79

key:
034100624f0a29355c193f1a39192356
[+] Inside Decrypt()  ======
Pt:
060207d81ac915010205000000000000
Key:
034100624f0a29355c193f1a39192356
[+] Inside Encrypt()  ======
Pt:
02010101d81ac91579694f552d0f2b00
key:
034100624f0a29355c193f1a39192356
[+] Inside Decrypt()  ======
Pt:
0202015e1ac915010205000000000000
Key:
034100624f0a29355c193f1a39192356
[+] Inside Encrypt()  ======
Pt:
050106303030303030d81ac915037b6e
key:
034100624f0a29355c193f1a39192356
```

Ilustración 27 - Frida en ejecución pintando las key

Revisando el primer payload desencriptado, este comienza con 0x0601, que es 1537 en decimal. Analizando el código descompilado con JADX se encuentra una

orden dentro de la de clase **com.coolu.Blelibrary** como se muestra en la Ilustración 28, que contiene la variable de tipo constante con el valor asignado. Es una clase con las constantes que referencian a cada orden, el valor 1537 representa a la constante **GET_ TOKEN**, que a su vez, es un tipo de comando.

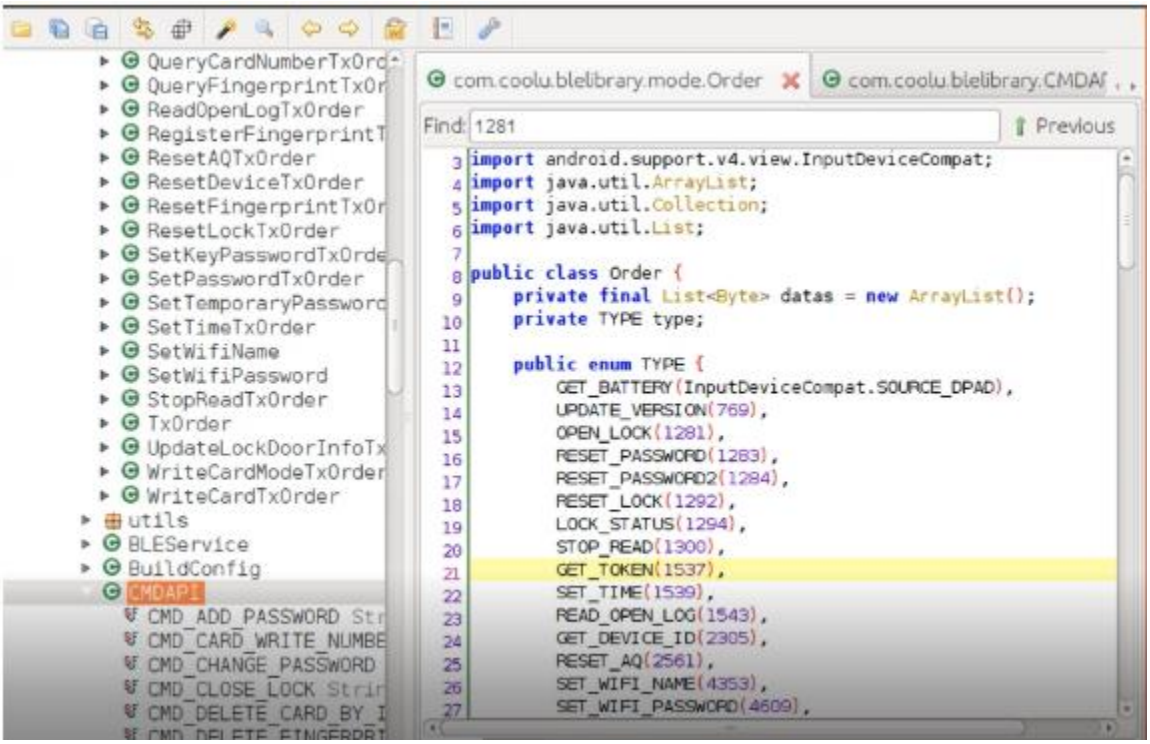

Ilustración 28 - Clase Order constantes

El segundo comando es 0x0201, que en decimal representa el valor 513, siendo utilizado para obtener la información de la batería. El tercer comando es 0x0501 o 1281 en decimal, representa la orden **OPEN_ LOCK**, la orden que parece diseñada para realizar la apertura del candado.

