Securización de un entorno de telemetría IoT

Autor: Enrique Arias Martínez
Tutor: Joan Caparrós Ramírez
Directora: Helena Rifá Pous
Máster Universitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones
Área Seguridad en redes y sistemas

15/01/2021

Créditos/Copyright

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Securización de un entorno de telemetría IoT*
Nombre del autor:	*Enrique Arias Martínez*
Nombre del colaborador/a docente :	*Joan Caparrós Ramírez*
Nombre del PRA:	*Helena Rifà Pou*
Fecha de entrega (mm/aaaa):	*12/2020*
Titulación o programa:	*Máster Universitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones*
Área del Trabajo Final:	*Seguridad de la Internet de las cosas*
Idioma del trabajo:	*Español*
Palabras clave	*IoT, LoRa, LoRaWAN, Securización*

Resumen del Trabajo:

Cada día esta más presente en nuestras vidas los dispositivos IoT que nos permiten nuevas formas de entender y conocer el mundo que nos rodea. Este crecimiento ha ido de la mano en la forma en que nuevos ataques se dirigen hacia estos dispositivos y las infraestructuras que les dan soporte. Esto ha hecho que pongamos el foco en la necesidad de fomentar la seguridad, no solo en el ámbito tecnológico que esta suficientemente desarrollado sino en la forma de su aplicación practica y las buenas practicas que debemos seguir en todo momento a la hora de securizar entornos de dispositivos IoT.

Esto nos lleva a tener que realizar un estudio y el correspondiente análisis de las tecnologías, equipamientos, entornos de programación y demás elementos que son necesarios en el despliegue de redes de sensores basados en dispositivos IoT, incluyendo los aspectos más bajo de la seguridad como es la securización física de los dispositivos.

Abstract:

Every day IoT devices are more present in our lives that allow us new ways of understanding and knowing the world around us. This growth has gone hand in hand with the way new attacks are targeting these devices and the infrastructures that support them. This has made us focus on the need to promote security, not only in the technological field that is sufficiently developed but in the form of its practical application and the good practices that we must follow at all times when securing environments of IoT devices.

This leads us to have to carry out a study and the corresponding analysis of the technologies, equipment, programming environments and other elements that are necessary in the deployment of sensor networks based on IoT devices, including the lowest aspects of security such as the physical security of the devices.

Dedicatoria

Quiero dedicar este Trabajo Final de Máster a mi mujer y compañera en mi vida Anabel Rodríguez Ferreiro, que tanto me ha apoyado y ayudado para conseguir superar mis retos en esta vida.

Resumen

Cada día esta más presente en nuestras vidas los dispositivos IoT que nos permiten nuevas formas de entender y conocer el mundo que nos rodea. Este crecimiento ha ido de la mano en la forma en que nuevos ataques se dirigen hacia estos dispositivos y las infraestructuras que les dan soporte. Esto ha hecho que pongamos el foco en la necesidad de fomentar la seguridad, no solo en el ámbito tecnológico que esta suficientemente desarrollado sino en la forma de su aplicación practica y las buenas practicas que debemos seguir en todo momento a la hora de securizar entornos de dispositivos IoT.

Esto nos lleva a tener que realizar un estudio y el correspondiente análisis de las tecnologías, equipamientos, entornos de programación y demás elementos que son necesarios en el despliegue de redes de sensores basados en dispositivos IoT, incluyendo los aspectos más bajo de la seguridad como es la securización física de los dispositivos.

Abstract

Every day IoT devices are more present in our lives that allow us new ways of understanding and knowing the world around us. This growth has gone hand in hand with the way new attacks are targeting these devices and the infrastructures that support them. This has made us focus on the need to promote security, not only in the technological field that is sufficiently developed but in the form of its practical application and the good practices that we must follow at all times when securing environments of IoT devices.

This leads us to have to carry out a study and the corresponding analysis of the technologies, equipment, programming environments and other elements that are necessary in the deployment of sensor networks based on IoT devices, including the lowest aspects of security such as the physical security of the devices.

Palabras clave

Trabajo de Final de Máster

Índice

Índice de figuras

Índice de tablas

1. Introducción

1.1. Introducción/Prefacio

Actualmente se vive una revolución en el mundo de la TIC con la explosión de los entornos IoT. La visón actual de la computación a gran escala y centralizada, esta dando paso a un nuevo paradigma donde el despliegue de miles o millones dispositivos interconectados, que nos permite tener una nueva perspectiva del uso de la información y de los datos no solo para el procesamiento empresarial centralizado y exclusivo, sino para nuevas maneras de entender la tecnología y su uso en los lugares menos inesperados.

El mundo de los dispositivos IoT, esta sufriendo una gran expansión, sobre todo con los nuevos concepto de SmartCities, SmartEnergy y otros que surgirán en el abrigo de estos entornos tecnológicos, que tienen como finalidad ayudarnos en el día a día de una manera más eficiente y rápida. Así, soluciones de IoT para la gestión de aparcamiento de ciudades, sistemas de recogida de basuras eficientes, control de trafico inteligente, etc., están muy orientadas a miles de dispositivos que recogen información para ser procesada y tratada en tiempo real, ayudando a entidades gubernamentales y empresas de sectores estratégicos como las empresas energéticas a la gestión de entornos descentralizados.

Pero también hay otros mundos aparte de los antes mencionados, que están observando los entornos IoT como una solución ideal para sus problemas actuales que no están dentro del radar de los grandes desarrolladores de soluciones informáticas.

De esta manera, el crecimiento de las dispositivos basados en *Open-Source hardware* dentro del mundo IoT, esta permitiendo disponer de desarrollos relativamente baratos que pueden permitir a pequeñas pymes implantar soluciones de sensores con un esfuerzo económico muy pequeño.

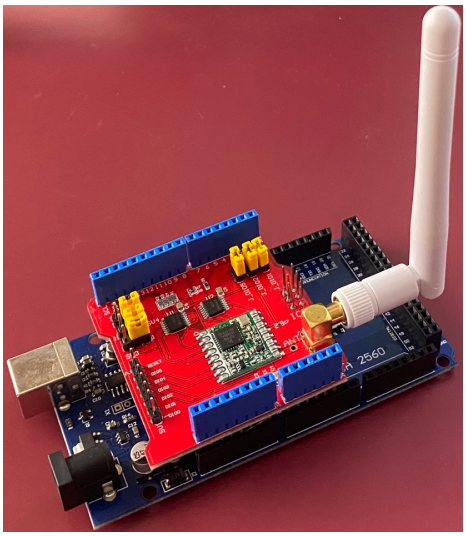


Figura 1: Arduino con módulo LoRa@ [Elaboración propia]

En el mundo IoT nos encontramos una gran gama de productos comerciales donde una gran parte de estos dispositivos vienen de integradores muy arraigados del mundo tecnológico de las TIC y también nos encontramos un nicho de fabricantes emergentes que han visto en los entornos IoT un gran futuro. Por otra parte nos encontramos con el mundo del desarrollo *Open-Source* de IoT, donde una gran comunidad de usuarios está poniendo mucho empeño en hacer asequible esta tecnología para usuarios finales y/o pequeñas empresas con desarrolladores modestos. También hay una gran cantidad de fabricantes de equipamiento electrónico con diseños sencillos que sean orientados a esta comunidad de desarrolladores con diseños de hardware sencillo pero muy versátiles que se integran con el mundo *Open-Source software* de IoT.

El problema que vamos a tratar en este trabajo fin de máster es la aplicación de la seguridad integrada en una red de sensores de telemetría IoT viendo las implicaciones tanto teóricas como prácticas necesarias para alcanzar un nivel alto de seguridad en la implementación de estos sistemas.

Hemos visto recientemente el público en los medios de comunicación noticias referente a ciberataques orientados a dispositivos IoT, estos dispositivos aunque sencillos puede ser una fuente de problemas y vulnerabilidades qué hacen que los proyectos en los que están involucrados dichos elementos corran serios peligros.

Nos centraremos en cómo se debe aplicar la seguridad tanto a nivel hardware como en la capa de software de los dispositivos, elementos de comunicación, aplicativos y elementos que conforman una nube privada de gestión IoT.

Nuestro objetivo es conseguir una arquitectura de una red de telemetría IoT robusta ante los intentos de penetración, robo o manipulación de datos y denegación de servicio de los sistemas. Revisaremos en cada una de las capas de nuestra arquitectura IoT, para saber cómo se debe aplicar la seguridad correctamente, que vulnerabilidades podemos esperar que se produzca y qué contramedidas deberemos a tomar para garantizar tener una arquitectura segura.

1.2. Descripción/Definición

Para el desarrollo de este proyecto, se plantea un supuesto practico, donde un Ayuntamiento de una comarca ganadera y con la ayuda de una empresa del ramo de las TIC, desea implementar una solución IoT para el control de las reses de los ganaderos locales que se encuentran pastando por diferentes fincas que cubre una gran superficie del termino municipal, con una red de telemetría de sensores IoT y con la garantía de que se cumplen con los más alto estándares de seguridad informática.

El Ayuntamiento quiere ofrecer este servicio a los ganaderos del termino municipal que se enfrentan a la necesidad de controlar sus manadas de animales, no solo por las perdidas que puede ocasionar la perdida de un res, sino también al peligro que supone que los animales escapen e invadan las carreteras con el consiguiente riesgos para los conductores, o las calles del los diferentes núcleos urbanos de población que componen el termino municipal (pedanías), con los problemas legales que pueden ocasionar a los ganaderos.

En la actualidad los ganaderos, utilizan “pastores eléctricos” o cercados electrificados para evitar que el ganado salga de sus limites pero de un resultado muy escaso, dado que requiere la instalación de la cintas de vallado este correctamente realizado para evitar su caída y de los elementos de generación eléctrica, como baterías, que deben ser cargadas regularmente y con el problema de estar sometidas a las inclemencias del tiempo. Aparte muchas de estas fincas son de una gran extensión por lo que se hace inviable el cercado de las mismas.

Se desea desarrollar una solución IoT, que con un reducido coste, fácil despliegue y una gran seguridad permita a los ganaderos para el seguimiento en tiempo real de sus reses, que puedan recibir alertas cuando alguna de ellas salga de los limites establecidos así como saber a las inclemencias del tiempo que se enfrenta estos animales.

El Ayuntamiento a su vez, lograra validar e implementar una red IoT, sin asumir un coste excesivo, que servirá como base para futuros proyectos tecnológicos como la recogida selectiva de basura, donde una red de sensors indicaran cuando los contenedores de basura y reciclaje de las distintas pedanías del termino municipal están en el punto optimo de llenado para su recogida selectiva, una red de teleasistencia a personas mayores para caso de emergencias, donde un pequeño sensor con un botón de pánico y un acelerómetro que detecta las caídas, ayudara a la seguridad de este núcleo de población vulnerable, entre muchos otros proyectos IoT que se pueden desarrollar en los entornos rurales.

1.3. Objetivos

El objetivo principal de este protecto es abordar el desarrollo de una red de sensores IoT con bajo presupuesto pero con alto grado de seguridad aplicada a los diferentes componentes y/o entornos que lo conforman.

1.3.1. Objetivos principales

Se han definido los siguientes objetivos principales para este proyecto:

- Implementación segura de un sensor de coste reducido capaz de enviar su geolocalización y temperatura ambiente.
- Configurar una pasarela LoRaWAN que reciba la información del sensor y la reenvié a los sistemas de gestión y securizarla convenientemente.
- Configurar un entorno de contenerización para la recepción de los datos del sensor y presentarlos en una pagina Web de una forma segura.
- Configurar una red de servicio segura mediante el empleo de cortafuego de nueva generación.
- Implementar un acceso VPN a la base de datos de los sensores con factor de doble autentificación.

Objetivos personales del autor del TF:

- Ahondar en los conocimientos de la seguridad aplicada al hardware IoT.
- Estudiar la seguridad de los entornos de contenerización de aplicaciones Docker.

1.4. Metodología y proceso de trabajo

En este proyecto realizaremos una investigación aplicada de las necesidades y formas de securización de los entornos IoT propuestos en este trabajo donde adaptaremos productos existentes, para generar un entorno seguro de una manera rápida y versátil.

Realizaremos una metodología en cascada, con el fin de explorar los aspectos actuales de la seguridad, así como su experimentación practica en los elementos de comunicación, software e infraestructuras que conforman el supuesto practico expuesto en este proyecto. El desarrollo en cascada se ajusta a los requerimientos del proyecto, eliminando únicamente la fase de mantenimiento que no esta dentro del alcance del mismo.

Las fases de esta metodología las detallamos a continuación.

- Fase de análisis: Durante la fase de análisis, realizaremos una investigación de la seguridad en el entorno del proyecto que se desea securizar, revisando literatura de proyectos similares a través de búsquedas en Internet principalmente.
- Fase de diseño: En esta fase realizaremos la estructura de la arquitectura de la red de sensores, las relaciones entre los diferentes sistemas, especificando la funcionalidad de cada una de ellas y buscando las cualidades de seguridad más robustas.
- Fase de implementación: Durante esta fase, se realizará la implementación de la arquitectura especificada en la fase anterior según los requisitos indicados del proyecto.

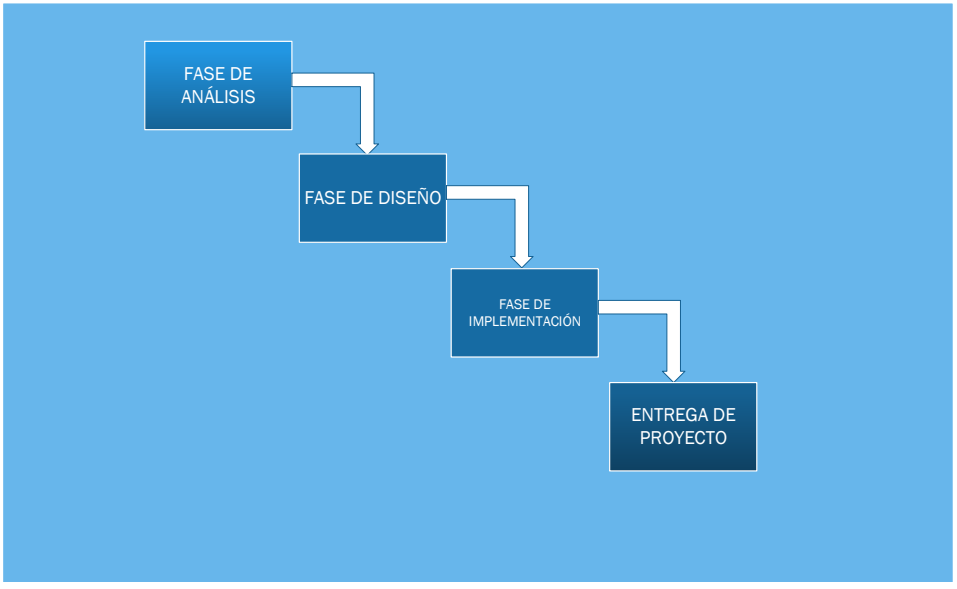


Figura 2: Metodología del TFM [1]

Todo esto nos deberá llevar a respondernos una pregunta que siempre nos debe estar realizando en el mundo de la seguridad de las TIC, ¿Trabajo con un entorno seguro antes ataques y amenazas?.

1.5. Planificación

Para la realización de nuestra planificación, nos basaremos en el numero de créditos de esta asignatura, que son 9 y el números de horas estipuladas por cada crédito que son 25 horas.

PEC 2: (4,2 créditos)

Fase de análisis de seguridad IoT (2,4 créditos)

Subtarea: Analizar el diseño seguro de hardware IoT

Tiempo previsto: 15 horas (0,6 créditos)

Objetivo: Investigar sobre las tecnológicas de seguridad aplicadas al hardware

Nivel previo de conocimientos: Amplios conocimientos de electrónica aplicada.

Subtarea: Analizar la seguridad en LoRaWAN

Tiempo previsto: 15 horas (0,6 créditos)

Objetivo: Estudiar la seguridad de la tecnológica LoRaWAN.

Nivel previo de conocimientos: Conocimientos altos de la tecnológica LoRaWAN.

Subtarea: Analizar la securización de la contenerización de aplicaciones

Tiempo previsto: 15 horas (0,6 créditos)

Objetivo: Estudiar la seguridad de la contenerización de aplicaciones.

Nivel previo de conocimientos: Conocimientos básicos de Docker.

Subtarea: Analizar la seguridad en Internet

Tiempo previsto: 15 horas (0,6 créditos)

Objetivo: Analizar la seguridad de las tecnológicas de seguridad en Internet.

Nivel previo de conocimientos: Experto en seguridad TIC.

