Artículo Científico / Scientific Paper

UNIVERSIDAD POLITÉCNICA SALESIANA ECUADOR
INGENIUS
Revista de Ciencia y Tecnología
https://doi.org/10.17163/ings.n25.2021.09
pISSN: 1390-650X / eISSN: 1390-860X

Reduciendo la brecha de seguridad del IoT con una arquitectura de microservicios basada en TLS y OAuth2

Reducing the IoT security breach with a microservice architecture based on TLS and OAuth2

Diego Ordóñez-Camacho[1,*]

Recibido: 15-09-2020, Revisado: 19-10-2020 Aprobado tras revisión: 14-11-2020

Resumen

El Internet de las cosas es una de las tendencias más prometedoras en la actualidad. La rapidez de su adopción, sin embargo, ha provocado ciertas brechas críticas en la seguridad de los sistemas involucrados. Este proyecto analizó el problema de seguridad de una manera amplia, pero enfocándose en entornos de tipo hogar inteligente, donde el uso de dispositivos con tecnologías ampliamente heterogéneas genera problemas en la autenticación con múltiples servicios, y en la confidencialidad de los datos, si la red llegara a verse comprometida. Para atacar estos problemas, se juntaron tecnologías de última generación como OAuth2 y TLS, entre otras, junto a una metodología arquitectural de microservicios de acoplamiento ligero, para generar una arquitectura IoT segura y de amplio alcance, respaldada y validada por una implementación de referencia. La división en capas funcionales permite que tanto los dispositivos y sensores fijos como aquellos móviles, puedan acoplarse al sistema de manera transparente y fluida. El esquema de seguridad estructurado en tres niveles incrementales permite que cada equipo pueda integrarse al que mejor se adapte tanto a sus recursos computacionales como al tipo de información que debe entregar o consumir. Los resultados muestran la flexibilidad de la solución y la solidez del esquema de seguridad presentado.

Palabras clave: IoT, microservicios, arquitectura de *software*, seguridad de sistemas, TLS, OAuth

Abstract

The Internet of Things has emerged as one of the most promising trends today. The speed of its adoption, however, has caused certain gaps. Amongst the most critical there is the one related with the security of the systems involved. This project addressed the security problem in a broad way but focusing on smart-home environments, where the use of devices with widely heterogeneous technologies and multiple services, generates problems with authentication and with the confidentiality of the data, if the network is compromised. To tackle these problems, state-of-the-art technologies such as OAuth2 and TLS, among others, were put together, along with an architectural methodology of lightly coupled microservices. As a result, a secure and broad range IoT architecture was built, backed up and validated by a reference implementation. The division into functional layers enables both fixed and mobile devices and sensors, to get connected into the system transparently and fluently. The security scheme structured in three incremental levels enables a better device integration, at the level that best adapts to its computing resources and the type of information it shares. The results show the flexibility of the solution and the robustness and novelty of the security scheme presented.

Keywords: IoT, microservices, *software* architecture, systems security, TLS, OAuth.

[1,*]Grupo de Investigación en Informática (GrIInf), Universidad UTE – Ecuador.
Autor para correspondencia ✉: dordonez@ute.edu.ec. http://orcid.org/0000-0001-8390-634X

Forma sugerida de citación: Ordóñez-Camacho, D. (2021). «Reduciendo la brecha de seguridad del IoT con una arquitectura de microservicios basada en TLS y OAuth2». Ingenius. N.° 25, (enero-junio). pp. 94-103. doi: https://doi.org/10.17163/ings.n25.2021.09.

1. Introducción

El Internet de las cosas, IoT por sus siglas en inglés (*Internet of Things*), es una tecnología que ingresa con fuerza en la realidad de las personas. Todos los entornos están involucrados, urbanos, industriales, de oficina o el hogar. El interés generado y la rapidez de adopción de la tecnología han producido un cierto desorden e informalidad en el proceso. Esto tiene como consecuencia que elementos importantes se dejaron de lado, uno de los más relevantes es el de la seguridad [1].

En principio, la seguridad del IoT no tiene por qué distar de la seguridad en una red típica de computadores. En la práctica, sin embargo, hay dificultades propias del entorno que complican aún más el problema de la seguridad. Muchos dispositivos para IoT son limitados computacionalmente, lo que impide utilizar varios mecanismos robustos de seguridad conocidos. La gran cantidad de dispositivos que pueden estar involucrados en una red IoT, así como el incremento exponencial en el número de interacciones, agrava el problema. La diversidad de los equipos que se utilizan, tanto en *hardware* como en *software*, complican la posibilidad de generalizar las soluciones propuestas [2]. Hay una gran variedad de métodos y herramientas que pueden utilizarse para acometer la labor [3]. En este trabajo la intención fue la de utilizar aquellas técnicas que, así como el IoT, marcan tendencia y son utilizadas exitosamente en otros campos relacionados. De entre estas técnicas, las más prometedoras fueron los microservicios [4] y OAuth2 [5], las cuales, al aunarse con técnicas y tecnologías, tal vez un poco más tradicionales a la base, pero igualmente exitosas como TLS [6] y MQTT [7], proporcionaron un entorno estructurado apropiado.

El objetivo general fue proporcionar al mundo del IoT una alternativa arquitectural segura y adaptada a las nuevas tendencias tecnológicas, que pueda ser utilizada de manera genérica en una multitud de situaciones. Más específicamente, primero, se dio especial relevancia a generar una alternativa para hogares inteligentes, por esto la metodología propicia la clara división de funciones, pero cuidando la integración fluida y la disponibilidad de herramientas para la creación de nuevas utilidades. Segundo, de entre los múltiples problemas de seguridad existentes, se atacó a aquellos relacionados con la autenticación de clientes al llamar a múltiples servicios y con la confidencialidad al transmitir información, especialmente al comprometerse la red. Finalmente, conscientes de las restricciones de *hardware* de muchos dispositivos, sobre todo, sensores, se trabajó en una arquitectura que contemple una jerarquía de varios niveles de seguridad, permitiendo una interconexión ajustada a las capacidades de varios tipos de equipos, especialmente en ambientes con tecnologías heterogéneas.