Cuando se capturó el tráfico BLE, también se capturaron los paquetes que vienen con origen handle 0x06, siendo estos de tipo notificación.

Estos paquetes realizan la función de notificar si una petición de escritura sobre alguno de los handle ha sido correcta o no. Es decir, utilizando el valor 0x0601 para escribir **GET_TOKEN** y el dispositivo utiliza el valor 0x0602 para notificar si ha obtenido o no correctamente, siendo un handle de notificación.

La información de las notificaciones se puede encontrar en la clase CMDAP dentro del blelibrary.

Desbloqueando el candado

Una vez se ha identificado como funciona el candado, y los distintos payload tanto de escritura como de notificación, que dichos payload están cifrados, la clave necesaria para descifrar los payload y en que parte del código esta la información que representa cada orden del candado, se puede crear un script para poder realizar la acción de apertura sin necesidad de la aplicación.

La aplicación envía un comando de escritura al handle 0x03 del dispositivo y el candado envía su respuesta en formato notificación a través del handle 0x6.

Si se vuelve al principio y se observan los valores obtenidos por Gatttool sobre las características del dispositivo, se descubre que el 0x03 tiene asociado un UUID especifico al igual que el handle 0x06 y el resto de características.

Esto se puede comprobar en la clase BLEService dentro de package.com.coolu.blelibrary mediante JADX.

Analizando los payload mostrados por Frida y sabiendo a que hace referencia cada orden ya que conocemos las constantes declaradas en la aplicación por JadX, se genera un diagrama completo (véase Ilustración 29) de cómo se realiza todo el flujo, pudiendo así crear un mecanismo automático que replique este comportamiento. Para ello la empresa Attify proporciona un script que realiza las mismas acciones (Todo el código utiliza la librería bluez para el manejo de dispositivos BLE).

Ilustración 29 - Flujograma conexión (elaboración propia)

La primera orden la realiza la aplicación, solicitándole obtener un token al candando, a partir de esta negociación, el token es utilizado como parte del payload en cada petición, es una forma de identificar que las ordenes enviadas desde la aplicación al candado y viceversa forman parte del emparejamiento anterior, ya que el token cambia cada nuevo emparejamiento.

Por lo tanto, para abrir el candado sin el uso de la aplicación, es necesario crear un código que realice todas estas acciones.

Todos los payload que componen las ordenes están formados por 16 bytes, donde los dos primeros bytes representan el tipo de orden. Para poder entender cómo construir dichas órdenes hay que revisar la clase Order de la aplicación OKLOK, en esa clase se encuentran las constantes de las órdenes y desde la misma navegar a la clase GetTokenTxOrder.

En la clase GetTokenTxOrder se encuentra una función llamada add que agrega 2 bytes de tipo 01 después de la Orden, en este caso seria **0601** que representa la orden y **0101** que son los 2 bytes que agregamos y el resto del buffer hasta completar los 12

bytes se rellena con valores aleatorios, estos datos son únicamente para completar, no importa cuales sean.

La respuesta a esa petición, que se enviará por el handle 0x06, contendrá el token de 4 bytes.

Se realiza el mismo relleno para las ordenes de escritura y notificación en la clase OpenlockTxOrder para realizar la orden de apertura del candado.

En este caso los 2 primeros bytes contienen la orden de tipo 0x0501 seguido de 06, Los siguientes 6 bytes contiene un parámetro fijo que se obtiene siempre con los valores 0x30 que van por defecto, los siguientes 4 bytes contienen el token obtenido, los siguientes bytes hasta completar los 16 se rellenan con valores aleatorios, en nuestro caso son 3 bytes a 0x00.

Quedando finalmente 0x0501063030303030**xxxxxxxx** como se muestra en la Ilustración 14, los bytes a 0 no se muestran.

Esta es la información necesaria para escribir un script que sea capaz de conectarse al dispositivo sin ninguna autenticación por parte de la aplicación. No necesita autenticación porque una vez se dispone de la clave de cifrado del candado y como se construyen las ordenes, ya es posible conectarse y enviarles las ordenes cifradas que el candando sepa descifrar e interpretar.