Fase de diseño de red segura IoT (1,8 créditos)

Subtarea: Diseño seguro de sensores IoT

Tiempo previsto: 10 horas (0,4 créditos)

Objetivo: Analizar y evaluar la seguridad en el hardware

Nivel previo de conocimientos: Conocimiento de seguridad lógica y física.

Subtarea: Securización de red LoRaWAN

Tiempo previsto: 10 horas (0,4 créditos)

Objetivo: Aplicar la seguridad en la tecnológica LoRaWAN.

Nivel previo de conocimientos: Conocimientos altos de protocolos de seguridad.

Subtarea: Securización en contenerización de aplicaciones

Tiempo previsto: 15 horas (0,6 créditos)

Objetivo: Analizar la seguridad de la tecnológica LoRaWAN.

Nivel previo de conocimientos: Conocimientos básicos de la seguridad en Docker.

Subtarea: Securización de servicios en Internet

Tiempo previsto: 10 horas (0,4 créditos)

Objetivo: Definir la seguridad de las tecnológicas de seguridad en Internet.

Nivel previo de conocimientos: Amplios conocimientos en el diseño de arquitecturas de seguridad TIC.

PEC 3: (4,8 créditos)

Fase de inmplementación de la seguridad IoT (4,8 créditos)

Subtarea: Despliegue de sensores IoT

Tiempo previsto: 25 horas (1 créditos)

Objetivo: Realizar la securización de los sensores IoT

Nivel previo de conocimientos: Amplios conocimientos de electrónica aplicada.

Subtarea: Configuración segura de pasarela LoRaWAN

Tiempo previsto: 20 horas (0,8 créditos)

Objetivo: Configurar pasarela LoRaWAN de forma segura.

Nivel previo de conocimientos: Conocimientos altos de la tecnológica LoRaWAN.

Subtarea: Despliegue seguro de entorno Docker

Tiempo previsto: 35 horas (1,4 créditos)

Objetivo: Securizar entorno de aplicaciones Docker.

Nivel previo de conocimientos: Conocimientos básicos de Docker.

Subtarea: Implementación segura de NGFW en Internet

Tiempo previsto: 15 horas (0,6 créditos)

Objetivo: Realizar la configuración segura del NGFW.

Nivel previo de conocimientos: Experto en seguridad TIC.

La planificación prevista para el desarrollo del proyecto vista desde un diagrama de Gantt es la siguiente:

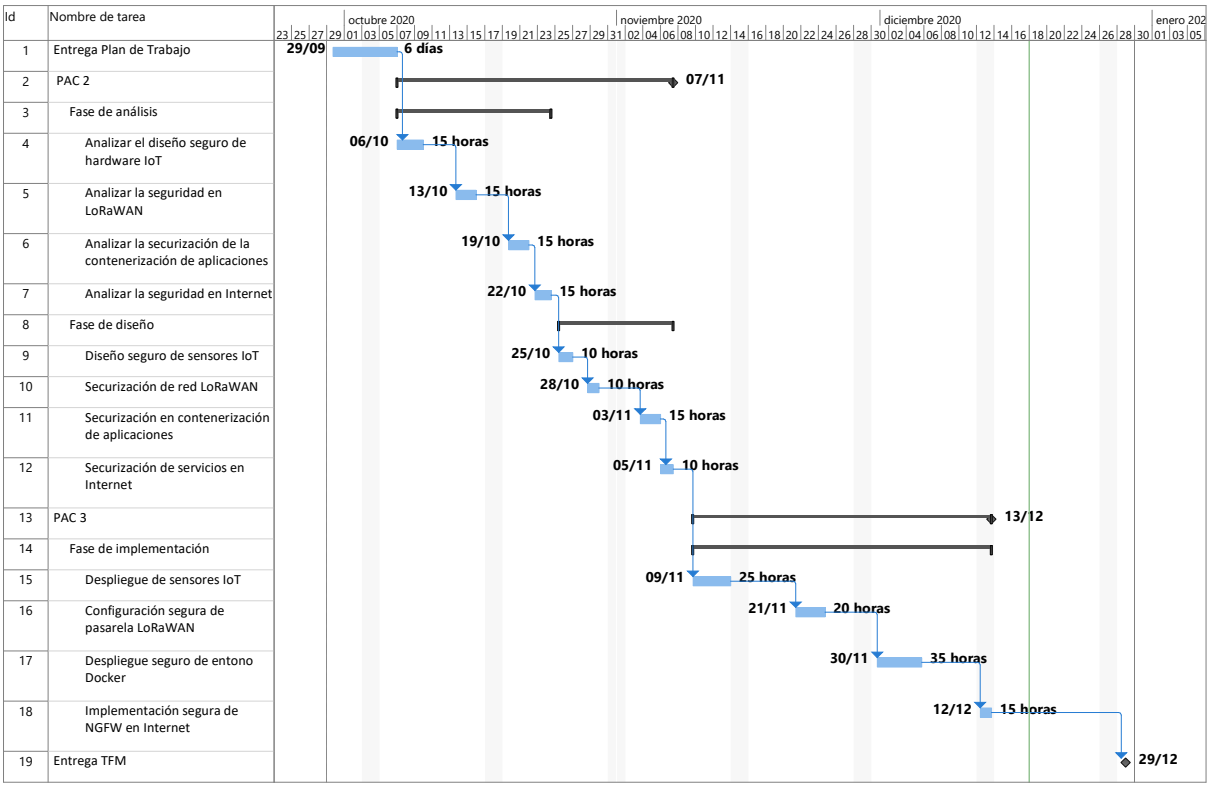


Figura 3: Diagrama de Gantt

1.5.1. Gestión del riesgo

Durante las diferentes fases del proyecto nos podemos encontrar diversos riesgos que tenemos que gestionar para hacerles frente y minimizarlos para que el desarrollo del proyecto no se vea afectado. Indicaremos los riesgos del proyecto, su nivel de severidad, la posibilidad de que ocurra así como las medidas de salvaguardas para minimizar el riesgo.

Riesgos en la fase de análisis de seguridad IoT

Subtarea: Analizar el diseño seguro de hardware IoT

Riegos asociados: Amplitud de información existente en el ámbito de los procesadores aplicados al IoT.

Riego / Probabilidad: Bajo / Baja.

Mitigación riesgos: Reducir ámbito de investigación.

Subtarea: Analizar la seguridad en LoRaWAN

Riegos asociados: Exceso de entornos involucrados.

Riego / Probabilidad: Bajo / Baja.

Mitigación riesgos: Acotar alcance de la investigación a la seguridad del protocolo.

Subtarea: Analizar la securización de la contenerización de aplicaciones

Riegos asociados: Dificultad en la resolución de dudas y problemas.

Riego / Probabilidad: Bajo / Media.

Mitigación riesgos: Consultas con expertos en tecnología de contenerización.

Subtarea: Analizar la seguridad en Internet

Riegos asociados: Exceso de detalle en los conceptos.

Riego / Probabilidad: Bajo / Media.

Mitigación riesgos: Concentrar y resumir los conocimientos.

Riesgos en la dase de diseño de red segura IoT

Subtarea: Diseño seguro de sensores IoT

Riegos asociados: Múltiples análisis de cruzados de información.

Riego / Probabilidad: Bajo / Baja.

Mitigación riesgos: Acotar el rango de análisis.

Subtarea: Securización de red LoRaWAN

Riegos asociados: Complejidad de la solución.

Riego / Probabilidad: Bajo / Media.

Mitigación riesgos: Establecer criterios sencillos y concretos.

Subtarea: Securización en contenerización de aplicaciones

Riegos asociados: Necesidad de ampliación de conocimientos sobre la tecnologia .

Riego / Probabilidad: Medio / Alta

Mitigación riesgos: Derivar tiempo de otras tareas más avanzadas hacia esta.

Subtarea: Securización de servicios en Internet

Riegos asociados: Exceso de detalle en los conceptos.

Riego / Probabilidad: Bajo / Baja.

Mitigación riesgos: Concentrar y resumir los conocimientos.

Riesgos en la fase de inmplementación de la seguridad IoT

Subtarea: Despliegue de sensores IoT

Riegos asociados: Exceso de tiempo de programación .

Riego / Probabilidad: Alto / Media.

Mitigación riesgos: Reducir la programación al ámbito de la seguridad.

Subtarea: Configuración segura de pasarela LoRaWAN

Riegos asociados: Problemas en la comunicación entre plataformas.

Riego / Probabilidad: Medio / Media.

Mitigación riesgos: Establecer enfoque en los protocolos de securización .

Subtarea: Despliegue seguro de entorno Docker

Riegos asociados: Problemas en el despliegue y configuración de Docker.

Riego / Probabilidad: Alto / Alta.

Mitigación riesgos: Centrarse en la securización de la plataforma.

Subtarea: Implementación segura de NGFW en Internet

Riegos asociados: No disponer de todos los sistemas asociados.

Riego / Probabilidad: Alto / Baja.

Mitigación riesgos: Realizar las configuraciones sobre el modelo teórico.

Podemos observar dentro de este análisis, que el punto más critico en el proyecto es la tarea "*Despliegue seguro de entorno Docker"*, donde observamos que es un riesgo alto y hay una gran probabilidad de que ocurra, por lo que tenemos que estar muy pendientes de esta tarea y quizás desviar recursos de otras tareas más adelantadas para desviarlo a esta tarea y poder minimizar el riesgo.

1.6. Presupuesto

A continuación detallamos el presupuesto de una red de telemetría IoT, que incluirá todos los elementos necesarios para el desarrollo del proyecto propuesto en este proyecto.

Descripción	Uds.	Precio	Total
Modulo IoT + GPS + LoRa@ **Hardware IoT basado en SoC ESP32, con modulo GPS integrado, modulo de transmisión LoRa@ en banda 868 Mhz., bus de datos para recopilación de datos telemétricos**	5	21,22 €	106,1 €
Sensor AM2320 (Temperatura + Humedad) **Sensor de temperatura y humedad de bajo consumo , bus I2C, tensión de trabajo de 3 a 5 voltios, 2.5 mA de consumo, rango de temperatura de -40 ^{0}C a 80 ^{0}C y rango de humedad entre 0 a 100%.**	5	1,27 €	6,2 €
Gateway LoRaWan Dragino LG308 **Gateway de 10+1 canal basado en 1 modulo SX1301 y 2 módulos SX1257 LoRa y sistema operativo Linux Openwrt. Modem 4G/LTE incorporado, 2 puertos FastEthernet y Wifi. Soporta de 1 a 1000 sensores.**	1	198 €	198 €
QNAP TS-251-B con 3 TB de capacidad **NAS (Network Access Server) con procesador Intel Celeron J3335 de doble núcleo de 2.0 Ghz y 8 Gb de memoria RAM, soporte RAID 0,1 y 10, cifrado hardware AES-NI, sistema operativo Linux 4.14.24-qnap, 1 puerto Gigabit Ethernet, 3 puertos USB 3.0, soporte nativo Docker y LCX.**	1	375 €	375 €
Firewall FortiGate 50E **Cortafuego de capa 7 con UTM integrado, 7 puertos Gigabit Ethernet, hasta 2,5 Gbps de procesamiento de trafico, hasta 220 Mbps de inspección de trafico UTM, 2 Licencias FortiToken (OTP) para VPN SSL incluidas.**	1	440 €	440 €
Tarifa Internet 1 año (3 €/meses/2 Gb) Gateway LoRaWan	1	36 €	36 €
Conexión Internet FFTH 600 Mbps (coste anual)	1	480 €	480 €

Tabla 1: Presupuesto material

1.7. Estado del Arte

La seguridad en el mundo IoT es unos de los grandes problemas que se encuentran las empresas tecnológicas qué se dedican a desplegar este tipo de infraestructuras. La seguridad tiene que ser un ciclo continuo en el mundo IoT que debe involucrar desde el desarrollo de los sensores, pasando por la transmisión de los datos, su almacenamiento y su posterior presentación grafica a los usuarios. Por eso hay que ser muy conscientes de la necesidad de securizar estos entornos para evitar cualquier posible brecha de seguridad tanto en los dispositivos hardware, las comunicaciones, así como en los aplicativos, de lo que llamamos un Ecosistema IoT.

Ya la Unión Europea, a través del Grupo de Trabajo "**El avance de la Internet de las cosas en Europa**" [2] hace una mención muy acertada sobre los aspectos de la seguridad: "*Los problemas de seguridad y protección de los datos personales son una preocupación clave para una implementación exitosa de IoT. Si bien la implementación de IoT está en su infancia, varios ejemplos recientes de piratería de objetos han demostrado que la cantidad de ataques aumentará exponencialmente si persisten las vulnerabilidades conocidas a medida que los objetos conectados se utilizan cada vez más. Un tema importante tiene que ver con la autenticación segura. Los dispositivos en red que intercambian datos con otros dispositivos de IoT deben estar debidamente autenticados para evitar problemas de seguridad. Esto puede necesitar incluir ciertos protocolos de autenticación y utilizar canales de comunicación encriptados o de integridad segura. Los servicios de la Comisión consideran importante reflexionar sobre las posibilidades de certificación de dispositivos en red que proporcionarían un nivel mínimo de autenticación segura, desde el nivel del hardware hasta la integridad de la red. Esto implicaría algún análisis de las funciones con las que está equipado cada dispositivo, procesamiento de datos y conectividad seguros para los dispositivos a los que se transmiten los datos*.", con esto podemos llegar a la conclusión que en lo referente a la seguridad en el mundo IoT esta todavía por desarrollarse en gran medida.

En la actualidad hay soluciones empresariales similares [3] a la planteada en este proyecto, basadas principalmente en servicios de suscripción como SigFox[1] o NarrowBand[2], donde nos encontramos los típicos problemas de falta de cobertura en las zonas rurales profundas, dado que a los operadores de red de estas tecnologías, siguen sin garantizar una cobertura en el 100% del territorio español, por la imposibilidad de obtener algún beneficio económico o un retorno de la inversión (ROI) adecuado en estas zonas debido al alto coste asociado que llevan el despliegue de estas tecnologías del tipo propietarias de comunicación IoT.

[1] https://www.sigfox.com/en
[2] https://www.vodafone.com/business/iot/managed-iot-connectivity/nb-iot

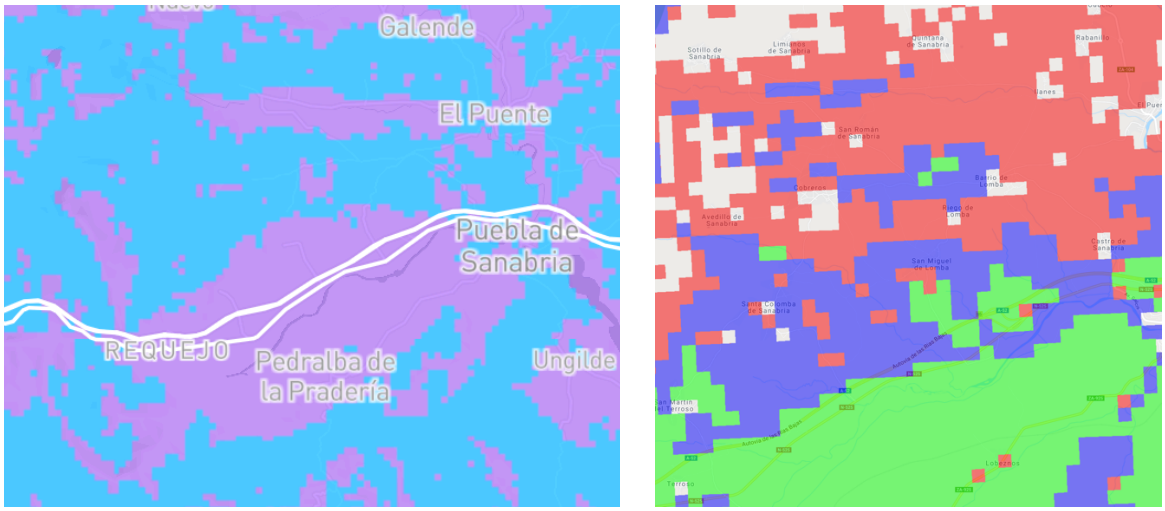


Figura 4: Cobertura SigFox [4] (izquierda) y Vodafone NarrowBand [5] (derecha) en la zona de Sanabria.

Esto es el aspecto fundamental por el cual muchos gobiernos autonómicos [6] y ayuntamientos ofrecen facilidades o subvenciones tantos económicas como de infraestructuras a sus vecinos y empresas locales para despliegues económicos de redes LPWAN, con el fin de evitar que estas zonas queden rezagadas en los avances tecnológicos del siglo XXI y sigan siendo un foco de despoblación y desindustrialización.