1.1. Trabajos relacionados

En el ecosistema de dispositivos IoT, estos son en gran medida poco seguros, por ser equipos pequeños y de bajo consumo energético, por ende, también tienen recursos computacionales limitados. Esta última condición afecta mucho al momento de intentar incluir en ellos esquemas de seguridad complejos [8]. A nivel industrial hay varios emprendimientos aplicando IoT, por ejemplo, en el transporte inteligente [9, 10] y en la agricultura [11]; sin embargo, uno de los grandes objetivos actuales relacionados con la automatización en el hogar y oficina es la vida conectada (*connected living*). Este objetivo requiere de importantes avances en el campo del IoT, donde se requiere brindar respuestas para los problemas relacionados con el enorme incremento de dispositivos que deberán interactuar [12]. Un caso particular en el IoT es aquel de los hogares inteligentes, dado que muchas veces las soluciones que se implementan son *ad-hoc*, por parte de los mismos usuarios, quienes suelen intentar reducir al máximo costos y esfuerzos, lo cual generalmente se ve reflejado en un esquema de seguridad mínimo y probablemente inexistente [13].

Los dispositivos involucrados requieren interconectarse en un esquema de muchos-a-muchos. Para asegurar el intercambio de información es necesario implementar un sistema de administración de identidades que escale apropiadamente. En este sentido, Ayed, Boujezza y Riabi [14] proponen un sistema para el hogar que combina EAP, OAuth y DTLS. También preocupados por la administración de identidades y el control de acceso, Fernández, Alonso, Marco y Salvachúa [15] confirman las posibilidades de OAuth y hacen una propuesta arquitectural compatible con servicios.

Bugeja, Jacobsson y Davidsson [16] analizan la problemática de la seguridad IoT en el hogar y destacan lo importante que es evitar que los sensores capturen y distribuyan, de manera indiscriminada, los datos del hogar. A manera de ejemplo presentan el caso de las conversaciones privadas, las cuales no deberían ser publicadas. Entre los enfoques posibles, mencionan como una alternativa viable, aquel orientado a servicios, para equilibrar entre centralización y distribución del control; más aún, la tendencia en *software* y aplicaciones distribuidas parecería dirigirse en general hacia el empleo de microservicios [17–19]. Ahondando en esta línea, Díaz-Sánchez, Marín-López, Almenarez, Arias y Sherrat [20] destacan los beneficios que una arquitectura de microservicios, adicionalmente basada en TLS/PKI, puede tener en IoT, al aligerar las tareas de desarrollo y mantenimiento, beneficioso tanto para proveedores y distribuidores como para usuarios, al mismo tiempo, reforzando la seguridad de interconexión. Estudios de caso como el de Urien [21] confirman de manera práctica las posibilidades de estas técnicas.

Si bien en la mayoría de los trabajos relacionados es SSL/TLS el mecanismo de seguridad y cifrado preferido, es muy interesante lo que proponen Hoz *et al.*, [22] al analizar las complicaciones que a nivel práctico pueden existir en IoT con TLS. Dicho trabajo, entonces, analiza las posibilidades de utilizar SSH y destaca las ventajas que proporciona el tema de compresión de datos incluido en dicho protocolo, que especialmente se muestra ventajoso al trabajar sobre HTTP.

Khan, Anwar, Azam, Samea y Shinwari [23] aportan al entorno del IoT con un enfoque administrado por modelos y proponen justamente un modelo orientado a OAuth con una fuerte inclinación UML. Esta propuesta, a través de transformaciones, podría ser adaptada a una arquitectura específica, brindando la posibilidad de personalizarse al entorno requerido. Otra interesante propuesta arquitectural y de seguridad es aquella presentada por Kim, Wasicek, Mehne y Lee [24], donde en lugar de un mecanismo tradicional como el presentado por las autoridades de certificación en SSL, se utilizaría un acercamiento de autoridades de certificación locales, las cuales de manera más frecuente, pero también con un proceso más ligero, autenticarían los equipos de IoT. Se puede considerar que un punto medio entre las dos propuestas que acabamos de revisar sería aquella de Pahl y Donini [25] la cual usa certificados tradicionales, pero con autenticación próxima, más bien a nivel de nodos; destacan que este mecanismo podría complementarse con alguno de autorización, como OAuth o similares.

Sciancalepore, Piro, Caldarola, Boggia y Bianchi [26] dan énfasis a OAuth, pero sobre todo, con la particularidad de concentrar su arquitectura de seguridad en el equipo gateway, donde reside la estación de base o sink node, que se encarga del procesamiento pesado de autenticar, autorizar y establecer los lazos entre clientes y recursos. Este enfoque es muy relevante cuando consideramos lo susceptibles que son aquellos equipos de borde ligados al *edge computing* [27], a través de los cuales se puede comprometer todo un sistema de IoT. Uno de los puntos álgidos de la seguridad en los sistemas de borde en IoT suele ser el relacionado con el uso de MQTT (o protocolos similares), por lo cual varios trabajos, como el de Singh, Rajan, Shivraj y Balamuralidhar [28] proponen mejoras sobre dicho protocolo.

2. Materiales y métodos

Para este trabajo se consideró principalmente que dentro del ecosistema IoT es necesario segmentar la ubicación, alcance y acceso de los equipos involucrados en dos capas: local o de borde y centralizada.

En la capa local, representada esquemáticamente en la Figura 1, tenemos aquellos elementos que estarán invariablemente instalados en la casa u oficina inteligente, como es el caso de sensores, actuadores, procesadores de borde como gateways y brókeres, y dispositivos móviles de usuario.

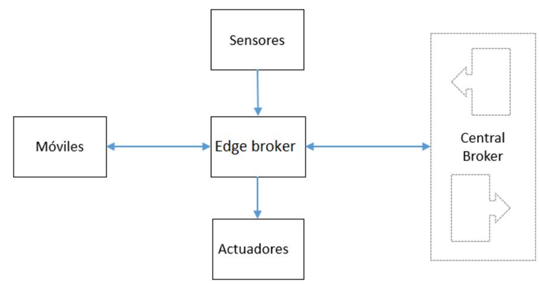

Figura 1. Componentes en la capa local o de borde del sistema IoT

Los sensores son todo equipo capaz de capturar fenómenos físicos, eventos virtuales o señales periódicas. Aquellos sensores con la capacidad necesaria pueden comunicarse directamente con el bróker central, caso contrario interactuarán con equipos en el subsistema de procesamiento de borde. Los sensores serán estáticos, cuando emitan una señal constante que sería utilizada generalmente por equipos móviles desplazándose en el entorno como balizas Bluetooth para posicionamiento. Los sensores dinámicos capturarán mediciones del ambiente, las cuales variarán dependiendo de las condiciones del entorno, como luminiscencia, temperatura, humedad, entre otros.