La empresa Attify proporciona un script, es importante revisarlo porque puede generar algunos errores con las librerías de Python 3 ya que está escrito en Python 2 y por ejemplo la conversión de tipos a hexadecimal no se realiza igual.

El Script, está implementado con la librería bluepy basada en la librería bluez, lo primero que se realiza es un escaneo de los dispositivos que se encuentran al alcance de la máquina virtual, conforme va descubriendo dispositivos, intenta leer el nombre de los mismos, leyendo el handle 0x9 se debe encontrar uno con el nombre BlueFPL, esto se puede realizar también utilizando la dirección física del dispositivo. Esta información la muestra en abierto el dispositivo como parte de los servicios de descubrimiento.

Una vez encuentra el candando a su alcance llama a la función Connect() para realizar el emparejamiento, y le envía una petición de GET_TOKEN al handle 3 con la orden 0x06010101XXXXXXXXXX.

Una vez se ha enviado el token, el Script se queda esperando a recibir una notificación por parte del candando con el token que posteriormente se le añade a todas las ordenes (Ver Ilustración 30 e Ilustración 31).

Una vez con el token generado por el candando, es posible enviar la orden de apertura.

```
main(){
while(Scanner){
 if(nombre=BlueFPL){
    conectarse;
    Solicitar token al candando;
    if(waitToken){
      Envio comando apertura
    }else{
      FIN
    }
 }else{

 }
}
}
```

Ilustración 30 - Pseudocodigo Script

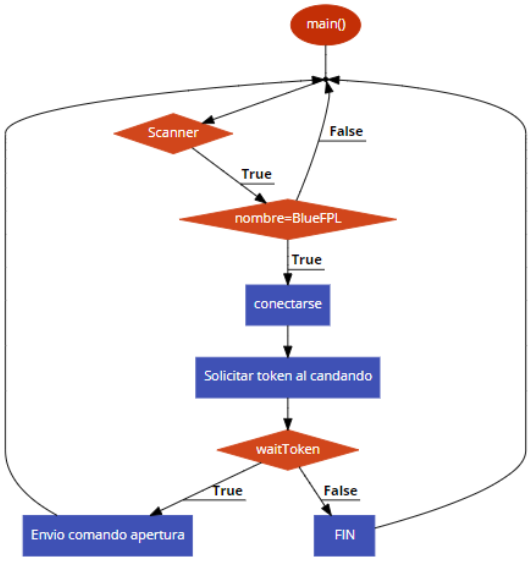

Ilustración 31- Diagrama de Flujo Script

Caso de uso real

¿Qué ocurre si nos encontráramos el candado en la calle?

Al ser un dispositivo BLE, lo normal es poder vincularse sin problemas y tomar el control del dispositivo.

Para comprobar si esto es cierto, es necesario crear una nueva cuenta en la herramienta OKLOK y una vez logado en la aplicación intentar vincularse al dispositivo.

La aplicación es capaz de ver el candado, pero en el momento de vincularlo lo que hace es comprobar si ese ID de candado (dirección física) está vinculado a otra cuenta y de esta forma no te permite vincular el candando a la aplicación, la propia aplicación se cierra para proteger la integridad del dispositivo. (véase Ilustración 32).