Por esto, en este proyecto resolveremos las preguntas de ¿Qué grado de seguridad puedo implementar en un Ecosistema IoT de coste reducido?, ¿Son seguros todos los elementos del Ecosistema IoT propuesto?.

En otro aspecto de la seguridad, entramos en el campo de las aplicaciones necesarias para operar toda la arquitectura del entorno IoT. Este campo formado sobre todo por los Network Server y Applications Server que operan dentro de un ecosistema IoT, donde existen varias vertientes de despliegue de estas infraestructuras. Tenemos las Cloud publicas IoT formadas por servicios distribuidos en Internet y disponibles para un numero ilimitado de usuarios, las Cloud privadas donde todo los sistemas son independientes y de uso exclusivo, y como es obvio existen Cloud mixtas o hibridas donde se mezclan servicios exclusivos privados como el uso de otro servicios públicos en Internet. Todos estos servicios, bien de grandes empresas, proveedores de servicios, empresas privadas o entidades gubernamentales, están normalmente formadas por una gran cantidad de aplicaciones que se ejecutan sobre servidores virtualizados para la optimización de los recursos hardware. Pero en la evolución de las tecnologías TIC esta creciendo con fuerza una nueva forma de virtualización orientada a las propias aplicaciones. Esta virtualización o “contenerización” de aplicaciones surgió como una evolución de la virtualización de los servidores, cuando la optimización de recursos hardware que se realizaba incluyendo dentro de un único servidor varias instancias de servidores virtuales se traslado a la capa de aplicación. De esta manera, se estableció que el contexto de virtualización y aislamiento no se enfocaba a la capa de abstracción del sistema operativo de servidor sino a la capa de abstracción de la aplicación que se ejecuta en el servidor. De esta manera,

un servidor ejecutando una única instancia de sistema operativo, puede ejecutar múltiples instancias de aplicaciones como si se ejecutasen en servidores independientes. Aquí surge la duda del beneficio que otorga la ejecución contenerizada de aplicaciones con la forma clásica de ejecución en modo usuario dentro del mismo servidor. La respuesta es el aislamiento [7] que nos permite la contenerización con respecto al plano de ejecución del sistema operativo anfitrión. Esto nos permite ahorrar recursos que anteriormente estarían destinados a la capa de virtualización y a cada sistema operativo virtualizado.

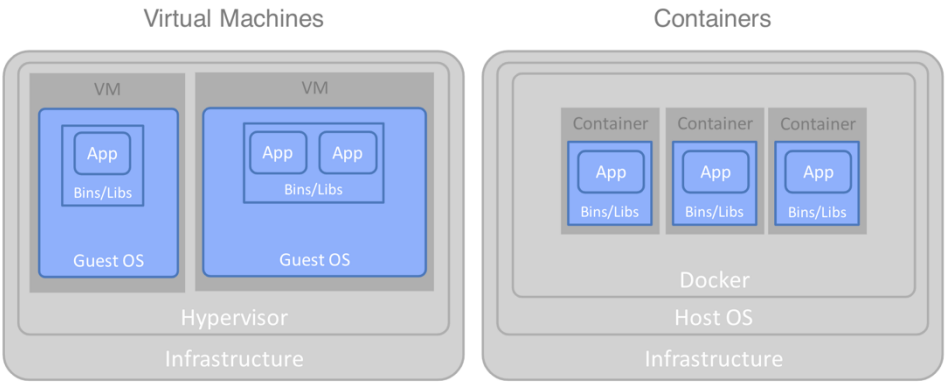


Figura 5: Comparación de arquitecturas. [8]

Otra ventaja de la contenerización es la facilidad de despliegue y de puesta en marcha de las aplicaciones. Con solo disponer de la imagen del contenedor de una aplicación, esta se puede desplegar en minutos en un servidor y estar operativa sin tener que recurrir a configuración y adaptaciones costosas en tiempo y recursos humanos. De esta manera se puede crear procesos seguros mediante el despliegue de aplicaciones contenerizadas, donde estas se arrancan automáticamente ante un evento, como por ejemplo una conexión de un usuario a una pagina Web, que provoca la puesta en marcha de un contenedor de un servidor Web, donde se realizan las tareas requeridas, como una compra on-line y que una vez terminada la sesión del usuario, se borra dicho contenedor, eliminando cualquier problema de seguridad asociado a la posibilidad de que un atacante hubiese hecho con el control del servidor Web, dado que los datos de las transacciones esta alojados en otros entornos ajenos al contenedor comprometido.

1.8. Estructura del resto del documento

En el capitulo 2 realizaremos el análisis del proyecto, sus implicaciones con respecto a la seguridad del entorno donde se va a desarrollar, bajo un supuesto practico de una red de sensores con propósito comercial.

En el capitulo 3 se llevará a cabo el diseño de la arquitectura de la red de sensores con todos los elementos que la componen, sus relaciones tanto lógicas como físicas en el ámbito de la seguridad.

En el capitulo 4 se procederá a realizar la implantación de la arquitectura diseñada en el capitulo 3, como especial mención a la seguridad en las tecnologías usadas en el proyecto.

2. Análisis de la seguridad en redes IoT

En este capitulo realizaremos un análisis detallado de los aspectos a tener en cuenta para un diseño seguro de la solución propuesta en este proyecto.

2.1. Analizar el diseño seguro de harware IoT

En los últimos años se ha observado una explosión en lo llamado *Open-Source hardware*, dispositivos al alcance de todo el mundo con un amplio soporte comunitario y que permiten rápidos diseños con un mínimo esfuerzo. Muchos de estos dispositivos sirven como base a los entornos IoT, debido a su versatilidad y coste reducido. Esto unido a que los fabricantes de microchip han visto en este mercado emergente un camino de futuro y han apostado por soluciones integradas y promoviendo muchos de los estándares actuales que se utilizan para la comunicación de los dispositivos tales como LoRa@, Zigbee, Bluethoot, etc. Esto ha propiciado que el mundo de los dispositivos IoT haya crecido exponencialmente en los últimos años, llegando a ser uno de los mercados con mejor perspectiva de futuro.

Arduino y Raspberry Pi son unos de los grandes referentes dentro del mundo de *Open-Source hardware*, pero que ha sido aprovechado por otros fabricantes, sobre todo en el mercado asiático, para poner en el mercado módulos programables de bajo coste de similares características, como LilyGO, Heltech, etc. basados en MCU o SoC como el ESP32 de Espressif.

Esto ha permitido poder disponer de módulos de bajo coste y fácilmente integrables, dado que desde la perspectiva del mundo IoT, los sensores pueden llegar a ser unidades con una vida limitada, dado su exposición a las condiciones climáticas, vandalismo, ambientes agresivos, etc., por lo que su abaratamiento ayuda a tener un adecuado ROI de los proyectos, donde los sensores se pueden contar por cientos o miles de unidades.

La seguridad en estos elementos viene dada por la forma de configuración y de fabricación de los mismo. Debemos tener en cuenta que estos dispositivos van a estar en entornos no controlados susceptibles de ser robados y/o utilizados para realizar ingeniería inversa con el consiguiente riesgo de conseguir información confidencial, como las claves de seguridad de nuestras redes LoRaWAN, contraseñas, logs, etc. Por ello, deberemos aplicar unas medidas de seguridad tanto físicas como lógicas en los sensores IoT, para evitar exponer dados sensibles a terceros.

De esta manera, si nos fijamos en la seguridad de los componentes hardware de los sensores IoT nos encontramos con los siguientes puntos a analizar:

- Construcción física del sensor.
- Almacenamiento estático de la información dentro del sensor.
- Relación coste-seguridad.

La construcción o diseño del entorno físico del sensor en muy importante de la seguridad. En el sensor tenemos elementos muy sensible a la seguridad como son los datos almacenados en la memoria Flash, que pueden estar incluidos dentro de los MCU, SoC o en chip independientes.

Muchos de los ataques basado en ingeniería inversa parte del hecho de que se pueden leer los datos almacenados en estos chip sin ningún tipo de problema o restricción, lo cual permite a un tercero conocer datos sensibles de nuestra red de sensores, como las claves de sesión LoRaWAN, la identificación del dispositivo, etc., tal como se indica la guía **Seguridad en la instalación y uso de dispositivos IoT** [9] de INCIBE "*Si un ciberdelincuente consigue acceso físico al dispositivo podría robarlo o destruirlo, causando una denegación del servicio, que podría dejar de estar accesible para los usuarios, con las consiguientes pérdidas económicas que se derivarían de la pérdida de funcionalidad. Asimismo, podría acceder a la información almacenada en el propio dispositivo para buscar datos confidenciales como credenciales de acceso, direcciones URL sensibles, registros de actividad o logs, etc*."

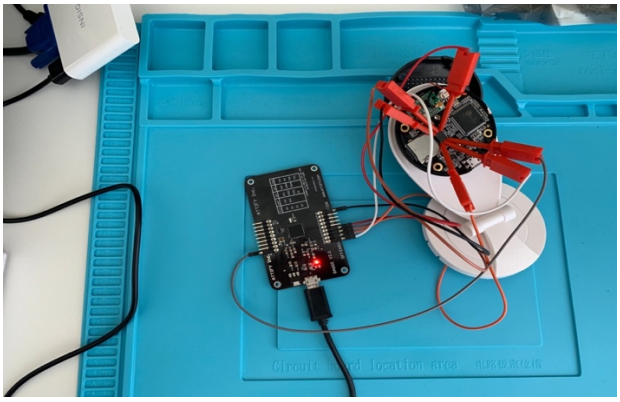


Figura 6: Lectura de memoria Flash de una cámara IP Sricam SP009 [10]

Por esto, debemos de dar la suficiente importancia a la fabricación física de los dispositivos IoT, para protegernos ante este tipo de ataques de ingeniería inversa.

Una de las medidas más usuales suele ser el borrado de los códigos de identificación de los chip, con lo cual evitamos a priori facilitar la extracción de información de los chips, al no saber que chip y su configuración de pines son los necesarios para poder la información de la memoria Flash. Pero esta claro que esto es una pobre medida de seguridad si no atenemos a dos parámetros, el primero es que trabajaremos con dispositivos open-source hardware, por lo que la información de los esquemas electrónicos y componentes que conforman los dispositivos IoT es de sobra conocidos por la comunidad y en segundo lugar hay que siempre tener en cuenta dentro del mundo de la seguridad de las TIC, la premisa más importante en cuanto a la seguridad informática es, "*la ocultación de información no es seguridad, dado que tenemos que trabajar con la premisa de que un atacante tiene suficiente información sobre nuestro entorno o componentes que conforman nuestra arquitectura*", y así tenemos que pensar que cualquier ataque sobre un dispositivo físico IoT, empereza porque el atacante tiene suficiente información sobre el dispositivo que desea atacar.

Dado esto, donde suponemos que el atacante conoce todo sobre el hardware que conforma el dispositivo IoT, nos deberemos centrar en como proteger el acceso al dispositivo por parte del atacante. En este punto una buena opción es rugerizar los dispositivos, esto es, hacerlos los más impenetrables posibles a sus elementos físicos. Para ello podemos hacerlo de dos maneras:

- Rugerización externa
- Rugerización interna

La rugerización externa se basa en cubrir el dispositivo IoT con una carcasa robusta [11] que evite su manipulación interna sin que esto afecte al dispositivo. El inconveniente de este tipo de soluciones suelen ser el alto coste de estos elementos y que suelen equipos voluminosos, que es lo contrario a lo que deseamos en este proyecto, unos costes reducidos, a la par que tampoco son soluciones impenetrables.

Después tenemos la rugerización interna o encapsulado electrónico, donde lo que hacemos es realizar un bloque macizo mediante resina de tipo Epoxi del dispositivo IoT, para que esta resina se infiltre entre todos los pines de los chips, de hay el termino “interna”, con el objetivo que cualquier tipo de acción sobre el dispositivo requerirá tener que rompe la capa de resina endurecida, lo que podrá producir la rotura de los pines de los chips y la imposibilidad de poder leer datos de los mismos. Además esta técnica tiene un beneficio añadido para los dispositivos IoT, al ser estos elementos expuestos a condiciones ambientales intempestivas, este rugerizado previene que el agua o la humedad puedan inutilizar o averiar el dispositivo prematuramente. Aparte es un solución muy barata y fácil de realizar, basta con rellenar la caja del dispositivo de resina, donde solo debemos tener en cuenta las necesidades de refrigeración de los componentes electrónicos que necesiten disipar mucho calor, para lo cual habrá que añadir un radiador a dichos componentes con necesidades extras de disipación y que el 40% del radiador quede al menos en el exterior del bloque de resina.

Figura 7: Encapsulado mediante resina [12]

Aun así, este método de protección física tiene un alcance limitado, dado por el tiempo que un atacante quiera utilizar para liberar el dispositivo de la capa de resina con sumo cuidado, lo cual nos debe llevar a incorporar otras medidas de seguridad para la protección de los datos del dispositivo.

Otro aspecto en cuanto la seguridad del hardware IoT es la elección del tipo de procesador que deseamos utilizar en los sensores, pues la seguridad en gran medida dependerá de la manera que el fabricante la ha implementado en el chip y que nivel alcanza. Los fabricantes de MCU y SoC llevan tiempo incorporando medidas de seguridad para evitar que una vez programado los dispositivos, las áreas sensibles de memoria y su acceso estén protegidas mediante sistemas de protección incorporados en los propios chip. Así, los fabricante de los chips más utilizados en las plataformas IoT, incorporan funcionalidades que permite bloquear al memoria Flash y/o encriptarla para así proteger los datos almacenados en ella. El uso de estas medidas de seguridad nos garantizará que aunque el dispositivo sea sustraído con propósito de extraer información confidencial de el, para realizar ingeniería inversa u obtener datos confidenciales para lograr un ataque más profundo, no será posible al estar la información protegida. Además, con este tipo de protección, la reutilización del dispositivo pasa por tener que borrar la memoria Flash para eliminar dicha protección, lo que hace que los datos que estuviesen residentes en el equipo se pierdan, manteniendo así la privacidad de los mismos.

Con todo esto que hemos visto, llegamos al punto en que tenemos que establecer hasta que nivel de seguridad queremos implementar asumiendo los costes que ello conlleva. Esta claro que no estamos manejando equipamiento con alta clasificación de seguridad, como por ejemplo tipo NATO SECRET (NS), lo que requeriría incluso que los chip fabricados estuviesen homologados y certificados para evitar ataques del tipo “Hardware Trojans” [13].

Por ello, debemos ser muy consciente de la relación seguridad/coste en este proyecto, donde lo que tenemos que hacer es de disponer de la máxima seguridad posible en nuestros sensores sin que ello suponga un coste excesivo que haga inviable el desarrollo del proyecto. En esta relación seguridad/precio la tendremos muy presente durante el desarrollo de nuestro proyecto, donde uno de sus objetivos es conseguir la máxima seguridad a un precio bajo.

Un esquema de seguridad propuesto [14] en el cual nos podemos fijar, nos da cuatro recomendaciones que deben seguir un dispositivo IoT para garantizar que es un dispositivo seguro en cuanto al almacenamiento de sus datos contra ataques.

Estas recomendaciones son:

1. Encriptación completa del disco o memoria Flash.
2. Almacenamiento seguro de claves y verificación de la integridad del sistema.
3. Cargador de arranque firmado.
4. Cargador de arranque cifrado.

Durante la análisis para la selección de un dispositivo IoT necesario para el desarrollo del proyecto, elegiremos el que cumpla un mayor de estas recomendaciones dentro del rango de precios fijado para la elección de dicho dispositivo.

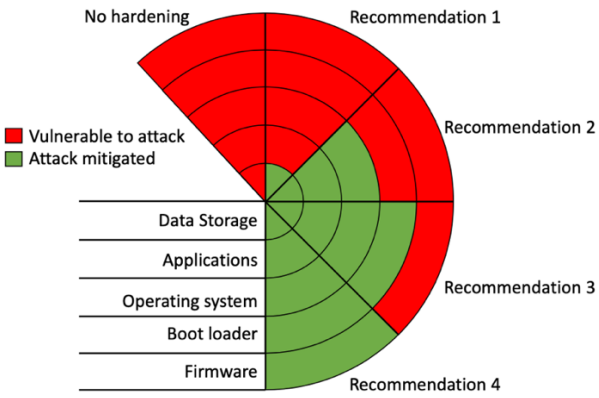


Figura 8: Recomendaciones de seguridad en dispositivos IoT [14]

2.2. Analizar la seguridad en LoRaWAN

Dado que desde los inicios de los dispositivos IoT se ha buscado un equilibrio entre los costes y las funcionalidades, se vio que estos dispositivos o sensores no iban a ser grandes generadores de datos, dado que la carga de información que generan es muy baja, hablamos de kilobytes de información, por lo que se puso más empeño en conseguir que el consumo de energía fue mínimo para lograr que los dispositivos puedan funcionar durante mucho tiempo sin necesidad de recargar de energía u operaciones de mantenimiento. De hay surgió las redes LPWAN y dentro de ellas, unas de las que más éxito esta teniendo son las redes LoRaWAN [15], una red LPWAN con capa de modulación física LoRa@ SFK.