Los actuadores permitirán la interacción con *hardware* o *software* generando eventos o acciones. Recibirán instrucciones sea directamente del bróker, o desde algún preprocesador cuando es necesario. Se dividen en locales, ubicados en el entorno inteligente, como controladores de luz o temperatura; pueden ser también remotos como aquellos capaces de enviar instrucciones, probablemente por la red, para algún equipo distante, pero controlado desde la casa u oficina inteligente, como cuando se requiere enviar un sms, correo o tuit.

Finalmente, esta capa contempla los equipos preprocesadores, que también pueden llamarse procesadores o brókeres de borde. Estos capturan información en bruto proveniente de los sensores para reenviarla al bróker centralizado o al actuador cuando el sensor es incapaz. La información se podrá enviar tal como se recibió del sensor, o preprocesarla y enviar este resultado. Estos equipos pueden ser Raspberri Pi o computadoras pequeñas como las tabletas.

En la capa centralizada, presentada en la Figura 2, se consideran los equipos encargados de la coordinación general de todos los componentes y se considera que son necesarios tres subsistemas: administración, procesamiento y persistencia. Estos subsistemas se ligan entre ellos y también con la capa de borde mediante un bróker centralizado.

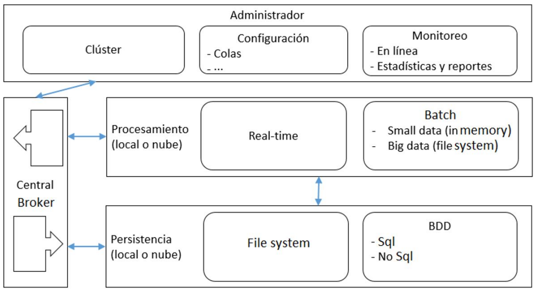

Figura 2. Componentes en la capa centralizada del sistema IoT

El subsistema de administración define los parámetros y la configuración del sistema presenta las interfaces web para que los usuarios administradores interactúen con todo el sistema. En el caso de que el procesamiento central se haga con varios equipos, también administra el clúster resultante. Parte principal dentro de este subsistema es entonces el monitoreo, que permitirá a todo tipo de usuarios revisar la información pertinente, de preferencia mediante *dashboards* y cuadros estadísticos.

Para todo el procesamiento pesado de información, el subsistema correspondiente toma los datos recolectados desde el bróker y los procesa según se defina por las aplicaciones o necesidades específicas del sistema IoT. En general el procesamiento se dividirá dependiendo, sobre todo, de la urgencia del procesamiento, en *real-time*, que procesa los datos en un flujo continuo, según vayan llegando de los sensores; *in memory*, que recolecta la información en la memoria del clúster, dependiendo de las necesidades, y la procesa en pequeños lotes; y *batch*, que interactúa en general con el sistema de almacenamiento, para procesamientos donde la cantidad de datos es mayor que la que cabe en la memoria viva del sistema.

La información generada por el sistema IoT se direcciona al módulo de persistencia, el cual salvaguarda los datos para su uso posterior sea en el desarrollo de modelos o en la generación de reportes. Varias alternativas deben considerarse, dependiendo del tamaño de la información y la forma como será accedida. Al menos es necesario considerar un soporte HDFS, uno de bases de datos tanto SQL como NoSQL, y en caso de que la evolución del sistema así lo requiera, un soporte en la nube.

Finalmente, todo el sistema, y más específicamente las capas de borde y centralizada, deben conectarse e intercambiar información, lo cual se consigue mediante un bróker central, encargado de manejar todas las colas de mensajes reduciendo así la complejidad de las interacciones.

3. Resultados

El diseño arquitectural resultante tomó en consideración, sobre todo, la necesidad de asegurar el intercambio de información de todos los componentes del sistema, procurando cuidar en todo momento la rapidez de cálculo. Estos elementos requieren conciliar características que muchas veces son incompatibles. Por ejemplo, sistemas de criptografía más robustos pueden requerir más poder de cómputo del que muchos dispositivos ligeros, como sensores, proporcionan.

La arquitectura final diseñada, implementada y probada, es aquella esquematizada en la Figura 3, que se describirá en detalle a continuación. En primer lugar, se puntualizarán los componentes involucrados, para luego presentar la funcionalidad de seguridad en general.

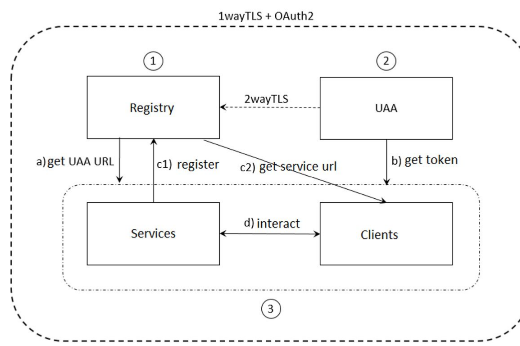

Figura 3. Arquitectura segura

3.1. Componentes

En esta sección se intentarán definir de manera general los tipos de componentes o equipos involucrados en la arquitectura propuesta en la Figura 3, en la cual se cuenta básicamente con los encargados del servicio de registro (*Registry*), los equipos brindando servicios de autenticación para clientes y usuarios (UAA) y luego ya, de forma amplia, todos los equipos brindando servicios generales, así como todos los equipos cliente (*Services* y *Clients*). Si bien por simplicidad se hará referencia a los componentes en singular, la arquitectura considera que, para cada categoría o tipo de componente, especialmente los servicios, estos pueden trabajar en clústeres.