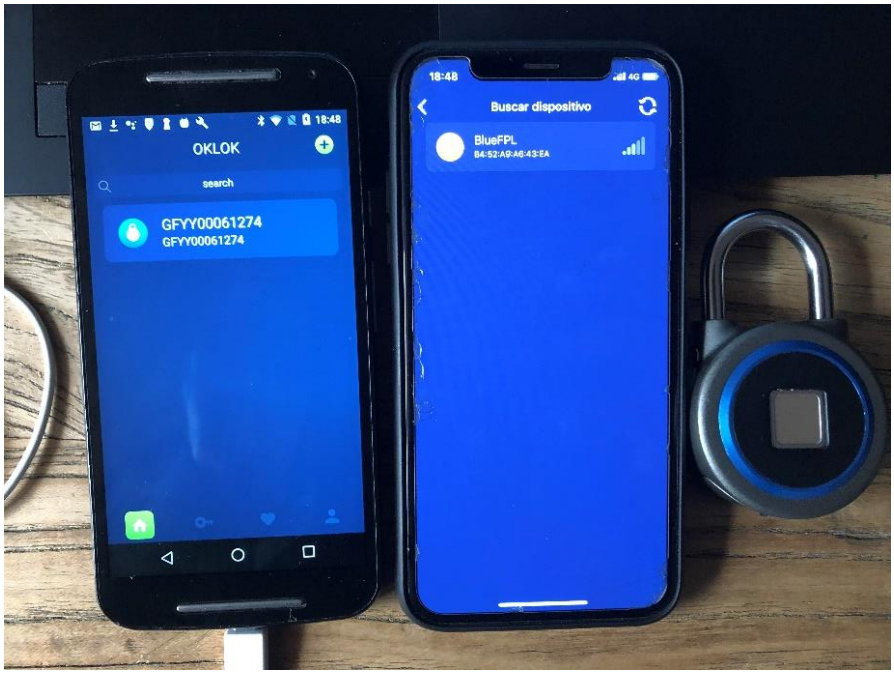

Ilustración 32 - Intento de Vinculación (Edición propia)

Conclusiones y trabajo futuro

Tras el contexto analizado y el trabajo práctico realizado, resulta indudable que los dispositivos Inteligentes ya forman parte de nuestro día a día. Si bien es cierto, que al ser una tecnología relativamente nueva no están centrando su desarrollo en afrontar un fallo crítico de seguridad, pero a medida que vayan tomando más relevancia, la seguridad será el factor diferencial.

El trabajo ha demostrado cómo se puede "hackear" un dispositivo. Supuestamente estos productos son seguros para cualquier usuario, por lo que se hace evidente que la industria debe comenzar a establecer la seguridad desde el diseño y comenzar a trabajar en un estándar global, que permita a estos dispositivos interactuar entre ellos de una forma segura.

La auditoría se considera incompleta, ya que, en la realización de la misma han surgido diferentes vectores de ataque que no han sido explotados. Como se mencionaba en el apartado *Alcance de la auditoría* es importante conocer hasta donde se va a llegar realizando la auditoria ya que su realización puede llevar a una auditoria del mismo dispositivo con un objetivo y alcance distintos.

Dado que la auditoria es incompleta se tendrían que abordar los vectores encontrados como continuación a este trabajo, para conocer cuáles son todas las vulnerabilidades reales del dispositivo. Los nuevos vectores de ataque que se encuentran son:

- La aplicación utiliza el algoritmo de encriptado AES ECB 128, que se considera vulnerable, por lo que seguramente se podría haber conseguido descifrar el payload mediante fuerza bruta.
- Conociendo que en cada sesión se negocia un nuevo token, sería posible con un sniffer bluetooth capturar el tráfico y una vez se ha obtenido el token, montar un proxy inverso, de esa forma sí funcionaria el ataque por repetición.

Recomendaciones:

- Utilizar un estándar, como la guía de diseño proporcionada por la GSMA, que abarca tanto el dispositivo como el ecosistema que los rodea.
- Realizar una auditoría de Pentesting basándose en el TOP 10 OWASP para IoT de forma que se revisen las 10 vulnerabilidades más típicas previo al lanzamiento de un producto.

Recomendación de diseño

Se propone un nuevo diseño en el cual se implementa un sistema de autorización vía JWT [37] para dispositivos IoT. Esto se realiza para que la aplicación no contenga la clave de cifrado y que sea un servidor de autorización a partir de la autenticación vía APP quien genere las ordenes ya cifradas para el candando. De esta forma la aplicación únicamente transmite el mensaje al candado ya que este no dispone de conexión a internet.

El usuario se loga desde la APP contra el servidor de autenticación de OKLOK.

1. Una vez logado se vincula el candando a la cuenta, siempre y cuando no esté vinculado a la cuenta de otro usuario de la app.
2. Desde la APP se envía un orden al candado para abrirse. Esta orden viaja al servidor de autorización de OKLOK, en ella viaja el token de autenticación, el id del candando, la orden a ejecutar.