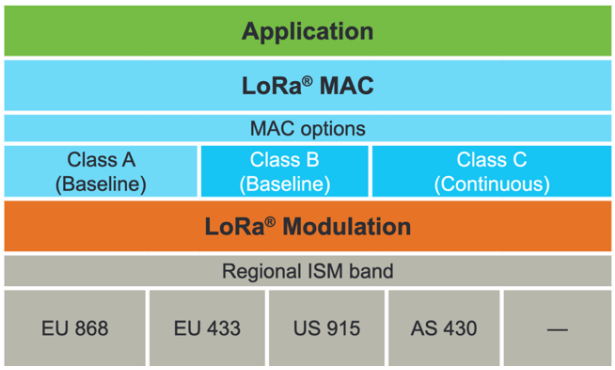


Figura 9: Esquema de arquitectura LoRaWAN [15]

LoRaWAN es un protocolo abierto desarrollado por LoRa@ Alliance[3] y con una la capa física de transmisión de LoRa@ desarrollada y licenciada por la empresa Semtech.

Cuando se ideo el protocolo LoRaWAN de transmisión, se puso mucho enfoque en la seguridad y para ello se embebido dentro del mismo protocolo una capa de seguridad que permita la transmisión de los datos por redes no licencias (ISM) de una forma segura y confidencial.

LoRaWAN realiza la encriptación de los datos transmitidos mediante AES128 de forma nativa en el propio protocolo, consiguiendo de esta manera un entorno de transmisión seguro. Esta seguridad se dividió en dos partes, y para entenderla mejor, tenemos que explicar como es la arquitectura de la red LoRaWAN.

En la red LoRaWAN nos encontramos los siguientes elementos:

- Los dispositivos o sensores IoT. Son los recolectores de datos que los transmiten mediante el protocolo LoRa@ en paquetes de radio para ser recibido por alguna pasarela de una red LoRaWAN.
- Los puertas de enlace o Gateway de la red LoRaWAN. Son los encargados de encaminar los paquetes de radio LoRa@ hacia otras redes de diferente tecnología como redes TCP/IP.
- Los gestores de la red de dispositivos o Network Server. Son los encargados de la gestión de los dispositivos IoT, su control y acceso a la red.

[3] https://lora-alliance.org/about-lora-alliance

- Y por último están los servidores que reciben los datos de los sensores o Application Server. Son los encargados de recopilar los datos originales enviados por los dispositivos IoT.

Los dispositivos IoT de red LoRaWAN, dependiendo del modo de funcionamiento (OTAA o ABP) tienen dos claves de sesión o que derivan estas a partir de un intercambio de información con el Network Server, una llamada NwkSKey y otra AppSKey.

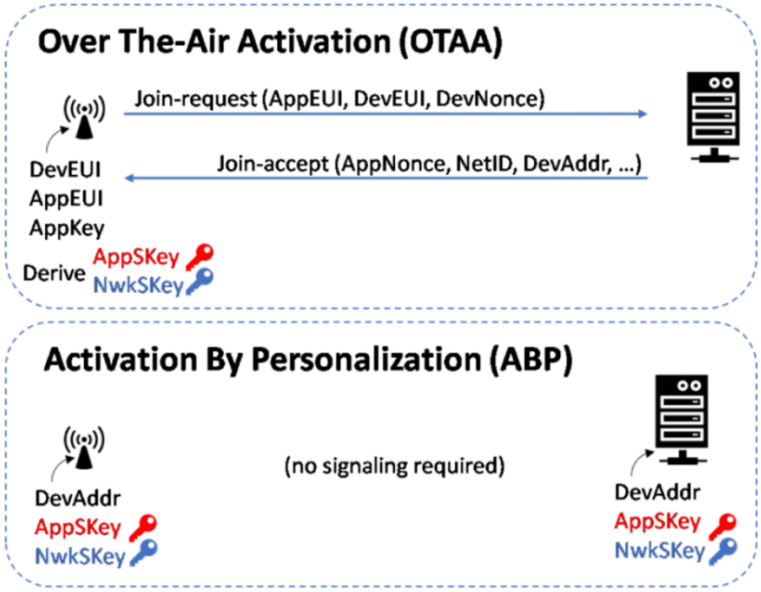


Figura 10: Gestión de claves en LoRaWAN según su modo de funcionamiento [16]

La clave NwSKey establece un contexto de sesión seguro entre el dispositivo y el Netwok Server, y así de esta manera separar la capa de gestión de los dispositivos ,de la capa de datos de los sensores.
La clave AppSKey es utilizada para el cifrado de los datos al Application Server para descifrarlos y de esta manera mantener un canal seguro de transmisión entre los dispositivos IoT y los servidores de aplicaciones. De esta manera, al tener dos planos de seguridad independientes, se consigue separa el ámbito de gestión de los dispositivos, del ámbito de la gestión de los datos, pudiendo ser gestionada cada capa de forma independiente sin verse comprometida la seguridad.
Estas dos claves dependiendo del tipo del funcionamiento elegido, deberán estar incluidas dentro del dispositivo IoT o no. Observamos que el modo más seguro es el OTAA dado que las claves de sesión se negocian en cada sesión. Además al no incluir dentro del dispositivo dichas claves, esto nos ayuda a mantener la seguridad de la red LoRaWAN.

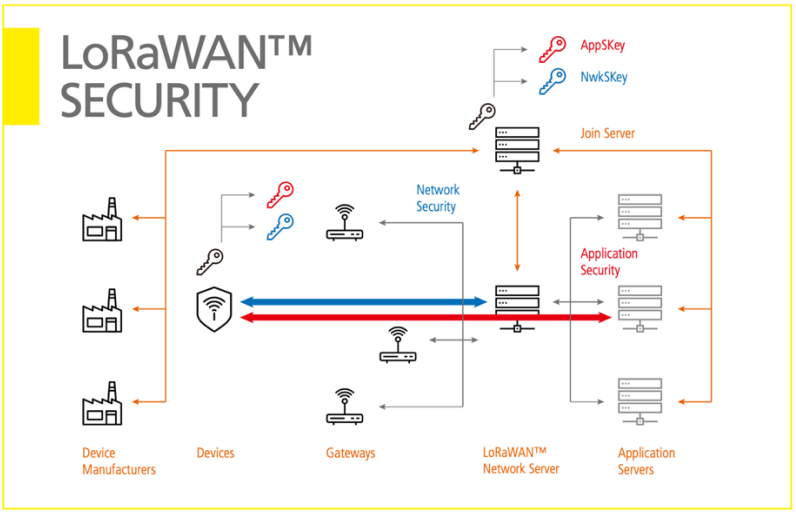


Figura 11: Modelo de seguridad en LoRaWAN

El punto de unión de los dispositivos IoT como el mundo TCP/IP de Internet se produce en las pasarelas o Gateway LoRaWAN. Estos dispositivos son los encargados de recibir los paquetes LoRa@ transmitidos por los dispositivos IoT, y de encapsularlos en paquetes IP para así entregarlos a través de una red publica o privada al Network Server que a su vez, los reenviara a los Application Servers.

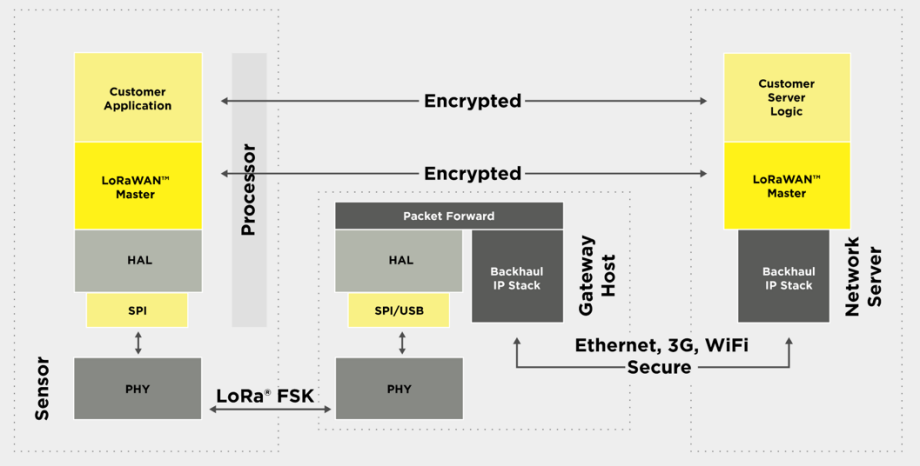


Figura 12: Esquema de capas LoRaWAN [17]

Cómo observamos en el gráfico, la pasarela LoRaWAN sólo realizan un reenvió de paquetes (packet forward) LoRa@ encapsulando los paquete recibidos en paquetes IP. Ahora bien esta encapsulación en paquetes IP requiere realizar una seguridad adicional de comunicación entre la pasarela LoRaWAN y el Network Server encargado de recibir dicho paquete.

La pasarela no tiene ninguna función en la seguridad de la comunicación entre los dispositivos y el Application Server, pero debe incorporar una capa de seguridad en lo que denominamos “Backhaul IP”, en la entrega de los paquetes LoRa@ al Network Server mediante una red IP. Los posibles ataques

que podíamos sufrir en esta comunicación seria del tipo "*Man-In-the-Midle*" [18], con la posibilidad de que un atacante interceptase o inyectase paquetes LoRa@ al Network Server. Para ello debemos garantizar que las comunicaciones de nuestra pasarela LoRaWAN son seguras y a pruebas de intrusiones por lo que tendremos que realizar un esfuerzo en securizar todo el entorno de gestión de la pasarela y sus comunicaciones externas.

2.3. Analizar la securización de la contenerización de aplicaciones

Como ya hemos visto anteriormente, la contenerización de las aplicaciones, nos permite una optimización de los recursos y además una simplificación del entorno de trabajo lo que conlleva tener la seguridad más focalizada al solo tener que securizar un único servidor, en lugar que tener que realizar las tareas de securización en varios servidores independientes con el consiguiente riesgo de que alguno de ellos que mal configurado securizado.

En este ámbito tenemos que poner el foco en los dos entorno que deberemos securizar para evitar un compromiso de la seguridad, el bastionado del sistema operativo anfitrión donde se realizara la contenerización de las aplicaciones y la correcta configuración de seguridad del entorno de contenedores. El bastionado o la securización de los sistemas operativos, es un mundo que es ampliamente conocido y donde llevamos muchos años de experiencias y de aplicar las buenas practicas en este campo, pero la contenerización de aplicaciones, aunque lleva más de una década en desarrollo, el continuo crecimiento tanto en tipo de plataforma como en su implementación, como por ejemplo en entornos DevOp, esto hace que todavía los conocimientos de seguridad de estos entornos no estén muy arraigados entre los profesionales del mundo TIC. Basta como ejemplo uno de los ataques más conocidos a la plataforma de contenerización Docker a través del malware Doki, donde se pone de relieve que esta vulnerabilidad no tuvo su origen en un fallo de diseño de la plataforma sino en las configuraciones incorrectas [19] de los entornos de producción, estando las plataformas vulnerables accesibles a través de Internet (nubes publicas).

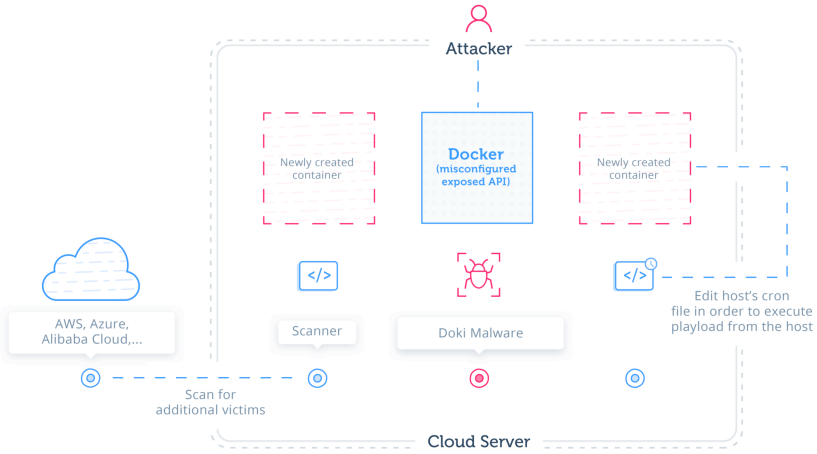


Figura 13: Malware Doki. [19]

Por esto, es adecuado profundizar más en el conocimiento de las buenas practicas de seguridad en el despliegue de contenerización de aplicaciones. Empresas [20] y organismos gubernamentales, tales como NIST [21] publican guías para el despliegue seguro de aplicaciones contenerizadas. Las guías indicadas en este punto nos servirán como un punto de referencia para la correcta implementación de una contenerización segura.

Otro aspecto de la seguridad en la contenerización de las aplicaciones es el uso de las imágenes de los contenedores. Un contenedor de una aplicación puede ser desarrollado por empresas, usuarios, comunidades de programadores, etc. y puesto a disposición del publico en general a través de las diferentes plataformas de librerías de imágenes de cada plataforma. En este punto hay que ser muy cauteloso a la hora de utilizar imágenes de aplicaciones desarrolladas por terceros. Un punto a seguir es utilizar imágenes originales de registros de confianza [22]. Estos registros de confianza pueden ser empresas o usuarios autorizados o de confianza que nos asegurara que el contenedor esta analizado, parcheado y escaneado convenientemente para evitar vulnerabilidades conocidas o directamente incluidas con propósitos maliciosos. Como ejemplo, nos podemos poner en el caso de utilizar una aplicación Open-Source que esta disponible en formato de contenedor. Si el contenedor que utilizamos es obsoleto y no incluye las mejoras que evitan vulnerabilidades ya conocidas de la aplicación, entonces ejecutaremos este contenedor con ciertas vulnerabilidades que pueden ser explotadas, donde estamos poniendo en riesgo nuestro entorno de trabajo y nuestro compromiso con la seguridad.

Con respecto a las securización de aplicaciones contenerizadas, no debemos olvidar el propio equipo que ejecutar el software de contenerización. Este equipo formado por un hardware, CPU, memoria, discos duros, dispositivos de entrada, etc., y por un sistema operativo, habitualmente y el más comúnmente utilizado en contenerización es Linux, pero ya habiéndose expandido a otros como Windows y Mac OS X, debe ser sometido a todos los procesos de bastionado o securización que garanticen su seguridad, así como la de la aplicación de contenerización. Para la realización de la securización de este equipo, podemos seguir guías como la realizada por el Centro Criptológico Nacional (CCN) para el sistema operativo Suse Linux 12[4], siendo muy fácilmente adaptables o transportable a otras versiones de Linux o guías más genéricas de bastionado de servidores como la "Guide to General Server Security[5]" de NIST. Pero las recomendaciones más básicas en securización de equipos y servidores son:

- Desactivar servicios y aplicaciones innecesarias.
- Configurar políticas de contraseñas robustas.
- Configuración de firewall de red.
- Habilitar los registros de actividad de usuarios y sistema.
- Utilizar el criterio de principio de mínimo privilegio [23] en el acceso a recursos.

[4] https://www.ccn-cert.cni.es/pdf/guias/series-ccn-stic/guias-de-acceso-publico-ccn-stic/3638-ccn-stic-617-implementacion-de-seguridad-sobre-suse-linux-enterprise-12-servidor-independiente/file.html
[5] https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-123.pdf

2.4. Analizar la seguridad en Internet

La seguridad de los sistemas que conforman la red Internet es uno de los grandes problemas a los que se enfrentan todos los ingenieros, investigados y científicos del mundo de las TIC qué diseñan y construyen esta inmensa red. En los tiempos actuales vivimos rodeados de amenazas constantes cuyo objetivo va desde simplemente conseguir una notoriedad publica hasta el enriquecimiento ilícito a través de acciones criminales.

Los elementos, protocolos y sistemas de seguridad han ido creciendo en potencia y complejidad de la misma manera que crecía Internet y sus amenazas. Así, hoy en día disponemos de múltiples sistemas que nos permiten garantizar la seguridad de nuestros sistemas que están conectados a Internet.