3.1.1. Servicios de registro (REG)

La labor principal del REG es permitir a los servicios registrarse mediante IP y alias (nombre de servicio) y ponerlos así a disposición de los clientes, quienes se conectarán al REG para pedir la información con la cual se conectarán finalmente a los servicios de interés. El REG proporciona también un servicio de balanceo de carga, al detectar que un servicio está registrado en clúster (varios equipos con el mismo servicio).

El primer punto de contacto para todos los demás componentes del sistema, sean estos servicios o clientes, es el REG, el cual requiere una IP estática; todos los demás componentes del sistema, sin embargo, pueden trabajar con IP dinámicas, mediante un DNS.

3.1.2. Servicios de autenticación (UAA)

El UAA toma sus siglas del inglés *User Authentication and Authorization*. Dentro de nuestra arquitectura básicamente nos proporciona el servicio de autenticación, que trabaja bajo OAuth2. El UAA guarda los datos de todos los clientes en el sistema, incluyendo sus roles; con esta información los servicios usuarios del UAA pueden también autorizar o no el uso de ciertos elementos. Cualquier componente puede conectarse con el UAA para, mediante sus credenciales de cliente (usuario y *password*) solicitar un *token* de acceso. De la misma manera, cualquier componente del sistema puede solicitar al UAA la validación de un *token* recibido por parte de un tercero.

3.1.3. Servicios genéricos (SRV) y equipos cliente (CLI)

La última categoría de componentes acoge todos los demás servicios y todos los clientes. En general estos componentes interactuarán entre ellos luego de haberse registrado/autenticado en el sistema con la ayuda del REG y del UAA. Los servicios pueden ser muy diversos y está en manos del administrador del sistema decidir cuáles requerirá. Sin embargo, en nuestra arquitectura para IoT, hay algunos que resultan fundamentales, por lo cual han sido implementados en el sistema de prueba, y serán mencionados a continuación.

Para permitir la interconectividad y, al mismo tiempo, reducir su complejidad, se implementó el servicio de mensajería, que en la metodología está representado por el bróker central (Figuras 1 y 2). El bróker está en capacidad de recibir y distribuir todos los mensajes circulando en el sistema y básicamente permite que en general todos los servicios y los clientes puedan establecer una conexión única con el bróker, para depositar mensajes y recuperarlos de una o más colas. Este bróker puede trabajar con cualquier protocolo de comunicación, o una combinación de ellos. Sin embargo, dado que en el mundo del IoT, al menos en la actualidad, el protocolo más expandido es el Message Queuing Telemetry Transport (MQTT), es el que se usa en la implementación que aquí se presenta.

Otro de los servicios implementados para la prueba de concepto de la arquitectura es el que tiene relación con la persistencia, como soporte necesario para posteriormente implementar procesamiento en batch. Para esto se implementó un servicio de tránsito que toma la información del bróker y la traslada a un clúster Hadoop [29], donde se pueden utilizar distintos tipos de herramientas de dicho ecosistema para el tratamiento de la información. Uno de los casos que se trabajó, dada la naturaleza de la información IoT, especialmente aquella en proveniencia de los sensores, fue el de series temporales. Para ello se construyeron dos servicios de series de datos, proporcionando de esta manera, graficación y análisis de tendencias, entre otros.

En cuanto a los clientes, aquí se consideran todos los sensores, los cuales entregan información al sistema, los actuadores, que reaccionan con el entorno gracias a la información del sistema, y todos aquellos dispositivos, móviles o de escritorio, que permitan al usuario ingresar para configurar el sistema, recolectar información procesada, o incluso actuar también como sensores y actuadores. Un móvil, por ejemplo, puede entregar información de la actividad del usuario o enviar tuits automáticamente.

3.2. Esquemas de seguridad

La arquitectura de seguridad del sistema comprende tres escenarios fundamentales: el básico, al cual todos los elementos deberán ceñirse en sus transacciones, salvo que se especifique de otra manera; el esquema ligero, generalmente usado únicamente al iniciar un proceso de trabajo en el sistema; y el fortalecido, para relaciones de confianza entre servicios.

3.2.1. Esquema básico

Este es el esquema por defecto que los componentes del sistema utilizarán en sus transacciones. Este esquema está representado en la Figura 3 por la línea punteada que engloba el sistema, y utiliza una combinación de TLS de una vía más OAuth2. Todo servicio debe proporcionar su certificado público de seguridad (PKI) a los clientes, quienes podrán así validarlo con la autoridad de certificación (CA). Asimismo, todo cliente deberá proporcionar a los servicios un token de acceso OAuth para que estos puedan validarlo con el servicio de autenticación.

El uso de TLS, en nuestro esquema, es necesario especialmente para poder cifrar el contenido de la información que se transmite. Se utiliza únicamente del lado de los servicios para limitar al máximo la sobrecarga que implicaría, sobre todo, a nivel de administración (pero también de recursos y procesamiento), utilizarlo en todos los componentes. La brecha de seguridad que aparece se compensa con el uso de OAuth2, mediante el cual los clientes son a su vez validados por los servicios.

3.2.2. Esquema ligero

Este es un esquema que se pudiera considerar inseguro, razón por la cual se proporciona solamente para aquellos casos donde no se puede aún obtener un *token* de acceso, o cuando se considera redundante solicitarlo.

Dos casos existen al momento en el entorno de trabajo, que implementan este esquema. Cuando los componentes ingresan al sistema para poder iniciar sus transacciones, es en general necesario contar con el *token* OAuth, pero dado que la IP del servidor UAA puede haber cambiado, el primer paso es contactarse con REG para solicitar la IP actualizada. Para esta conexión el cliente aún no tiene el *token*, razón por la cual no se puede trabajar con el esquema básico. La segunda implementación de este esquema se aplicó para evitar conexiones redundantes innecesarias y se da cuando el servicio recibe el *token* y debe validarlo con el UAA. Dado que el servicio ya se validó inicialmente con el UAA se evita duplicar este paso.

Es para este tipo de casos que entra en juego el esquema ligero. El servicio proporciona su PKI con lo cual se cifra la comunicación, pero el cliente no está obligado a validarlo (aunque se recomienda que lo haga), el servicio proporciona un punto de acceso «inseguro» para el cliente, el cual no exige el *token*. Este es el esquema proporcionado por REG exclusivamente para poder entregar los datos del UAA.