3. El servidor de autorización le devuelve a la aplicación, mediante una comunicación bajo HTTPS y TLS 1.2 o superior, un mensaje donde el cuerpo del mensaje contiene la orden de apertura del candando cifrada únicamente con una clave única que contiene el candando de fábrica.
4. El candando desencripta el cuerpo del mensaje con su clave única y ejecuta la orden. De esta forma, únicamente el candando puede leer los mensajes, y el servidor generara el cifrado únicamente para que lo pueda desencriptar ese candando.

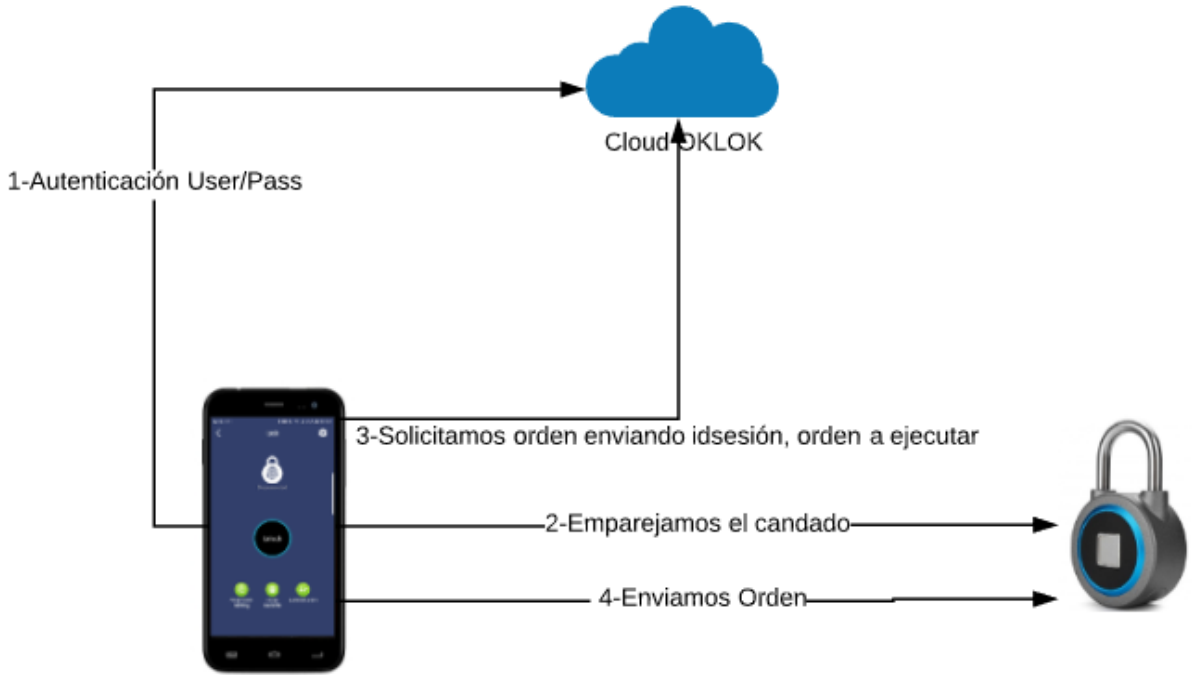

Ilustración 33- Diseño JWT para IoT sin conexión a internet

Con esta implementación se consigue solucionar la vulnerabilidad actual, ya que en ningún momento la información es visible y la aplicación es un mero intermediario.

En ambas implementaciones es necesario tener acceso al servidor para acceder a la aplicación, pero en esta es necesario tener acceso al servidor para ejecutar cada orden. En caso de que no haya acceso al servidor, siembre se podría desbloquear el candando, usando la interfaz para el reconocimiento de huella que hay sobre la superficie.

Lo que obtenemos con esta implementación es que la aplicación no trabaja en ningún momento con la clave de cifrado/descifrado. Por lo que la aplicación deja de ser un vector de ataque válido.

Adicionalmente vamos a utilizar un cifrado de clave asimétrica, asignándole al servidor web la clave privada y al candando la clave pública. De esta forma el servidor cifra la orden con su clave privada y solo el candando puede descifrarlo con la clave pública. Este es el caso más favorable dado que se consigue la identificación y autenticación

del servidor, ya que el candado sabe que sólo pudo haber sido el servidor quien utilizo su clave privada.