En el mundo de Internet, debemos tener en cuenta dos tipos de amenazas a las cuales nos debemos enfrentar para garantizar que nuestros sistemas son seguros y robustos antes las amenazas existentes en la actualidad, las primeras amenazas son aquellas que son inherentes al mundo de Internet, esto es, no podemos evitarlas dado que nuestros sistemas deben estar conectados constantemente a Internet y su desconexión quita el sentido al propio fundamento de estos sistemas, que es dar servicio a usuarios a través de Internet. Este tipo de amenazas suelen ser los ataques de denegación de servicio, ataques de inyección de código a servidores, etc., cuyo objetivo son los servicios publicados en Internet que tienen vulnerabilidades que permiten ser explotadas por terceros. Luego nos encontramos un segundo tipo de amenazas que no se orientan hacia los sistemas de Internet, si no a las redes privadas de gestión de los sistemas de Internet o sistemas que utilizan Internet para sus comunicaciones, por decirlo de otra manera, no es un ataque directo a una infraestructura de Internet, sino a los elementos que dispone de conexión a Internet. De esta manera, este tipo de ataques son los más habituales por su facilidad y capacidad de alcance que tienen. Entre este tipo de amenazas tenemos las infección por virus o malware, ataques de ingeniería social, phishing, entre otros, donde los atacantes consiguen hacerse con el control de estos sistemas conectados a Internet, que pueden ser unos pocos o millones de ellos, e intentar conseguir un beneficio económico, social, político, etc. o utilizarlos para otro fin como un ataque de denegación de servicio distribuido. También al hacerse con el control de estos elementos, los atacantes pueden acceder a otros sistemas a través de redes de gestión y de esta manera llegar indirectamente a las infraestructuras de Internet, tales como servidores Web.

Con esta ingentes cantidad de amenazas, los sistemas creados para garantizar la seguridad de los sistemas informáticos han ido en dos direcciones. La primera ha sido en la seguridad activa, elementos de control que en tiempo real monitorizan y detectan amenazas y aplican contramedidas para evitar que los ataques tengan éxito. En la seguridad activa podemos encontrar elementos hardware como cortafuegos que filtran las comunicaciones para garantizar que solo las permitidas se establezcan, y elementos software como los antivirus que monitorizan los archivos, memoria y procesos de los sistemas y detectan programas o rutinas maliciosas, provocando su eliminación para evitar que se puedan producir brechas de seguridad. Por otro lado esta la seguridad pasiva, más enfocada a la detección y alerta ante incidentes de seguridad. En este campo nos encontramos sistemas como los

detectores de intrusos o IDS, que nos alertan de posibles ataques de seguridad relacionados con patrones detectados tanto en las comunicaciones entre los sistemas como en la actividad interna de los mismo. La versión de IDS en seguridad activa es la protección ante intrusos o IPS.
Pero en los últimos años y con la necesidad cada vez más urgente de dar una repuesta inmediata a las amenazas o ataques han aparecido sistemas que integran en un único elementos todos los beneficios de la seguridad activa y pasiva, esto elementos se conocen como cortafuegos de siguiente generación "*Next-Generation Firewall*" (NGFW) [24] o gestor de amenazas unificadas "*Unified Theat Management*" (UTM). Estos sistemas integran en una única entidad las siguientes funcionalidades:

- Filtrado a nivel de aplicación: Las políticas de acceso ya no solo se realizan en base a la dirección IP o puerto TCP/UDP, sino que es capaz de reconocer que servicio de capa de aplicación se esta ejecutando en la comunicación para permitir o denegar. Así sería posible realizar una policita de seguridad que por ejemplo denegase el uso de la aplicación FaceBook solamente.
- Filtrado de URL (WebFilter): Permite reconocer las URL en base a categorizaciones realizadas por los fabricantes de los NGFW y de esta manera poder realizar políticas de acceso en base al tipo de URL que se establece en la comunicación. De esta manera se podría bloquear el acceso a URL que estén categorizadas por URL Maliciosas.
- Detección de virus: Mediante de la incorporación de la lógica de funcionamiento de los clásicos antivirus, estos NGFW, pueden ensamblar las comunicaciones que realizan transferencia de ficheros como SMB, analizar el fichero transmitido y comprobar que no continúe ningún virus o malware. Algunos NGFW realizan esta función mediante el uso de SandBox para el análisis de amenazas tipo "*Zero Day*" [25].
- Inspección SSL: Muchas técnicas que utilizan las amenazas para evitar se detectadas es el cifrado de sus comunicaciones mediante SSL. Los NGFW pueden realizar la inspección profunda de estos paquetes y detectar amenazas que de otra manera pasarían desapercibidas.
- IPS integrado: Los NGFW permiten realizar IPS de las comunicaciones que se establecen a través de el.
- Prevención de DoS: Podemos parametrizar múltiples parámetros de las comunicaciones TCP/IP para evitar ataques del tipo DoS y DDoS.

Así este tipo de sistemas, son los lideres en la protección de las infraestructuras tanto publicas como privadas frente a ataques y amenazas a los que están expuestos hoy en día.
Estos equipos que son los que más expuestos están a los ataques, dado que son los elementos que sepan nuestras redes privadas o internas de Internet, donde no hay ninguna política de seguridad aplicada en modo general (algunos ISP si proveen de algunos de control y monitorización de ataques), dado que prima el concepto de red abierta y se considera que todo trafico es legitimo. Por ello debemos aplicar una securización o bastionado de alto nivel para evitar que se puedan convertir en un objetivo

de fácil vulneración. Hoy en día, hay una gran cantidad de guías de empleo seguro de Cortafuegos, tanto de fabricantes como de entidades gubernamentales tales como el CNN que en su apartado de Guías de Seguridad de las TIC, publica regularmente las guías de procedimiento de empleo seguro de los fabricantes más utilizados en el mercado, tales como FortiNet, Palo Alto, etc. Estas guías nos ofrecen un buen punto de partida de la securización de los equipos, pero que siempre pueden ser ampliados por nuestros conocimientos y buenas practicas adquiridas. Una de estas buenas practicas en la securización de NGFW son:

- Desactivar el acceso de administración desde redes externas.
- Solo permití administración solamente mediante HTTPS y SSH.
- Utilizar puertos no estándar en la administración HTTPS y SSH.
- Requerir TLS 1.2 o superior en administración mediante HTTPS.
- Mantener tiempo de inactividad de sesión bajos.
- Restringir el acceso a administración a equipos confiados.
- Utilizar factor de doble autentificación para acceso de administración.
- Utilizar políticas de contraseñas robusta.
- Configurar autentificación SSH mediante uso de claves.
- Realizar copias de seguridad cifradas.
- Actualización del firmware ante alertas de CVE o del fabricante.
- Habilitar modo FIPS 140-2[6]/CC[7] si esta disponible.

[6] https://csrc.nist.gov/publications/detail/fips/140/2/final
[7] https://www.commoncriteriaportal.org/

3. Diseño de una red segura

El siguiente paso en el desarrollo de este proyecto, es el diseño de las infraestructura de la red de sensores IoT con todos sus elementos integradores.

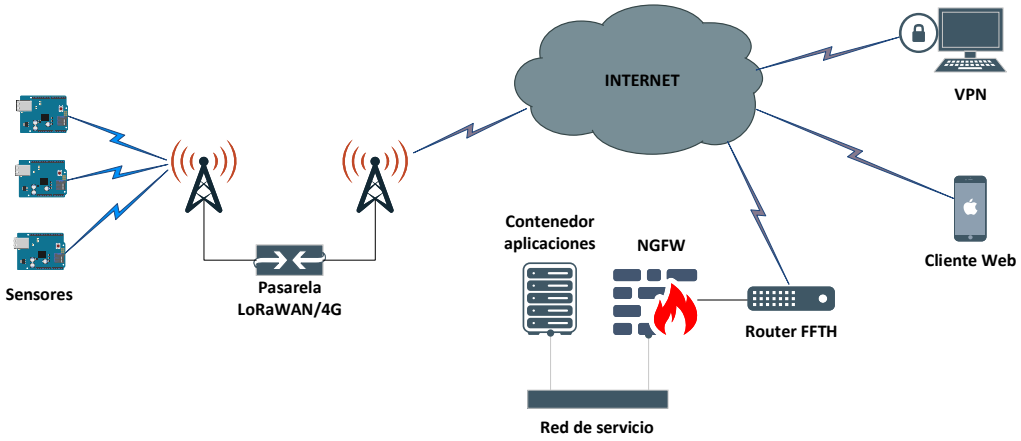


Figura 14: Arquitectura de alto nivel [Elaboración propia].

El funcionamiento de la arquitectura será el siguiente:

- Los sensores recopilaran los datos y los enviaran mediante protocolo LoRa@ a la pasarela.
- La pasarela recibirá los datos a través del protocolo LoRa@ y los reenviará al Network Server mediante una conectividad de 4G con Internet. El servicio para la recepción de datos del NetWork Server es publicado en Internet y protegido mediante el NGFW.
- Los datos son recibidos en el NGFW son reenviados al Network Server en la plataforma de contenerización.
- Dentro de la plataforma de contenerización los datos son recibidos, procesados y mostrados en un frontal Web que también esta presentado en Internet mediante el NGFW.
- Los usuarios dispondrán dos vías para la visualización de los datos, mediante el frontal Web accesible desde cualquier terminal y a través de VPN para la gestión de datos directos desde la base de datos.

La seguridad que ofrecerá la arquitectura es la siguiente:

- Los datos de los sensores hasta el Application Server están protegidos mediante el cifrado del protocolo LoRaWAN.
- Los acceso a aplicativos de gestión y/o visualización de gestión están cifrados con el protocolo TLS 1.2 o superior.
- Todos los datos procesados dentro de entorno de contenerización (Application Server, Base de datos) no tienen conexión fuera de la Red de servicio.

- El acceso directo a la base de datos solo es posible mediante conexión VPN a la red de servicio.
- La autentificación en el acceso VPN se realiza mediante factor de doble autentificación basado en OTP.

3.1. Diseño seguro de sensores IoT

Los sensores deberán contar con las siguientes características:

- Lectura de temperatura ambiente.
- Localización mediante GPS.
- Protección antirrobo y antivandálico.
- Modo de bajo consumo

3.1.1. Selección de plataforma hardware

Hemos analizado las distintas plataformas hardware open-source disponible en el mercado y de todas ellas hemos seleccionado cuatro para analizarlas y seleccionar la más adecuada. Las plataformas seleccionadas son:

- Arduino Uno
- Ardurino Mega 256
- Heltch Wifi 32 LoRa
- TTGO T-Beam

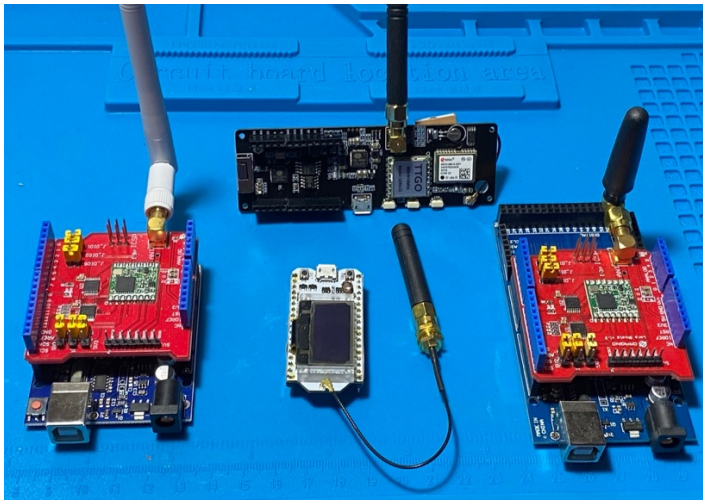


Figura 15: Plataformas a evaluar. [Elaboración propia].

Las dos primeras plataformas están basadas en MCU Atmel AVR y las otras dos están basadas en SoC ESP32. A continuación detallamos las características técnicas y de seguridad de cada plataforma:

- Arduino Uno:
    - MCU ATmega328 [26] / 8 bits
    - Memoria Flash: 32 KB, 0.5 KB usados por el Bootloader
    - SRAM: 2 KB
    - EEEPROM: 1 KB

- Puertos E/S digitales:14
- Puertos analógicos: 6
- Tensión de funcionamiento: 5 V.
- Consumo de corriente en modo reposo: 4.2 μ A
- Protección de lectura de Flash mediante Boot Lock Bit (BLB)[8]
- Precio venta publico[9]: 3,94 €/unidad.

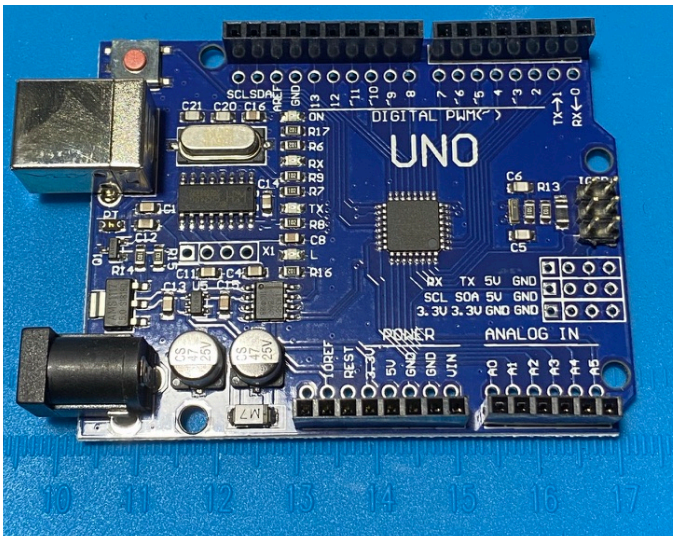


Figura 16: Placa Arduino UNO [Elaboración propia].

- Arduino Mega 256
  - MCU ATmega256 [27] / 8 bits
  - Memoria Flash: 256 KB, 0.5 KB usados por el Bootloader
  - SRAM: 4 KB
  - EEEPROM: 2 KB
  - Puertos E/S digitales:54
  - Puertos analógicos: 16
  - Tensión de funcionamiento: 5 V.
  - Consumo de corriente en modo reposo: <5 μ A
  - Protección de lectura de Flash mediante Boot Lock Bit (BLB)
  - Precio venta publico[10] : 6,69 €

[8] Requiere modificación del Bootloader
[9] https://es.aliexpress.com/item/32948661593.html
[10] https://es.aliexpress.com/item/33045194907.html

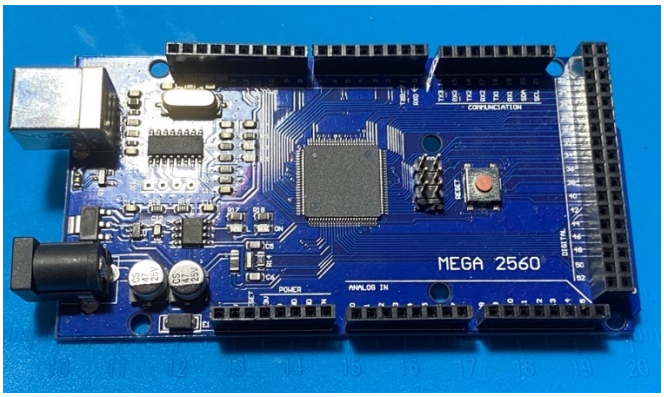


Figura 17: Placa Arduino MEGA 256 [Elaboración propia].

- Heltec Wifi 32 Lora
  - SoC ESP32-D0WDQ6 [28] / Procesador Xtensa LX6 de 32 bits con doble núcleo.
  - Hardware criptográfico acelerado en SOCs (AES, SHA-2, RSA, ECC, RNG).
  - Memoria Flash: 8 MB SPI Flash externa.
  - SRAM: 520 KB.
  - Puertos E/S digitales: 22.
  - Puertos analógicos: 18.
  - Tensión de funcionamiento: 5 V.
  - LoRa@ Chip SX1276.
  - Wifi 802.11 b/g/n.
  - Bluetooth 4.2.
  - Consumo de corriente en modo reposo: 800 µA.
  - Arranque Seguro.
  - Encriptación de memoria Flash.
  - 1024 bits OTP, 768 disponibles para el usuario.
  - Precio venta publico[11] : 15,33 €

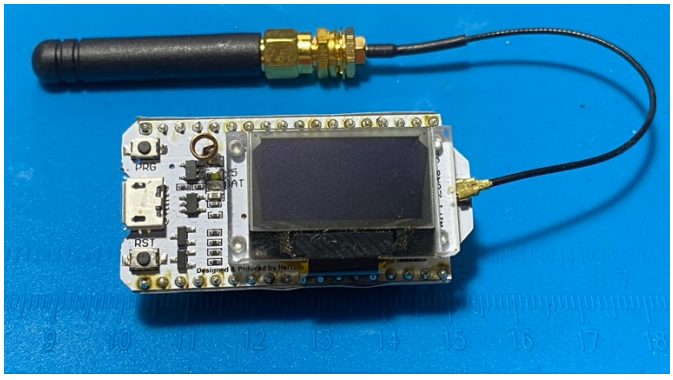


Figura 18: Placa Heltec WIFI32 LoRa [Elaboración propia].