3.2.3. Esquema fortalecido

Similar al problema trabajado en el esquema ligero, en ocasiones dos servicios requieren interconectarse, pero al menos uno de ellos (quien actúa como cliente) está en incapacidad de obtener su *token* de acceso. Al tratarse de servicios, no es conveniente abrir un canal inseguro como se hace en el esquema ligero. Para mantener el estándar de seguridad, entonces, se decidió implementar un esquema TLS de doble vía, lo cual es posible, sin incurrir en mayor sobrecarga, dado que, al ser servicios, ya poseen de todas maneras su PKI. Adicionalmente que en general los servicios se ejecutarán en equipos con mayor capacidad de proceso.

Esta implementación requiere también un canal dedicado para poder ejecutar este tipo de validación y el ejemplo está dado por la comunicación entre el REG y el UAA. El UAA es el que proporciona los *tokens* de acceso y por consiguiente debería validarse a sí mismo lo cual generaría un hueco de seguridad. El REG, entonces, abre un canal dedicado para que en todo momento un servicio UAA pueda registrarse por esa vía. Al conectarse el UAA con el REG, intercambian sus respectivos PKI, validándose mutuamente por TLS, sin reducir el estándar de seguridad del sistema.

3.3. Funcionalidad

Retomando la Figura 3, la línea punteada representa el alcance del esquema de seguridad básico, el cual engloba todo el sistema. Internamente pueden verse los números 1, 2 y 3, encerrados en círculos, los que indican el orden de arranque recomendado para garantizar la fluidez del servicio. En la práctica, al menos los servicios, cuentan en su biblioteca de base con la funcionalidad para reintentar la conexión cuando este orden de inicio no se respeta. Sin embargo, esto puede ser susceptible a demoras innecesarias.

En primer lugar, se inicia el servidor de registro, REG, el cual brindará un punto central de acceso para la adquisición de la información de contacto para los demás servicios: todo servicio registrará en el REG su respectiva IP y su alias (nombre de servicio) y todo cliente buscará aquí, por alias, la IP del servicio requerido para poderse conectar con él. REG ofrece tres puntos de acceso, cada uno de los cuales debe manejar un modo de seguridad diferente: el primero, ligero, permite a cualquier cliente obtener la IP del UAA sin ninguna seguridad adicional; el segundo modo, fortalecido, permite la conexión del UAA mediante TLS de dos vías; el último, básico, que requiere de OAuth2, permite a los clientes pedir información de servicios, y a los servicios registrar su información de contacto.

En segundo lugar, se inicia un servidor de autenticación (UAA), el cual brindará credenciales OAuth2 a los clientes. Entre el UAA y el REG se utiliza el mecanismo de seguridad reforzado, con validación TLS de dos vías. El UAA se conecta con el REG, así como cualquier otro servicio, para entregarle su IP y alias y estar así disponible para todo el sistema. Una vez que estos dos servicios, REG y UAA, están en línea, todo el resto de los componentes, servicios y clientes pueden iniciar su trabajo.

Finalmente, entonces, como punto 3, cualquier otro componente, sea este servicio o cliente, procederán de la siguiente manera: en primer lugar, utilizando el esquema de seguridad ligero, se conectarán con el REG para pedir la IP del UAA; establecen la conexión con el UAA y solicitan el *token* de acceso, utilizando sus credenciales cliente. Con el *token* de acceso en mano ya se puede utilizar el esquema de seguridad básico y, en el caso de los servicios, se registrarán con el REG, entregando IP y alias, para quedar a la espera de peticiones de clientes, o actuar como cliente de otro servicio, según sea necesario. En el caso de un cliente, el siguiente paso es utilizar un servicio, donde aplicará el esquema de seguridad básico; se conecta al REG y mediante un alias solicita la IP del servicio de su interés, para luego conectarse con dicho servicio. La última acción viene a ser la de los servicios que reciben una petición de un cliente ya que en este caso deberán, mediante el esquema ligero, conectarse al UAA y pedir la validación del *token* de acceso, con lo cual podrían atender la petición del cliente.

3.4. Implementación y pruebas

La decisión de implementación fue principalmente que cada uno de los componentes pueda ser ejecutado en una variedad de equipos y con la menor interdependencia, para lo cual se procedió a trabajar en una arquitectura de microservicios que permita su despliegue sea como procesos independientes, o dentro de una estructura de contenedorización, como Docker [30]. Para los servicios y clientes de escritorio se utilizó Java con Spring Boot en general. Para el servicio de mensajería central se decidió trabajar utilizando el protocolo MQTT y para el desarrollo del bróker se tomó como base la biblioteca Moquette [31], a la cual se modificó para añadirle soporte para OAuth2 principalmente. Para los servicios de persistencia se utilizó como base un clúster HDFS [32] en la red local, y sobre él se construyeron servicios de series de tiempo, que, de acuerdo con los intereses actuales del proyecto, eran los más adecuados para procesar los datos en proveniencia de los sensores. Se interactuó con OpenTSDB [33] y Prometheus [34].

En cuanto a los clientes, se decidió construir bibliotecas genéricas para distintos tipos de sistemas, las cuales faciliten el proceso de desarrollar las aplicaciones específicas. Se desarrolló una biblioteca cliente Java, una para móviles Android, una para Arduino MKR [35] y otra para ESP32 [36], estas tres últimas basadas en Eclipse Paho [37]. Las bibliotecas de Arduino fueron orientadas exclusivamente a su uso en controladores de sensores y actuadores.

Las pruebas del sistema se efectuaron en un ambiente de oficina controlado. El equipo principal fue un RPi 3B+ que hizo las veces de Gateway wifi, brindando entre otros los servicios DNS y NTP. Se utilizaron simultáneamente controladores MKR 1010 y ESP32, los cuales recibían en permanencia información de sensores de temperatura, humedad y ruido, como el DHT11 y el KY038 [38]. El RPi3B+ también acogió los servicios REG, UAA y el bróker de mensajería MQTT. Los servicios de persistencia sobre HDFS, así como los de TSDB se instalaron en Linux sobre un i5-4210U con 8 GB de RAM. En este último equipo se instalaron también servicios genéricos para envío y recepción de mensajes, con los cuales se verificó la fluidez de la interacción. Como cliente móvil se utilizó un N9005, que tuvo dos funciones: sensor bobo, enviando en gran cantidad números aleatorios al sistema, y actuador ligero, avisando cada vez que las mediciones de los sensores reales sobrepasaban ciertos niveles parametrizados en la app. En la Figura 4 se presenta uno de los setup del sistema de pruebas.