El cifrado asimétrico se realiza de esta forma, porque si utilizáramos un cifrado asimétrico corriente sería el servidor quien cifraría la orden con la clave pública del candado y el candando quien las descifraría con su privada. De esta forma, aunque se "hackeara" el candado solo contendría la clave pública y esto no representa ningún problema de seguridad ya que no hay forma práctica de conseguir la clave privada a partir de la clave pública.

Introduction

The motivation to do this "Pentesting Smartlock" paper is given by the current need to implement security by design in all types of systems and environment. The growing threat of cybercrime, is leading companies to invest large sums of money in protecting their assets.

However, the connected devices sector is not following that path, there is thinking that it is more important to protect a database with customer information than the device itself, which can be something as critical as a pacemaker. In this project, an intelligent smart lock is selected to perform an ethical hacking audit, given that the device is intended to protect an asset.

The objective is to have a high-level view about current state of IoT devices, especially of devices based on BLE technologies, and to know what their characteristics and risk are and how to mitigate them.

For understanding the problem, the work contain a practical case with a demonstration of an audit, based on laboratory work from the company Attify .

Therefore, the main thing is to obtain a context about the problem, carry out a practical demonstration, and finalize with the security recommendations

Conclusions and future work

After the analysed context and the practical work carried out, is undeniable that smart devices are already part of our daily life. Due to this is a new sector, they are not yet focusing their efforts on facing a critical security failure. As they become more relevant, the security will be the differential factor.

The work demonstrates how a BLE device can be hacked. These products should be secure for any user, so this makes clear that the industry must start to establish security from the design stage and start working on a global standard to allow the devices interact with each other in a secure way.

The audit is not considered entirely conclusive since different attack vectors have emerged during the implementation and have not been exploited. As it is mention in the section *Audit* scope, it is crucial to define the point of the audit as it could conclude from the same device different scopes and objectives.

Attack vectors founded during the audit should be considered as a continuation of this work, to identify what the real vulnerabilities of the device are. The new attack vectors found are:

- The application uses the AES ECB 128 encryption algorithm which is considered vulnerable, so it could surely have been possible to decrypt the payload using brute force.
- Each session negotiates a new token, it would be possible with a Bluetooth sniffer to capture the traffic and once the token has been obtained, mount a reverse proxy, that way if the reply attack would be work.

Recommendations:

- Use a standard guide, as the design guide provided by GSMA, that cover the device and the surrounding ecosystem.
- Perform a Pentesting audit, based on TOP 10 OWASP for IoT. This way, the most typical vulnerabilities will be reviewed before the launch of a product.

Design Recommendation

A new design is proposed, where an authorization system JWT for IoT devices is implemented. This is to aim that the application does not have the encrypted key. Instead, a security server from an authentication in the application will generate the orders already encrypted for the device. This way, the app only transmit he message to the smart lock, due to this does not have internet connection.

The user log in from the app to the OKLOK authentication server.

1- Once the user log in, the lock is associated to the account, provided it is not associated to other user account.

2 - The app sent the command to open to the lock. This command goes through the OKLOK authorization server and gives the authentication token, lock ID and the command to execute.

3 - The authorization server gives the app, through a HTTPS and TLS 1.2 (or above) a message with the open command, encrypted just with a unique key that the lock has predetermined.

4 - The lock decrypt the message with its unique key and execute the command. This way, only the lock can read the messages, and the server will generate the encrypted only for this lock.