[11] https://es.aliexpress.com/item/32838226825.html

- TTGO T-Beam
  - SoC ESP32-D0WDQ6 / Procesador Xtensa LX6 de 32 bits con doble núcleo.
  - Hardware criptográfico acelerado en SOCs (AES, SHA-2, RSA, ECC, RNG).
  - Memoria Flash: 4 MB SPI Flash externa.
  - SRAM: 520 KB.
  - Puertos de propósito general: 26.
  - Tensión de funcionamiento: 5 V.
  - Soporte batería 18650 Li-on
  - Modulo LoRa@ 868 Mhz.
  - Wifi 802.11 b/g/n.
  - Bluetooth 4.2.
  - Modulo GPS NEO 6 con EEPROM de 32 KB.
  - Consumo de corriente en modo reposo: 800 µA
  - Arranque Seguro
  - Encriptación de memoria Flash
  - 1024 bits OTP, 768 disponibles para el usuario.
  - Precio venta publico[12] : 24,53 €

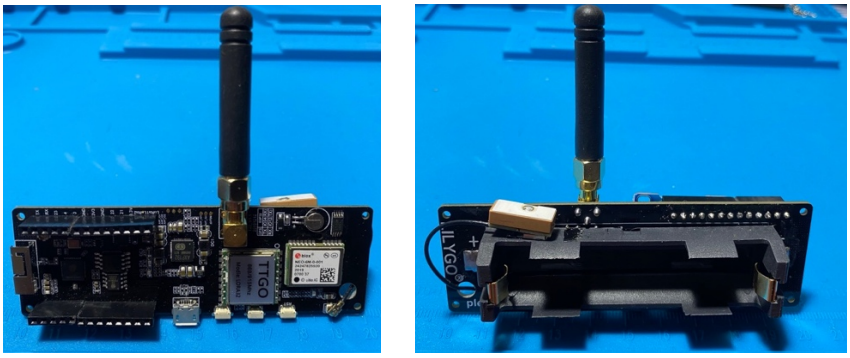


Figura 19: Placa TTGO T-Beam. Vista frontal y trasera [Elaboración propia].

De las 4 plataformas seleccionadas, las dos primeras son los más conocidas y sobre los que más información y desarrollo disponemos, pero con el inconveniente que debemos añadir los módulos externos para la localización GPS y la transmisión LoRa@.

Por otro lado, la seguridad de esta plataforma es muy escasa y solo se reduce a la utilización de bits de bloqueo de la Flash. No incluye de forma nativa ningún tipo de posibilidad de encriptación.

Las dos siguientes plataformas basadas en el SoC ESP32, son de mayor prestación que las dos anteriores, tanto por velocidad y capacidad en cuanto procesamiento de datos. Además en el plano de la seguridad los ESP32 incorporan una capa más potente y versátil de sistemas seguros que podemos

[12] https://es.aliexpress.com/item/4001054869183.html

utilizar para blindar la información del sensor. Disponemos de dos funcionalidades muy potentes como la encriptación de los datos en Flash y el arranque seguro. La encriptación de la Flash nos protege de lectura no autorizadas de firmware y el arranque seguro nos garantiza que solo nuestro código es cargado en el chip [29].

Ambas plataformas incorporan chip o modulo LoRa@ en modulación 868 Mhz para Europa, con conector de antena externa.

Por ultimo la plataforma T-Beam lleva integrado un modulo GPS con antena cerámica externa para geolocalización, que la plataforma Heltec Wifi32 no lleva y debería añadirse mediante modulo externo.

Los módulos externos que podrían ser necesarios en base a la plataforma seleccionada serian:

- Modulo LoRa@ Shield para Arduino[13]. Precio 16,23 €/unidad.
- Modulo GPS NEO-6M[14]. Precio 2,5 €/unidad.
- Modulo batería 18650 externa[15]. 1,84 €/unidad

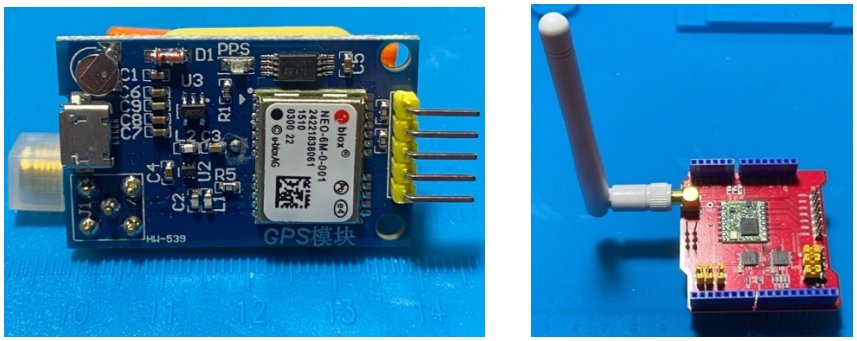


Figura 20: Módulos GPS y LoRa@ Shield para Arduino [Elaboración propia].

Además en todo los casos, necesitaremos el sensor externo de temperatura que hemos seleccionado, que es el siguiente:

- Sensor AM2320 de temperatura y humedad[16]. 1,27 €/unidad

[13] https://es.aliexpress.com/item/32892051769.html
[14] https://es.aliexpress.com/item/32849478314.html
[15] https://es.aliexpress.com/item/32996160839.html
[16] https://es.aliexpress.com/item/32829077742.html

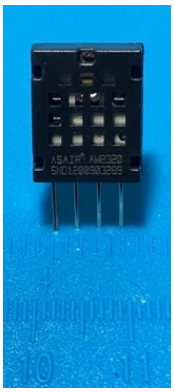


Figura 21: Sensor de temperatura AM2320 [Elaboración propia].

Con estos datos, podemos estimar el coste de cada sensor con todos sus componentes asociados.

	PLATAFORMA	MODULO GPS	MODULO LORA	MODULO BATERIA	SENSOR TEMPERATURA	COSTE TOTAL
ARDUINO UNO	3,94 €	2,50 €	16,23 €	1,84 €	1,27 €	25,78 €
ARDUINO MEGA	6,69 €	2,50 €	16,23 €	1,84 €	1,27 €	28,53 €
HELTEC WIFI32 LORA	15,33 €	2,50 €	INCLUIDO	1,84 €	1,27 €	20,94 €
TTGO T-BEAM	21,22 €	INCLUIDO	INCLUIDO	INCLUIDO	1,27 €	22,49 €

Tabla 2: Coste material dispositivo IoT

Siguiendo las recomendaciones realizadas en el punto 2.1 sobre la seguridad de los dispositivos IoT, analizamos que plataformas cumple con dichas recomendaciones.

	RECOMENDACIÓN 1 (ENCRIPTACION)	RECOMENDACIÓN 2 (ALMACENAMIENTO)	RECOMENDACIÓN 3 (BOOTLOADER)	RECOMENDACIÓN 4 (FIRMWARE)
ARDUINO UNO				
ARDUINO MEGA				
HELTEC WIFI32 LORA	✔	✔	✔	✔
TTGO T-BEAM	✔	✔	✔	✔

Tabla 3: Requerimiento de seguridad

Para finalizar, realizaremos una tabla de pesos para la elección de la plataforma sobre la que realizaremos el diseño de los sensores IoT. Estos pesos se regirán por las siguientes directrices:

- La máxima puntuación es de doce puntos.
- Precio: Sobre una escala lineal de cuatro puntos al más barato hasta un punto al más caro.
- Seguridad: Se suma un punto por cada recomendación de seguridad que se cumple.
- Integración: Sobre una puntuación de cuatro, se resta un punto por cada modulo exterior necesario. (Se excluye el sensor de temperatura).

La tabla resultante es la siguiente:

	PRECIO	SEGURIDAD	INTEGRACION	TOTAL
ARDUINO UNO	3	0	1	4
ARDUINO MEGA	4	0	1	5
HELTEC WIFI32 LORA	1	4	2	7
TTGO T-BEAM	2	4	4	10

Tabla 4: Tabla de pesos de plataformas IoT

Con un resultado de 10 puntos, la plataforma seleccionada para los sensores IoT del proyecto es la plataforma TTGO T-Beam, donde disponemos de la máxima seguridad y con una integración total en un único dispositivo lo que facilita su diseño.

3.1.2. Ensamblaje del sensor IoT

Para realizar el ensamblaje del sensor, utilizaremos una impresora 3D que nos permitirá realizar diseño personalizados a un bajo coste, a parte de la gran comunidad de diseñadores desinteresados que poden a disposición de todo el publico sus diseños realizamos. Hay que tener en cuenta que cualquier tipo de fabricación de moldeado en plástico en serie, requiere un gran desembolso inicial solo para realizar las matrices de inyección que pueden suponer más de 5.000€ sin ni siquiera haber realizar una sola unidad. Por esto el uso de impresoras 3D para la realización de prototipos o fabricaciones a pequeña escala son una opción muy rentable, con la rapidez y versatilidad que nos da el poder diseñar y fabricar por nosotros mismos estos elementos.

Tal como hemos vistos anteriormente, la rugerización de los dispositivos IoT nos permite dar un plus de seguridad al equipo así como mejorar su durabilidad al ser un elemento de exterior. Para ello realizaremos una caja en plástico ABS, donde insertaremos el sensor, con la batería incluida y lo rellenaremos con Epoxy, dejando solo al exterior la antena LoRa, la antena cerámica GPS, el botón de apagado/encendido y el puerto micro-USB para poder cargar la batería 18650 de Li-ion antes de ser puesto en producción.

3.2. Securización de comunicaciones LoRaWan

La securización de las comunicaciones LoRaWAN se basará en los siguientes puntos:

- Securización de las claves de sesión LoRaWAN.
- Securización de la pasarela LoRaWAN.
- Securización de las comunicaciones entre la pasarela LoRaWAN y el Network Server a través de Internet.

Para la securización de las claves de sesión LoRaWAN, utilizaremos el método de transmisión OTAA, que nos permitirá prescindir de tener que configurar las claves de sesión en cada dispositivo IoT.
El inconveniente de este modo de funcionamiento es que deberemos disponer de un único identificador LoRaWAN (DevEUI) para cada sensor. Esto tiene el problema que nos obliga a personalizar cada sensor con un firmware especifico para indicar en cada uno un DevEUI excluiso. Para solucionar este problema, nos basaremos en generar el DevEUI en base a la dirección MAC del interface Wifi de cada sensor. Recordemos que la plataforma elegida dispone de un interface Wifi 802.11. Al usar esta MAC para generar el DevEUI, podemos de disponer de una única versión de firmware y solo tenemos previamente disponer de la dirección MAC del interface Wifi de cada sensor, para dar de alta los sensores en nuestro Network Server. Solo debemos tener en cuenta que la dirección MAC del interface Wifi es de 48 bits, mientras que la dirección DevEUI es de 64 bits, por lo que deberemos añadir 2 bytes al principio de la dirección MAC para poder utilizarla como valor DevEUI en la red LoRaWAN.
La securización de la pasarela LoRaWAN es un punto muy importante dado que es el nexo de conexión de nuestra red LoRa con el mundo de Internet. Para ello debernos garantizar que no es accesible por personal no autorizado y que cumple los requisitos de seguridad y accesibilidad para su gestión.

Para esta funcionalidad hemos elegido la pasarela LG308 de Dragino, por sus siguientes características:

- Sistema basado en Open Source OpenWrt17.
- Bajo consumo de potencia.
- Múltiples protocolos de conexión a servidores IoT.
- Gestión por Web GUI, SSH via LAN o WiFi.
- Conexión a Internet vía LAN, WiFi, o 4G.
- 1 x SX1276/SX1278 modulo LoRa.
- Limited support in LoRaWAN/ Support Private LoRa protocol
- Máximo rango en LoRa: 5~10 km.

[17] https://openwrt.org/

Una ventaja de esta plataforma es estar desarrollada bajo OpenWrt, lo que nos permite, al disponer del código fuente[18], poder construir nuestro propio firmware, y securizarlo convenientemente a nuestras preferencias de seguridad.

Al estar basado en OpenWrt podemos seguir los pasos que se indican en “Hardening of your OpenWrt device” [30] para la securización del dispositivo.

[18] https://github.com/dragino/openwrt_lede-18.06

3.3. Securización de entorno de contenerización

Dado la envergadura del proyecto, los criterios de costes y la necesidad de ser una arquitectura lo más sencilla posible, se busca integrar todos las capas de aplicativos de la arquitectura en una única plataforma de contenerización. Con esto, conseguimos una reducción de costes, ya que solo se necesita adquirir un único equipo y simplificaremos la gestión y seguridad del entorno al tener que centrarnos en una única entidad.

Analizando las diferentes alternativas comerciales de servidores y equipamiento informático existentes en la actualidad, nos encontramos dos gamas muy diferencias de productos, la primera gama orientada para pequeñas empresas, también llamada "*Small Bunisses Server"* (SMB) y la gama profesional o "*Enterprise*" orientada a empresas de tamaño medio y alto. Para nuestro proyecto dada sus dimensiones nos centraremos en la gama de productos SMB para la adquisición de un servidor de contenerización.

Este servidor debe tener en cuenta las siguientes características:

- Disponer de sistema operativo Linux. Dado que la máximo de software de contenerización se ejecuta sobre este sistema operativo y es donde más compatibilidad encontraremos.
- Disponer de almacenamiento redundante. Es importante que ante un fallo de un disco duro, no se produzca ninguna perdida de datos.
- Consumo eléctrico reducido. Dado que será un equipo que funcionara 24x7, el consumo eléctrico es otro dato a tener en cuenta para evitar caer en costes indirectos excesivos.
- Coste reducido. Nos permitirá mantener el proyecto dentro de sus márgenes de gastos.

Con estos datos, hemos estudiado diferentes opciones y hemos selecciona tres equipos para realizar una comparativa y escoger el mejor de ellos.

- Servidor Dell Smart Value Power Edge T140[19]
    - Procesador Intel Celeron G4930 3.2 Ghz.
    - Memoria RAM 8GB/DDR4
    - Sistema RAID 1 de 2 disco duros de 2 TB.
    - Tarjeta de red 1Gb.
    - Sistema operativo no incluido.
    - Formato caja torre.
    - Consumo eléctrico 350 W.
    - Precio venta publico: 883,01 € + IVA.

[19] https://www.dell.com/es-es/work/shop/servidores-almacenamiento-y-redes/smart-value-power-edge-t140-server-standard/spd/poweredge-t140/pet1402b

- Servidor HPE Proliant MicroServer Gen10 Plus G5420[20]
    - Procesador Intel Xeon G5420 3.8 Ghz.
    - Memoria RAM 8GB/DDR4
    - Sistema RAID 1 de 2 disco duros de 1 TB.
    - Tarjeta de red 1Gb.
    - Sistema operativo no incluido.
    - Formato Microserver.
    - Consumo eléctrico 180 W.
    - Precio venta publico: 635,04 € + IVA.

- Servidor QNAP TS-251D[21]
    - Procesador Intel Celeron J4025 2.0 Ghz.
    - Memoria RAM 8GB/DDR4
    - Sistema RAID 1 de 2 disco duros de 3 TB.
    - Tarjeta de red 1Gb.
    - Sistema operativo Linux.
    - Incluye Docker y LCX.
    - Formato Microtorre.
    - Consumo eléctrico 15,25 W.
    - Precio venta publico: 563,03 € + IVA.

Para la elección del servidor de contenerización, realizaremos una tabla de pesos con los siguientes criterios:

- La puntuación máxima es de dieciocho puntos.
- Precio: Sobre una escala lineal de tres puntos al más barato hasta un punto al más caro.
- CPU: Sobre una escala lineal de tres puntos, siendo la mayor puntación para el procesador de mejor prestación y uno punto para el procesador con menos prestaciones.
- Memoria: tres puntos para 8 Gb, dos puntos para 4 GB y un punto para capacidad inferior a 1 GB.
- HDD: Sobre una escala lineal de tres puntos, siendo la mayor puntación para la mayor capacidad y uno punto para la menor capacidad total.
- Consumo eléctrico: Sobre una escala lineal de tres puntos, siendo la mayor puntación para el equipo con menor consumo y uno punto para el equipo con mayos consumo.