En la experimentación relativa a las pruebas de seguridad, se decidió asumir que un posible atacante se encontraba ya conectado a la red local y que estaba, potencialmente, en la capacidad de hacer cualquier petición y capturar todo el tráfico. En el primer test se utilizó *Zap* para efectuar un *scanning* tanto activo como pasivo, y se verificó que la seguridad se mantenía (tráfico encriptado), salvo en el caso (documentado en la arquitectura) del esquema ligero.

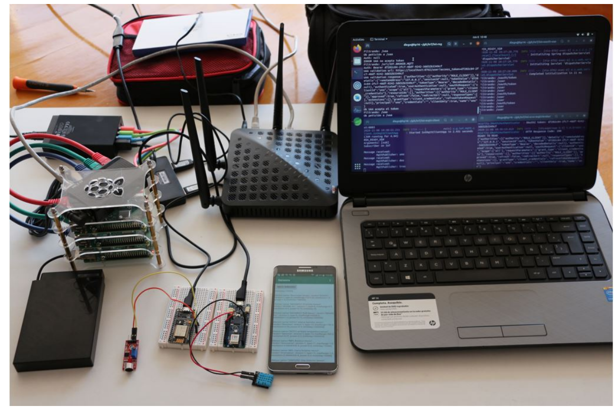

Figura 4. *Setup* de pruebas: de izquierda a derecha, se observa un clúster de 3 Raspberry Pi ejecutando todos los servicios sobre Docker; un ESP8266 monitoreando nivel de ruido con un KY038; un MKR1010 monitoreando temperatura ambiente con un DHT11; un N9005 inyectando mensajes aleatorios, y una laptop monitoreando todos los servicios.

Luego, se utilizó Wireshark para una verificación más bien de corte manual, analizando los paquetes por tipo de conexión o esquema, y nuevamente se validó, como esperado, que, salvo el esquema ligero, todo el tráfico se mantenía cifrado, como se presenta en la Figura 5.

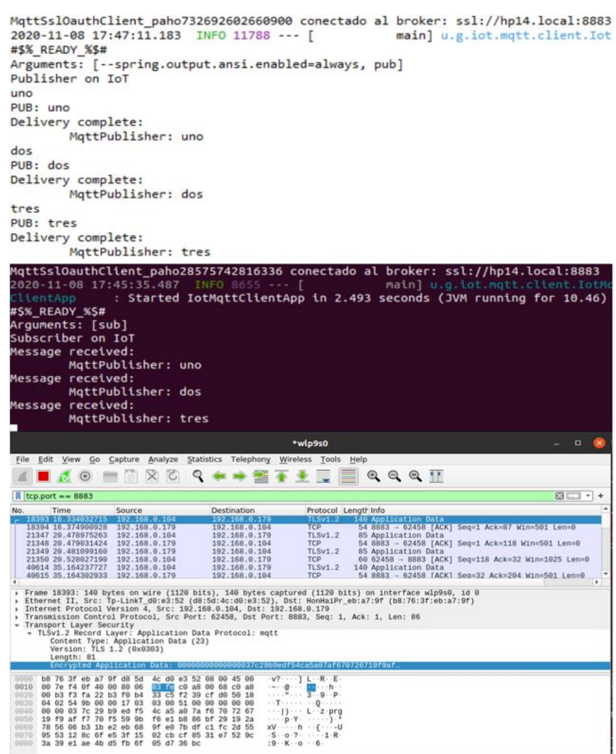

Figura 5. Ejecución y trazas: de arriba hacia abajo, primero, un cliente enviando tres mensajes simples de texto; luego el receptor obtiene los mensajes y, finalmente, la traza de Wireshark que captura los paquetes y verifica que, en tránsito, se encuentran cifrados.

4. Discusión

El proceso de incluir esquemas de seguridad complejos en dispositivos ligeros de IoT, no es trivial como ya lo afirmaron Khan y Salah [8] y en este trabajo se coincide con esta afirmación. Dos casos destacables fueron que el manejo de TLS con certificados autogenerados para MKR 1010 requirió regenerar todo el firmware para incluir la CA, y que no se pudo llevar a cabo en controladores ESP8266 debido a la indisponibilidad, en la práctica, de bibliotecas *open-source* suficientemente completas para garantizar el nivel de seguridad esperado.

El enfoque adoptado retoma en general las advertencias hechas por Lin y Bergmann [13] e intenta proporcionar un sistema suficientemente completo y sencillo para que con mínimo soporte técnico pudiera ser implementado en el hogar, y luego administrado directamente por el usuario.

Esto, claro, presenta limitaciones dadas por la gran variedad de tipos de usuarios que pueden desear incluirse en el IoT. Sin embargo, las pruebas realizadas apuntan a que el corazón del prototipo permitiría proporcionar esto, si se logran incluir algunas facilidades a nivel de interfaz de usuario, equipos e instaladores.

Este trabajo confirma lo ya expuesto por Ayed *et al.*, [14] y por Fernández *et al.*, [15] en relación con el problema del escalamiento de identidades y el beneficio que puede obtenerse tanto con OAuth como con un enfoque por servicios. Delegar en un punto único la autenticación, confiando luego en credenciales temporales, como lo permite OAuth2, mantiene la infraestructura de identidades ligera, mientras que la distribución en servicios brinda una gran facilidad a la hora de replicar y redundar servicios en clúster cuando la carga de trabajo del sistema así lo amerita. Es justamente este enfoque de servicios / microservicios lo que permite que trabajar con TLS no sea demasiado exigente a nivel de mantenimiento, como destacaron Díaz-Sánchez *et al.* [20].

Esta propuesta reconoce la importancia que a nivel de seguridad debe darse a los equipos de borde y siguiendo la línea de Shapsough *et al.*, [27] y de Sciancalepore *et al.*, [26] refuerza especialmente el equipo gateway para luego poder implementar mejoras en el protocolo MQTT, al estilo de lo realizado por Singh *et al.*, [28] pero principalmente incluyendo el uso de OAuth2 y TLS.