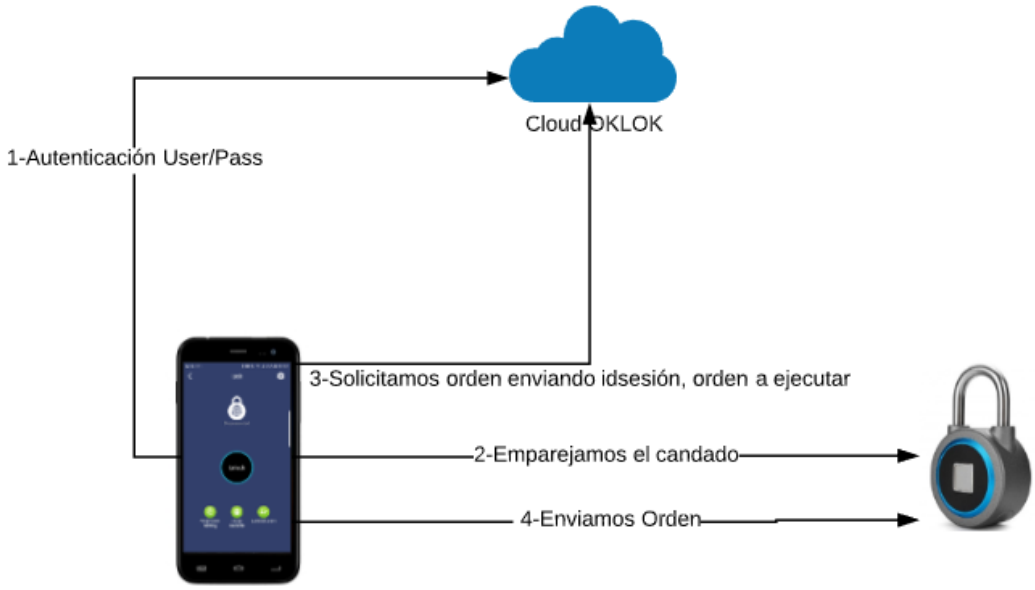

Ilustración 34- Diseño JWT para IoT sin conexión a internet

This implementation solves the existing vulnerability, due to in any time the information is visible, and the application is just an intermediary.

In both implementations, it is required to have access to the server to enter the application, but in this case, is necessary to have access to the server to execute every command. In case there is no access, the lock could be unlocked using the interface for fingerprint recognition.

What we could obtain from this implementation is that the application will not use the encrypted/decrypted key. The application will no longer be considered an attack vector.

Additionally, we will use an asymmetric encryption, giving the web server the private key and the lock the public key. This allows the server to encrypt the command with its private key and only the lock could decrypt with its public key. This is the most favourable case, given that the identification and authentication is through the server, and the lock knows that only the server could use its private key.

The asymmetric encrypted is using this way instead of the traditional one where the server is who encrypted the command with the public key, and the lock which decrypted with its private one. Thus, even if the lock is hacked, it will only have the public key information, which do not represent any security problem due to there is no way to get the private key from the public key.

BIBLIOGRAFÍA

[1] iniseg, «iniseg,» [En línea]. Available: https://www.iniseg.es/blog/ciberseguridad/que-es-el-hacking-etico/.

[2] «Attify,» [En línea]. Available: https://www.attify.com/.

[3] Gartner. [En línea]. Available: https://www.gartner.com/en/newsroom/press-releases/2019-08-29-gartner-says-5-8-billion-enterprise-and-automotive-io.

[4] P. Network. [En línea]. Available: https://unit42.paloaltonetworks.com/iot-threat-report-2020/.

[5] «Bluetooth about,» [En línea]. Available: https://www.bluetooth.com/about-us/.

[6] Bluetooth. [En línea]. Available: https://www.bluetooth.com/about-us/bluetooth-origin/.

[7] D. C. Yacchirema, «riunet.upv,» [En línea]. Available: https://riunet.upv.es/bitstream/handle/10251/123260/A%20Smart%20System%20for%20Sleep%20Monitoring%20by%20Integrating%20IoT%20with%20Big%20Data%20Analytics.pdf?sequence=1.

[8] link-labs. [En línea]. Available: https://www.link-labs.com/blog/bluetooth-vs-bluetooth-low-energy.

[9] E. Team, «everythingrf,» [En línea]. Available: https://www.everythingrf.com/community/what-is-the-difference-between-bluetooth-5-0-bluetooth-low-energy-bluetooth-v4-2-and-classic-bluetooth.

[10] «researchgate,» [En línea]. Available: https://www.researchgate.net/figure/Bluetooth-Low-Energy-Protocol-Stack_fig1_328541826.