[20] https://buy.hpe.com/es/es/servers/proliant-microserver/proliant-microserver/proliant-microserver/hpe-proliant-microserver-gen10-plus/hpe-proliant-microserver-gen10-plus-g5420-server-smb-offer/p/ENTMS-003
[21] https://www.qnap.com/es-es/product/ts-251d

- Integración: Sobre una escala lineal de tres puntos, siendo la mayor integración de software incluida y tamaño físico y uno punto las menores de estas características.

El resultado es el siguiente:

	PRECIO	CPU	MEMORIA	HDD	CONSUMO ELÉCTRICO	INTEGRACIÓN	TOTAL
DELL T140	1	2	3	2	1	1	10
PROLIANT MICROSERVER	2	3	3	1	2	2	13
QNAP TS-251D	3	1	3	3	3	3	16

Tabla 5: Tabla de pesos de servidores.

El equipo elegido como servidor de contenerización es el QNAP TS-251D.
Este equipo nos ofrece en un reducido dispositivo, un servidor Linux totalmente configurado y operativo, con posibilidad de implementar autentificación de doble factor, integración nativa con Docker y LCX para contenerización de aplicaciones además de un consumo eléctrico muy reducido.
Este equipo se puede gestionar tanto por Web como por SSH, por lo que pondremos énfasis en estos puntos para su securización.

Como software de contenerización utilizaremos la aplicación Docker, disponible para QNAP. Las ventajas que nos ofrece Docker en cuanto a securización de los contenedores son las siguientes:

- Espacio protegido para variables y datos sensibles de los contenedores.
- Aislamiento de redes y servicios.
- Actualizaciones de versiones muy rápidas.

Un punto muy importante en el uso de Docker es la securización de los parámetros de los contenedores como contraseñas y ficheros de claves de los certificados digitales que necesitan para su correcto funcionamiento. Docker tiene una funcionalidad llamada "*secrets*"[22], que evita que estos datos sensibles puedan ser transmitidos más allá de los contenedores que tienen permitido su acceso y además son enviados de forma cifrada a los contenedores para su uso.
Una vez seleccionada la plataforma sobre la cual ejecutaremos contenedores de aplicaciones, vamos a elegir las aplicaciones necesarias para que poder operar correctamente un red LoraWAN.
Realizando un estudio de las diferentes plataforma de código abierto para entornos LoRaWAN, una de las más populares y que disfruta de gran aceptación por parte de empresas y usuarios es "*The Things Network*"[23] (TTN).

[22] https://docs.docker.com/engine/swarm/secrets/
[23] https://www.thethingsnetwork.org/

TTN dispone de una plataforma nube publica open-source, disponible para el publico en general. A su vez ,esta plataforma disponible para su despliegue en nube privada mediante despliegue en servidor o en contenedores Docker, donde esta versión se conoce como TTN Stack. Este contenedor es el que utilizaremos para el despliegue del entorno LoRaWAN.

Para el tratamiento de los datos recibidos de los sensores y su procesamiento para su posterior almacenamiento en una base de datos se utilizará la aplicación Node-Red. Esta aplicación también esta disponible en contenedor Docker. Para la securización de este servicio, Node-Red[24] nos suministra una guía de configuración segura de la aplicación.

Como base de datos, se ha seleccionado la aplicación MySQL también disponible en contenedor Docker. MySQL dispone de una guía de securización[25] de su entorno y de buenas practicas en la programación de los clientes para evitar entre otras cosas posibles ataques de inyección SQL.

El ultimo contenedor que necesitamos en el frontal Web para la visualización de datos que utilizaremos Grafana[26]. Grafana nos suministra una guía de configuración segura incluyendo la securización de su contenedor con Docker Secrets[27].

Todas las guías de securización aquí mencionadas deben ser adaptadas convenientemente al entorno de ejecución de Docker, si no son una guía especifica de seguridad para contenerización, dado que algunas configuraciones y/o parámetros no serán aplicables a los contenedores por la propia idiosincrasia de dicho entorno.

Por ultimo, un aspecto de la securización en entorno de aplicaciones es el acceso a servicios API[28]. Estos servicios suelen tener un control de acceso basado en usuario/contraseña o OAuth2. Dado que estos servicios se suelen configurar una vez y las contraseñas API son entregadas a terceros lo que hace muy difícil el hecho de tener que cambiarlas regularmente además de estar incrustadas en los códigos de programación, es necesario que las políticas de estas contraseñas sean diferentes a las políticas de contraseñas de los usuarios donde podemos forzar el cambio de las mismas en periodos pequeños de tiempo. Por ello, la política de contraseña de servicios API serán la siguiente:

- Longitud mínima: 25 caracteres
- Complejidad: Números , letras (mayúsculas y minúsculas) y caracteres especiales. Como mínimo 5 caracteres de cada tipo.

[24] https://nodered.org/docs/user-guide/runtime/securing-node-red
[25] https://dev.mysql.com/doc/refman/8.0/en/security-guidelines.html
[26] https://grafana.com/oss/grafana/
[27] https://dev.mysql.com/doc/refman/8.0/en/security-guidelines.html
[28] https://es.wikipedia.org/wiki/Interfaz_de_programaci%C3%B3n_de_aplicaciones

3.4. Securización de servicios en Internet

En un proyecto de estas características es fundamental garantizar la seguridad de los servicios conectados a Internet, de una manera proactiva y autónoma. Los NGFW nos permite aplicación avanzadas de medidas de seguridad con el respaldo de los SOC (no confundir con SoC) de los fabricantes con actualización casi inmediatas de las amenazas que se van descubriendo en Internet.
Con este equipo podremos realizar las siguientes funciones básicas en nuestro proyecto:

- Securizar la conexión entre el Network Server y la pasarela LoRaWAN.
- Securizar el servicio Web del frontal web de presentación publicado en Internet.
- Disponer de conexión VPN SSL con autentificación mediante OTP para la gestión interna de los datos de la red de sensores IoT.

Por ello seleccionaremos un NGFW de costes reducido y con las características de seguridad más adecuadas para nuestro proyecto. Para ello seleccionaremos NGFW con las siguientes características:

- Políticas control de acceso basada protocolo y aplicación.
- Políticas WAF para protección de servidores Web.
- Política de protección contra DoS.
- Protección contra intrusos en red (IPS).
- Redes privadas virtuales sobre SSL (VPN SSL).
- Geolocalización IP.

Para la selección del NGFW nos fijaremos en el Cuadrante Mágico de Gartnet de 2019 [31] .

Figura 22: Cuadrante Mágico de Gartnet [31].

En el observamos que el líder del mercado de NGFW es la empresa Palo Alto Networks. Palo Alto Netwokrs nació de la escisión de ingenieros de la empresa de Firewall NetScreen que más tarde seria absorbida por Juniper Networks que deseaba agregar esta gama de productos a su portfolio, es una empresa relativamente joven, pero que gracias a unos productos revolucionarios en cuanto a tecnología de inspección de paquetes, se posiciono muy rápidamente entre las mejores del mercado. Esto hizo que orientase sus productos hacia el rango de las medias y grandes empresas, sin realizar mucho enfoque en productos para las pequeñas empresas. Actualmente Pao Alto solo tiene dos productos para las pequeñas empresas sin llegar a implementar todas las funcionalidades de los NGWF de rango superior.

Siguiendo a Palo Alto Networks se sitúa como segunda gran compañía Fortinet. Esta compañía fundada el año 2000, por un CEO de la compañía NetScreen (se observa que esta compañía fue el nicho de donde han surgido las dos grandes empresas de NGFW actuales), lleva muchos años consolidando sus productos y su rango de clientes, lo que la hace abarcar desde productos para SOHO (Small Office/House Office), pasando por los productos para las pequeñas empresas y llegando hasta la gama de grandes empresas. De esta manera, Fortinet dispone de hasta 8 modelos distintos de NGFW en su nivel básico, lo que la hace ideal para poder elegir un NGFW con las características deseadas.

Comparando los dos modelos más similares de NGFW para nuestro proyectos encontramos los siguientes productos:

- Palo Alto PA-220[29]
    - 8 puertos 10/100/1000 Mb.
    - Rendimiento de firewall: 560 Mbps.
    - Rendimiento de inspección IPS: 150 Mbps.
    - Rendimiento de protección de amenazas: 150/260 Mbps.
    - Soporta VPN SSL/IPSEC: Si
    - Soporte para OTP: No
    - Consumo eléctrico: 21 W.
    - Nº de sesiones máximo: 64.000.
    - Precio venta publico: 673,3 € + IVA.

- Fortinet Fortigate FG-50E[30]
    - 7 puertos 10/100/1000 Mb.
    - Rendimiento de firewall: 2,5 Gbps.
    - Rendimiento de inspección IPS: 350 Mbps.
    - Rendimiento de protección de amenazas: 160 Mbps.
    - Soporta VPN SSL/IPSEC: Si

[29] https://www.paloaltonetworks.com/resources/datasheets/pa-220-specsheet
[30] https://www.fortinet.com/content/dam/fortinet/assets/data-sheets/FortiGate_FortiWiFi_50E_Series.pdf

- Soporte para OTP: Si
- Consumo eléctrico: 18 W.
- Nº de sesiones máximo: 1.300.000.
- Precio venta publico: 340,9 € + IVA.

Se observa rápidamente que el equipo de Fortinet es el más indicado para este proyecto en cuando prestaciones totales y coste del equipo.
Una vez seleccionado el equipo, nos enfocaremos a la securizacion del mismo. Para ello seguiremos la Guía de Seguridad de las TIC CCN-STIC 1406 de Procedimiento de empleo seguro Cortafuegos FortiGate del CCN[31]. En este documento encontraremos las configuraciones necesarias que hay que realizar en el NGFW para su securización con respecto a lo indicado en el punto 2.4 de este proyecto.

[31] https://www.ccn-cert.cni.es/pdf/guias/series-ccn-stic/1000-procedimientos-de-empleo-seguro/4300-ccn-stic-1406-procedimiento-de-empleo-seguro-cortafuegos-fortigate/file.html

3.5. Proyecto de red de sensores IoT segura

Este proyecto se contratará con una empresa del ramo de las TIC de la zona, que dispondrán de los siguientes recursos humanos:

- Ingeniero Técnico en Telecomunicaciones: Encargado del despliegue de la red LoraWan, sensores y entorno de comunicaciones y seguridad de Internet.
- Ingeniero Técnico en Informática: Encargado de la arquitectura de aplicaciones, presentación de datos y entorno Web.
- Gestor de proyecto y seguridad: Sera la persona que dirija el proyecto, marque los requisitos de ejecución y se encargue de la planificación de los trabajos y aseguramiento de la calidad.

A su vez, una vez terminado el proyecto, la empresa, que este proyecto será RedesIPv6, ofrecerá un servicio de mantenimiento, para realizar un ciclo continuo del control de la seguridad de la arquitectura, como actualización de parches de seguridad, alertas de seguridad, gestión del control de acceso a los sistemas, etc.

Referente a la arquitectura de la solución, las capas de la arquitectura de telemetría IoT están formadas por la red de sensores, la red de transporte de datos dividida en la capa LoRaWan y la capa de red TCP/IP, la capa de aplicativos de gestión que la conforman los diferentes elementos de control de la red IoT y por ultimo la capa de presentación de los datos.

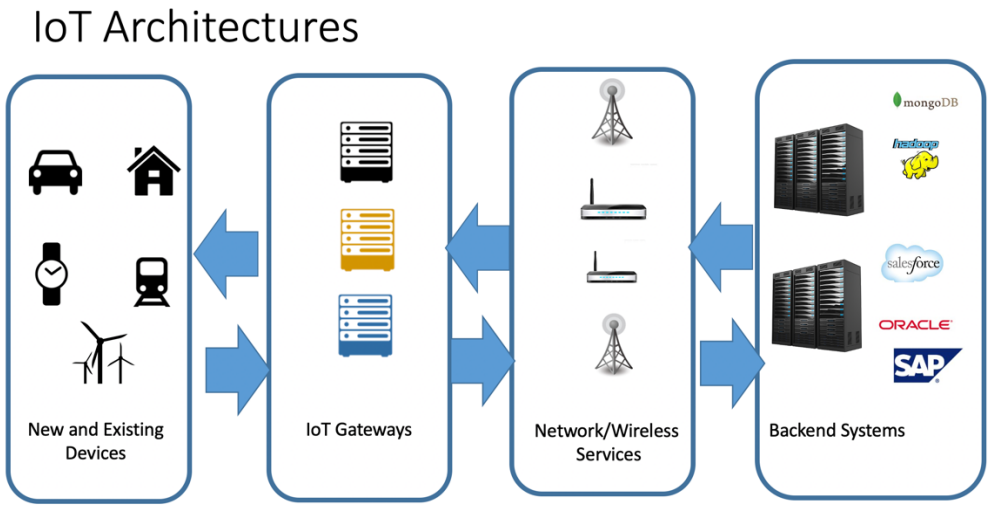


Figura 23: Arquitectura básica de plataforma IoT [32]

En todo momento nuestro objetivo será aplicar las mejores prácticas en la aplicación de la seguridad tanto física como lógica, siempre adaptada a la magnitud de nuestro proyecto.

En este trabajo fin de máster nos encontraremos con los siguientes elementos:

- Sensores IoT con conectividad LoRaWan, qué transmitirán una serie de información para ser procesada y mostrada por los frontales de la aplicación

- Gateway LoRaWAN, necesarios para conectar interconectar las redes de sensores con protocolo LoRa@ y las redes TCP/IP e Internet
- Network Server y Application Server, encargados de la gestión y seguridad de las conexiones con los dispositivos IoT.
- Bases de datos, necesarias para almacenar almacenamiento de la información obtenida por los sensores
- Servidor de visualización de datos, que nos da la interpretación visual de los datos obtenidos de la red de sensores.
- Equipo de securización de redes, para proteger los diferentes sistemas de las redes no controladas como Internet.

La solución debe ser lo suficientemente segura, como para evitar que los datos tratados puedan ser manipulados, robados y que además sea robusta ante ataques informáticos y que solo sea accesible por las personas autorizadas para su uso.

El proyecto se desplegará en el termino municipal de Cobreros, área principalmente dedicada a la ganadería no intensiva, y con diversas pedanías que hacen de esta un área de bajo población muy dispersa.

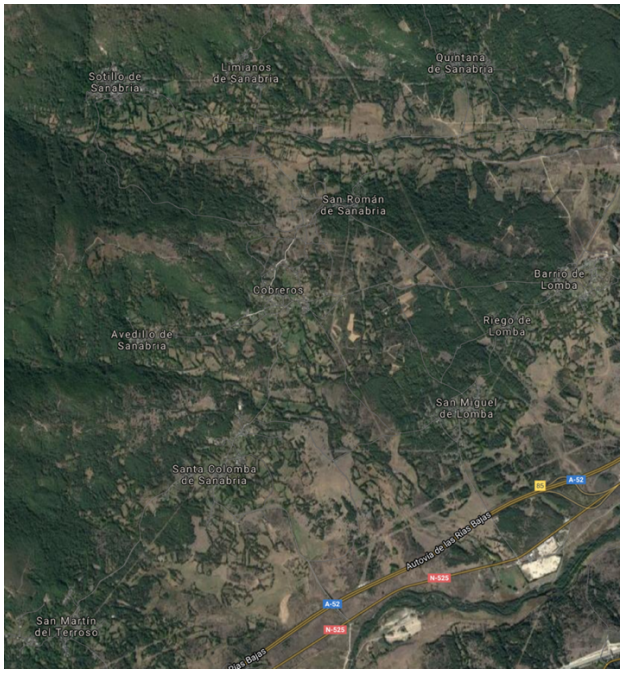


Figura 24: Termino municipal de Cobreros [Elaboración propia]

El equipamiento central, línea FTTH, QNAP y NGFW se encontrarán situados en el edificio del Ayuntamiento de Cobreros.

En este proyecto, no entraremos en cálculos de zonas de RF, dando por hecho que el radio de alcance en espacio libre de LoRa@ ronda los 8 a 10 Km y la distancia entre las pedanía más distante es de solo 2 Km, por lo que el alcance de las pasarelas LoRaWAN situadas en cada una de las pedanías, es

suficiente para cubrir de cobertura el termino municipal y además proveer de redundancia ante caída de diferentes nodos.