5. Conclusiones

Este trabajo se motivó en una necesidad actual acuciante: asegurar los sistemas de IoT, especialmente aquellos ligados a un contexto hogareño, y hacerlo sin menoscabar la libertad de acceso del usuario para recopilar la información y para modificar las configuraciones de su sistema. Esta necesidad, como se vio, parte entre otras cosas de la relativa informalidad del IoT, especialmente en un entorno de casa inteligente.

Se inició con una concepción metodológica que estratifica el entorno en capas: aquella relacionada más de cerca con la interacción con el espacio mediante recolección de información y ejecución de acciones para modificar el microambiente, y la capa de procesamiento centralizado y análisis de la información. Ambas capas se interrelacionaron con una conexión centralizada que las unifica manteniendo también su acoplamiento ligero.

Esta metodología se orientó a permitir una implementación basada en microservicios, donde, además de evitar en lo posible cualquier tipo de estructura monolítica, se proporcionó un grupo de servicios y bibliotecas para clientes que facilitan en buena medida la generación de nuevas utilidades y componentes. Las pruebas realizadas con los servicios, clientes y sensores generados bajo esta infraestructura permitieron confirmar tanto la solidez como la relativa facilidad de uso de los componentes.

El punto principal, la seguridad, si bien no fue el más fácil de implementar, una vez depurados los tres esquemas definidos para los diferentes tipos de conexiones, demostró ser una elección robusta, que resistió las pruebas de acceso indebido. Cabe indicar, sin embargo, que un esquema metódico de pruebas específico para el campo resta aún por diseñarse y aplicarse, lo cual será motivo de trabajos futuros. Podemos, empero, afirmar que la combinación TLS, OAuth2, MQTT, produjo los resultados esperados en gran medida.

Como principales contribuciones tenemos la implementación de una sólida arquitectura segura de IoT para el hogar; la combinación estable y fluida de al menos tres tecnologías de alto nivel para el manejo de seguridad y una implementación de referencia funcional que podrá ser puesta disposición públicamente para su libre uso.

Referencias

[1] Y. Lu and L. D. Xu, "Internet of things (IoT) cybersecurity research: A review of current research topics," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2103–2115, 2019. [Online]. Available: https://doi.org/10.1109/JIOT.2018.2869847

[2] A. Riahi Sfar, E. Natalizio, Y. Challal, and Z. Chtourou, "A roadmap for security challenges in the internet of things," *Digital Communications and Networks*, vol. 4, no. 2, pp. 118–137, 2018. [Online]. Available: https://doi.org/10.1016/j.dcan.2017.04.003

[3] P. Lea, *Internet of Things for Architects: Architecting IoT solutions by implementing sensors, communication infrastructure, edge computing,*

analytics, and security. Packt Publishing Ltd, 2018. [Online]. Available: https://bit.ly/3oJ1XRl

[4] P. Jamshidi, C. Pahl, N. C. Mendonça, J. Lewis, and S. Tilkov, "Microservices: The journey so far and challenges ahead," *IEEE Software*, vol. 35, no. 3, pp. 24–35, 2018. [Online]. Available: https://doi.org/10.1109/MS.2018.2141039

[5] J. Khan, J. p. Li, I. Ali, S. Parveen, G. a. Khan, M. Khalil, A. Khan, A. U. Haq, and M. Shahid, "An authentication technique based on oauth 2.0 protocol for internet of things (IoT) network," in *2018 15th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, 2018, pp. 160–165. [Online]. Available: https://doi.org/10.1109/ICCWAMTIP.2018.8632587

[6] C. Chan, R. Fontugne, K. Cho, and S. Goto, "Monitoring tls adoption using backbone and edge traffic," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2018, pp. 208–213. [Online]. Available: https://doi.org/10.1109/INFCOMW.2018.8406957

[7] F. Izquierdo, M. Ciurana, F. Barcelo, J. Paradells, and E. Zola, "Performance evaluation of a TOA-based trilateration method to locate terminals in WLAN," in *2006 1st International Symposium on Wireless Pervasive Computing*, 2006, pp. 1–6. [Online]. Available: https://doi.org/10.1109/ISWPC.2006.1613598

[8] M. A. Khan and K. Salah, "IoT security: Review, blockchain solutions, and open challenges," *Future Generation Computer Systems*, vol. 82, pp. 395–411, 2018. [Online]. Available: https://doi.org/10.1016/j.future.2017.11.022

[9] J. P. Rojas, J. C. Bustos, and D. Ordóñez Camacho, "Smart public transportation at your fingertips," *Enfoque UTE*, vol. 8, no. 1, pp. 122–134, Feb. 2017. [Online]. Available: https://doi.org/10.29019/enfoqueute.v8n1.143

[10] J. P. Rojas, J. C. Bustos, and D. Ordóñez-Camacho, "Qbus: Movilidad inteligente para el usuario de transporte público," in *Proceedings of the International Conference on Information Systems and Computer Science, INCISCOS 2016*, 2016. [Online]. Available: https://bit.ly/3jZlBpE

[11] E. A. Q. Montoya, S. F. J. Colorado, W. Y. C. Muñoz, and G. E. C. Golondrino, "Propuesta de una arquitectura para agricultura de precisión soportada en IoT," *RISTI - Revista Iberica de Sistemas e Tecnologias de Informacao*, pp. 39–56, 2017. [Online]. Available: http://dx.doi.org/10.17013/risti.24.39-56

[12] M. Agiwal, N. Saxena, and A. Roy, "Towards connected living: 5g enabled internet of things (IoT)," *IETE Technical Review*, vol. 36, no. 2, pp. 190–202, 2019. [Online]. Available: https://doi.org/10.1080/02564602.2018.1444516

[13] H. Lin and N. Bergmann, "IoT privacy and security challenges for smart home environments," *Information*, vol. 7, no. 3, p. 44, Jul 2016. [Online]. Available: http://dx.doi.org/10.3390/info7030044