[11] «learn.adafruit.,» [En línea]. Available: https://learn.adafruit.com/introduction-to-bluetooth-low-energy/gatt.

[12] L. Teschler, «microcontrollertips,» [En línea]. Available: https://www.microcontrollertips.com/breaking-ble-vulnerabilities-in-bluetooth-pairing-provide-openings-for-attack-faq/.

[13] CCN-CERT. [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.

[14] «mobilityarena,» [En línea]. Available: https://mobilityarena.com/bluetooth-mesh-will-connect-32000-devices/.

[15] ISACA. [En línea]. Available: www.isaca.org/Knowledge-Center/Research/Documents/Audit-Plan-Activities_res_eng_0316.pdf .

[16] I. Cooke, «ISACA,» [En línea]. Available: https://www.isaca.org/resources/isaca-journal/issues/2018/volume-5/is-audit-basics-auditing-the-iot#11.

[17] ISACA, «https://www.isaca.org,» [En línea]. Available: https://www.isaca.org/bookstore/bookstore-wht_papers-digital/whpiot.

[18] OWASP, «OWASP-TOP10-2018.pdf,» [En línea]. Available: https://owasp.org/www-pdf-archive/OWASP-IoT-Top-10-2018-final.pdf.

[19] GSMA, «gsma.com,» [En línea]. Available: https://www.gsma.com/membership/membership-types/.

[20] GSMA, «GSMA guidelines,» [En línea]. Available: https://www.gsma.com/iot/iot-security/iot-security-guidelines/.

[21] «Aliexpress,» [En línea]. Available: https://es.aliexpress.com/item/32904511661.html?spm=a2g0o.productlist.0.0.604b1047pmwNlq&algo_pvid=80b6f868-4b93-40fe-b710-6f6584153113&algo_expid=80b6f868-4b93-40fe-b710-6f6584153113-48&btsid=0b0a187915931232046838977e2228&ws_ab_test=searchweb0_0,searchw.

[22] «Play Store,» [En línea]. Available: https://play.google.com/store/apps/details?id=com.oklok.y&hl=es_419.

[23] «Frida,» [En línea]. Available: https://Frida.re/.

[24] M. K. y. M. Holtmann, «die.net,» [En línea]. Available: https://linux.die.net/man/1/hcitool.

[25] R. Gomez, «Universidad de valparaiso,» [En línea]. Available: http://profesores.elo.utfsm.cl/~agv/elo326/1s06/projects/rodrigoGomez/linux.html.

[26] «Oreilly,» [En línea]. Available: https://www.oreilly.com/library/view/getting-started-with/9781491900550/ch04.html.

[27] «Google Play,» [En línea]. Available: https://play.google.com/store/apps/details?id=com.alltek.android.bluetoothlegatt&hl=en_US.

[28] S. Avseyev, «github,» [En línea]. Available: https://github.com/greatscottgadgets/ubertooth/wiki/Ubertooth-One.

[29] «Wirishark,» [En línea]. Available: https://www.wireshark.org/.

[30] «developer.apple,» [En línea]. Available: https://developer.apple.com/bug-reporting/profiles-and-logs/?name=bluetooth.

[31] Bluetooth. [En línea]. Available: https://www.bluetooth.com/specifications/gatt/services/.

[32] skylot, «github,» [En línea]. Available: https://github.com/skylot/jadx .

[33] «Sonar Source,» [En línea]. Available: https://rules.sonarsource.com/java/type/Vulnerability/RSPEC-4432.

[34] «spouliot,» [En línea]. Available: https://spouliot.wordpress.com/2012/05/02/managed-crypto-vs-commoncrypto-deciphering-results/.

[35] «Google open source,» [En línea]. Available: https://opensource.google/projects/v8.

[36] «Github,» [En línea]. Available: HTTPS://GITHUB.COM/SENSEPOST/OBJECTION.

[37] «JWT,» [En línea]. Available: https://jwt.io/.

[40] K. Townsend, «learn.adafruit,» [En línea]. Available: https://learn.adafruit.com/introduction-to-bluetooth-low-energy.

[41] «Bugs Patterns,» [En línea]. Available: https://find-sec-bugs.github.io/bugs.htm#ECB_MODE.