[14] H. Kaffel-Ben Ayed, H. Boujezza, and I. Riabi, "An idms approach towards privacy and new requirements in IoT," in *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*, 2017, pp. 429–434. [Online]. Available: https://doi.org/10.1109/IWCMC.2017.7986324

[15] F. Fernández, A. Alonso, L. Marco, and J. Salvachúa, "A model to enable application-scoped access control as a service for IoT using OAuth 2.0," in *2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)*, 2017, pp. 322–324. [Online]. Available: https://doi.org/10.1109/ICIN.2017.7899433

[16] J. Bugeja, A. Jacobsson, and P. Davidsson, "On privacy and security challenges in smart connected homes," in *2016 European Intelligence and Security Informatics Conference (EISIC)*, 2016, pp. 172–175. [Online]. Available: https://doi.org/10.1109/EISIC.2016.044

[17] L. Sun, Y. Li, and R. A. Memon, "An open IoT framework based on microservices architecture," *China Communications*, vol. 14, no. 2, pp. 154–162, 2017. [Online]. Available: https://doi.org/10.1109/CC.2017.7868163

[18] T. Vresk and I. Çavrak, "Architecture of an interoperable IoT platform based on microservices," in *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2016, pp. 1196–1201. [Online]. Available: https://doi.org/10.1109/MIPRO.2016.7522321

[19] R. Yu, V. T. Kilari, G. Xue, and D. Yang, "Load balancing for interdependent IoT microservices," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 2019, pp. 298–306. [Online]. Available: https://doi.org/10.1109/INFOCOM.2019.8737450

[20] D. Díaz-Sánchez, A. Marín-Lopez, F. A. Mendoza, P. A. Cabarcos, and R. S. Sherratt, "TLS/PKI challenges and certificate pinning techniques for IoT and M2M secure communications," *IEEE Communications Surveys Tutorials*, vol. 21, no. 4, pp. 3502–3531, 2019. [Online]. Available: https://doi.org/10.1109/COMST.2019.2914453

[21] P. Urien, "Securing the IoT with TLS/DTLS server stacks embedded in secure elements: An ePlug usecase," in *2017 14th IEEE Annual Consumer Communications Networking Conference (CCNC)*, 2017, pp. 569–570. [Online]. Available: https://doi.org/10.1109/CCNC.2017.7983170

[22] J. D. Hoz, J. Saldana, J. Fernández-Navajas, J. Ruiz-Mas, R. G. Rodríguez, and F. d. J. M. Luna, "SSH as an alternative to TLS in IoT environments using HTTP," in *2018 Global Internet of Things Summit (GIoTS)*, 2018, pp. 1–6. [Online]. Available: https://doi.org/10.1109/GIOTS.2018.8534545

[23] M. Khan, M. W. Anwar, F. Azam, F. Samea, and M. F. Shinwari, *A Model-Driven Approach for Access Control in Internet of Things (IoT) Applications – An Introduction to UMLOA*. Communications in Computer and Information Science, Springer, 2018, vol. 920. [Online]. Available: https://doi.org/10.1007/978-3-319-99972-2_16

[24] H. Kim, A. Wasicek, B. Mehne, and E. A. Lee, "A secure network architecture for the internet of things based on local authorization entities," in *2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud)*, 2016, pp. 114–122. [Online]. Available: https://doi.org/10.1109/FiCloud.2016.24

[25] M. Pahl and L. Donini, "Securing IoT microservices with certificates," in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, 2018, pp. 1–5. [Online]. Available: https://doi.org/10.1109/NOMS.2018.8406189

[26] S. Sciancalepore, G. Piro, D. Caldarola, G. Boggia, and G. Bianchi, "Oauth-iot: An access control framework for the internet of things based on open standards," in *2017 IEEE Symposium on Computers and Communications (ISCC)*, 2017, pp. 676–681. [Online]. Available: https://doi.org/10.1109/ISCC.2017.8024606

[27] S. Shapsough, F. Aloul, and I. A. Zualkernan, "Securing low-resource edge devices for IoT systems," in *2018 International Symposium in Sensing and Instrumentation in IoT Era (ISSI)*, 2018, pp. 1–4. [Online]. Available: https://doi.org/10.1109/ISSI.2018.8538135

[28] M. Singh, M. A. Rajan, V. L. Shivraj, and P. Balamuralidhar, "Secure mqtt for internet of things (IoT)," in *2015 Fifth International Conference on Communication Systems and Network Technologies*, 2015, pp. 746–751. [Online]. Available: https://doi.org/10.1109/CSNT.2015.16

[29] C. Singh and M. Kumar, *Mastering Hadoop 3: Big data processing at scale to unlock unique business insights*. Packt Publishing, 2019. [Online]. Available: https://bit.ly/37Qi2O9

[30] J. Turnbull, *The Docker Book: Containerization is the new virtualization*, 2014. [Online]. Available: https://bit.ly/3m7nqRY

[31] A. Selva. (2014) Java MQTT lightweight broker. moquette. [Online]. Available: https://bit.ly/3gB82Mw

[32] M. Bhushan, *Big Data and Hadoop: Learn by Example*. BPB Publications, 2018. [Online]. Available: https://bit.ly/2W0AmP1

[33] T. Dunning and E. Friedman, *Time Series Databases: New Ways to Store and Access Data, Edition: 1. Sebastopol*. O'Reilly Media, Inc, 2014. [Online]. Available: https://bit.ly/2W1VnsU

[34] B. Brazil, *Prometheus: Up & Running: Infrastructure and Application Performance Monitoring*. O'Reilly Media, 2018. [Online]. Available: https://bit.ly/39V80xX

[35] A. Kurniawan, *Arduino MKR WIFI 1010 Development Workshop*. PE Press, 2018. [Online]. Available: https://bit.ly/37OEnvD

[36] I. Dogan and I. Ahmet, *The Official ESP32 Book*. Elektor International Media, 2017. [Online]. Available: https://bit.ly/2IzEW3G

[37] G. C. Hillar, *Hands-On MQTT Programming with Python: Work with the lightweight IoT protocol in Python*. Packt Publishing, 2018. [Online]. Available: https://bit.ly/33YpdTg

[38] B. Charles, *Beginning Sensor Networks with Arduino and Raspberry Pi*. Apress, 2013. [Online]. Available: https://bit.ly/3m5syGj