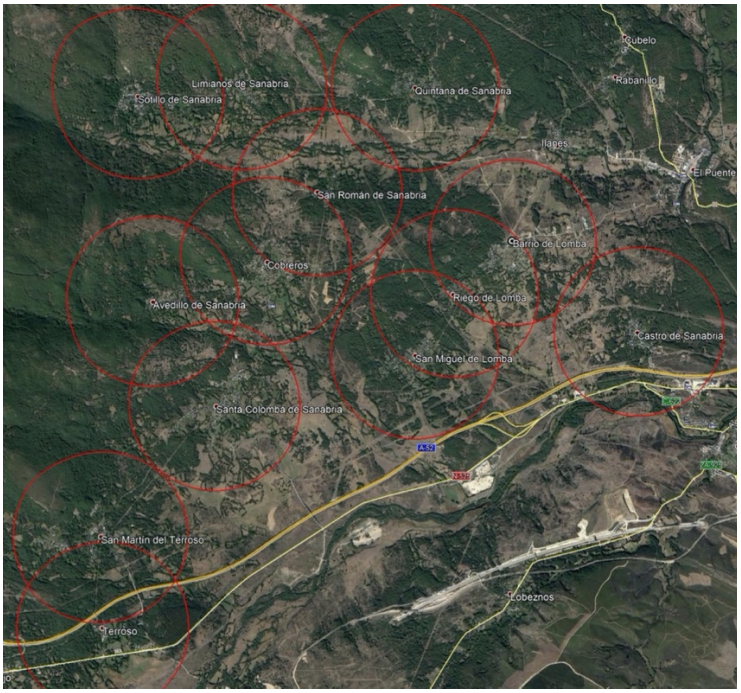


Figura 25: Áreas de cobertura de 1 Km de radio [Elaboración propia]

Los nodos serán en total 13, pero para el desarrollo de este proyecto, solo configuraremos una pasarela LoRaWAN, el nodo de Limianos de Sanabria, dado que el resto de los nodos tendrán la misma configuración a nivel de seguridad (excepto el usuario/password de administración que será único por cada pasarela) y solo cambiarán los parámetros de red LoRaWAN (Gateway ID) y el segmento IP de gestión.

Basándonos en el diseño de la arquitectura de red de alto nivel que hemos propuesto, pasaremos a definir la arquitectura a bajo nivel y establecer sus parámetros de conectividad y comunicación.

- Red pasarela LoraWAN (Nodo Limianos)
    - Segmento IP LAN de gestión: 192.168.10.0/24
        - Dirección IP LG308: 192.168.10.10/24. Esta red sirve para la administración del equipo mediante el interfaz LAN.
- Red de servicio
    - Segmento IP: 192.168.20.0/24
        - Dirección IP QNAP : 192.168.20.5
        - Dirección IP LAN Firewall : 192.168.20.1

- Red de interconexión Internet:
    - Segmento IP: 192.168.30.0/24
    - Dirección IP WAN Firewall: 192.168.30.1
    - Dirección IP Router FFTH: 192.168.30.2

El esquema de la infraestructura de red resultante es el siguiente:

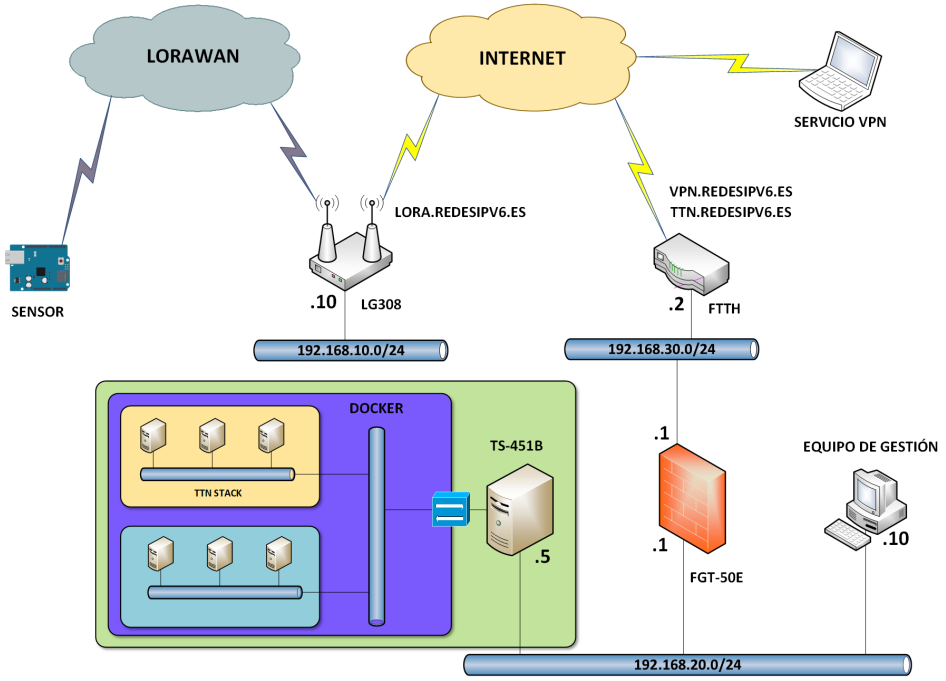


Figura 26: Arquitectura de red detallada [Elaboración propia]

Los nombres de dominio publicados y su servicio son los siguientes:

- TTN.REDESIPV6.ES → IP Publica FFHT para conexión al Gateway Server desde la pasarela LoRaWAN LG308.
- VPN.REDESIPV6.ES → IP Publica FFHT para conexión a los servicios internos mediante VPN.

Estos nombres de dominios se actualizan mediante servicios DDNS dado que las IP de los servicios de operador de FTTH no son IP estáticas y cambien en el tiempo. Por esto se mantendrá un TTL bajo en los registros A de estos nombres de dominios.

4. Implementación de red segura

4.1. Despliegue de sensores

A partir de la aplicación **TTGO T-Beam Tracker for The Things Network** [33], realizaremos nuestro desarrollo, dado que se ajusta a los requerimientos de nuestro proyecto. Realizaremos las modificaciones oportunas sobre este código para adecuarlo a los datos que queremos transmitir.
Para la programación de los sensores seguiremos las siguientes directrices:

- Funcionamiento OTAA del modulo LoRaWAN: Configuraremos nuestros sensores en modo OTAA y generaremos un APPKey único para la aplicación [Anexo C.1].
- El parámetro DevEUI se generará desde la MAC del interfaz Wifi.
- Se configurará el modo de arranque seguro del sensor: Mediante la utilidad **menuconfig** de las herramientas de programación ESP-IDF de Espressif [34] en nuestro proyecto, seleccionamos dentro del menú **Security features**, la opción **Enable hardware Secure Boot in bootloader** [Anexo C.2]. Como clave de firma de inicio segura para el **Secure Boot**, mediante OpenSSL [35] generamos una clave RSA con el comando ***openssl genrsa -out my_secure_boot_signing_key.pem 3072*** y la guardamos en el directorio de nuestro proyecto. Esta clave deberá guardarse de forma segura y controlada.
- La grabación del programa en la flash se realizará en modo encriptado: Dentro del menú **Security features**, seleccionamos la opción de compilación para la encriptación de la memoria Flash **Enable flash encryption on boot** [Anexo C.3] y cambiamos en modo de uso de **Development** a **Release** [Anexo C.4].

Una vez programado cada sensor y verificado su funcionamiento, se procederá a compilar y programar el sensor en modo seguro con las utilidades ESP-IDF, cifrando así su contenido y protegiéndolo de accesos indebidos.
En la ultima fase se realizará la rugerización del equipo, para ello realizaremos una caja contenedora para la resina epoxy mediante el programa Fusion360 de AutoCAD.

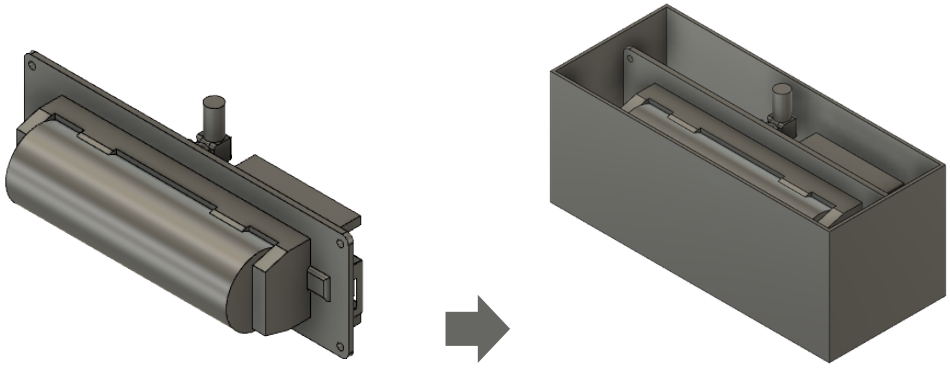


Figura 27: Proceso de rugerización del sensor [Elaboración propia]

En la parte superior tendrá la salida de la antena LoRaWAn y en la parte inferior el conector microusb para la recargar de la batería. Fabricamos la caja mediante impresión 3D con plástico ABS resistente a los rayos UV, e Introducimos en el sensor en la caja y la rellenamos de resina epoxy, teniendo cuidado de proteger los conectores de la antena LoRaWAN y microusb para evitar que queden cegados por la resina.

Alternativamente podemos utilizar algún diseño disponibles en Thingiverse[32] para la fabricación de la caja para TTGO T-Beam bajo licencia CC.

Con esto tenemos un sensor preparado para operar en nuestra red LoRaWAN y con un nivel seguridad bastante adecuada para el propósito del proyecto.

[32] https://www.thingiverse.com/thing:3830711

4.2. Configuración segura de la pasarela LoRaWAN

Después de realizar la configuración inicial de los parámetros de red LAN y 4G, parámetros de LoRaWAN y de LoRa, realizaremos la securización de la pasarela LoRaWAN.

Primero configuraremos los usuarios administradores (root y admin) con una contraseña lo suficientemente robusta, más de 10 caracteres alfanuméricos con mayúsculas y minúsculas y al menos un carácter especial ($%&+-_). Las dos contraseñas tienen que ser distintas y no deben ser parecidas, si la contraseña root es por ejemplo "*P@ssw0rd2020*", la contraseña admin no puede ser del estilo "*P@ssw0r2019*" ni ningún otra secuencia de caracteres similar.

Por defecto, el fimware original de OpenWRT en que se basa el equipo LG-308 y la compilación realizada por el fabricante Dragino para dicho equipo, no tiene activado el protocolo HTTPS para la administración mediante el interfaz GUI por lo que la información confidencial como la contraseña de autentificación se transmitirá en claro y esto no debe ser así. El certificado digital que utilizaremos para el cifrado del protocolo HTTPS lo obtendremos del servidor QNAP, tal como se explica en el siguiente punto de este proyecto.

Para activar el uso de HTTPS en la pasarela LG-308 deberemos realizar los siguientes pasos:

- Accedemos al equipo mediante SSH.
- Necesitamos carga el paquete libustream-openssl para habilitar el HTTPS, para ello introducimos el siguiente comando ***opkg install libustream-openssl*** a través de la consola SSH.
- Ahora mediante SCP, copiamos los certificados digitales necesarios para el cifrado en el directorio ***/etc***. El certificado con la clave privada se debe llamar ***uhttpd.key*** y el certificado con la clave publica se debe llamar ***uhttp.crt***.
- Ya solo nos queda reiniciar el servicio de HTTP con el comando ***/etc.init.d/uhttp restart*** para que empiece a funcionar la administración a través del interfaz Web también con HTTPS.

Podemos ahora comprobar que podemos acceder mediante la URL https://lora.redesipv6.es a la administración de la pasarela LoRaWAN, desde un equipo en la red de gestión.

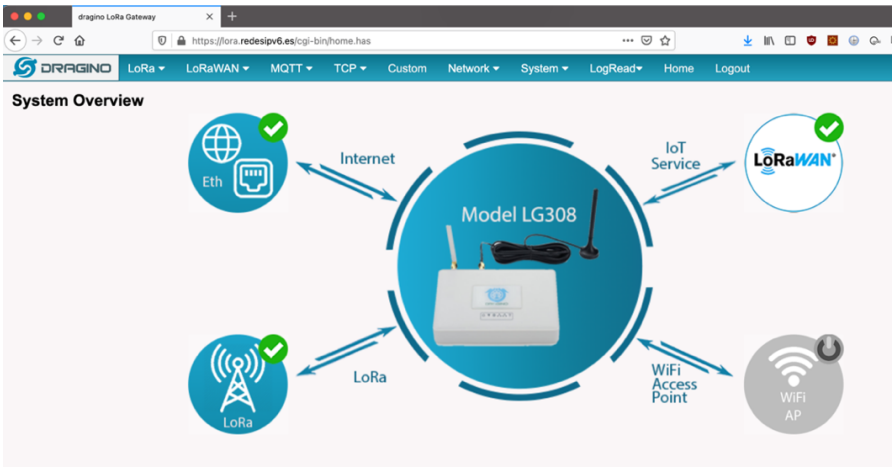


Figura 28: Pagina en HTTPS de administración Web de la pasarela LG-308 [Elaboración propia]

El siguiente paso en la securización de la pasarela, es la desactivación de servicios no esenciales para su funcionamiento y que puedan ser una fuente de vulnerabilidades. Para saber que servicios están activos, mediante un acceso al CLI por SSH comprobamos los procesos que se están ejecutando en el sistema [Anexo C.5].

Podemos comprobar que solo se están ejecutando los servicios necesarios para el funcionamiento de la pasarela LoRAWAN y esto es debido a que el fabricante del equipo, al personalidad el firmware OpenWRT, ha eliminado software y servicios que no eran útiles para la finalidad última del equipo. Esto nos garantiza que el equipo solo responderá a los servicios estrictamente necesarios.

Por último nos queda la securización de la gestión remota. Dado que la ubicación de las pasarelas LoRaWAN se realizará en lugares óptimos para la transmisión de radiofrecuencia, donde se elegirá emplazamientos que pueden ser de difícil acceso, para poder conectar un equipo en el interfaz LAN y poder gestionarlo. De esta manera, para gestionar el equipo remotamente tenemos dos opciones, la gestión directa a través del puerto WAN, en nuestro caso será el interfaz 4G o mediante un túnel SSH. La primera opción se basa en que el servicio de SSH este escuchando por el interfaz WAN y responda a cualquier petición de conexión. Esta opción de gestión tiene dos problemas, el primero es que nuestro equipo esta abierto a todo Internet con lo cual nuestro servicio SSH será susceptible de recibir peticiones de autentificación que no sean legitimas y que tenga como objetivo, hacerse con el control del equipo. El segundo problema viene dado por el tipo de conexión WAN que utilizamos que es una red 4G de telefonía móvil. Muchos operadores de telefonía móvil utilizan CG-NAT [36] para la conexión a Internet de los dispositivos de sus redes 3G/4G, para el ahorro de direccionamiento IPv4 que en la actualizada esta prácticamente agotado. Esto provoca que los dispositivos móviles y en nuestro caso el interfaz WAN 4G adquiera una dirección IP privada y no tenga una equivalencia 1 a 1 con una IP publica, por lo que nunca sabremos a que IP publica deberemos conectarnos para acceder al servicio SSH de la pasarela LoRaWAN. Estos dos problemas no se suelen presentar simultáneamente, esto es, si disponemos en el interfaz WAN de una IP publica seremos vulnerables a ataques a SSH (intrusión por fuerza bruta, DoS,etc.) pero no tendremos el problema de CG-NAT, pero si tenemos una IP privada porqué el operador dispone de CG-NAT, no estaremos visibles en Internet por lo que no nos podrán atacar al servicio SSH pero tampoco podremos conectarnos legítimamente para la administración remota.

Para solventar estos dos problemas, el equipo LG-308 dispone de la opción de establecer un túnel SSH [37] para la administración remota. Esto permite que las pasarelas LoRaWAN establecen un túnel SSH con un servidor, que en nuestro caso será el servidor QNAP, y así desde un equipo en la red de gestión, tomando como dirección IP de la pasarela LoRaWAN la dirección IP del servidor QNAP y el puerto especificado en el túnel, podremos conectarnos a las pasarela LoRaWAN directamente. De esta manera nos podremos conectar a la pagina de administración Web de la pasarela LoRaWAN desde la red de servicio sin necesidad de realizar ningún otra configuración. Además como es la pasarela LoRaWAN quien inicia la comunicación desde la red 4G solucionamos el problema del CG-NAT, dado

que el único dato que se necesita para establecer este túnel es la IP de nuestro NGFW y esta publicada en un registro DNS.

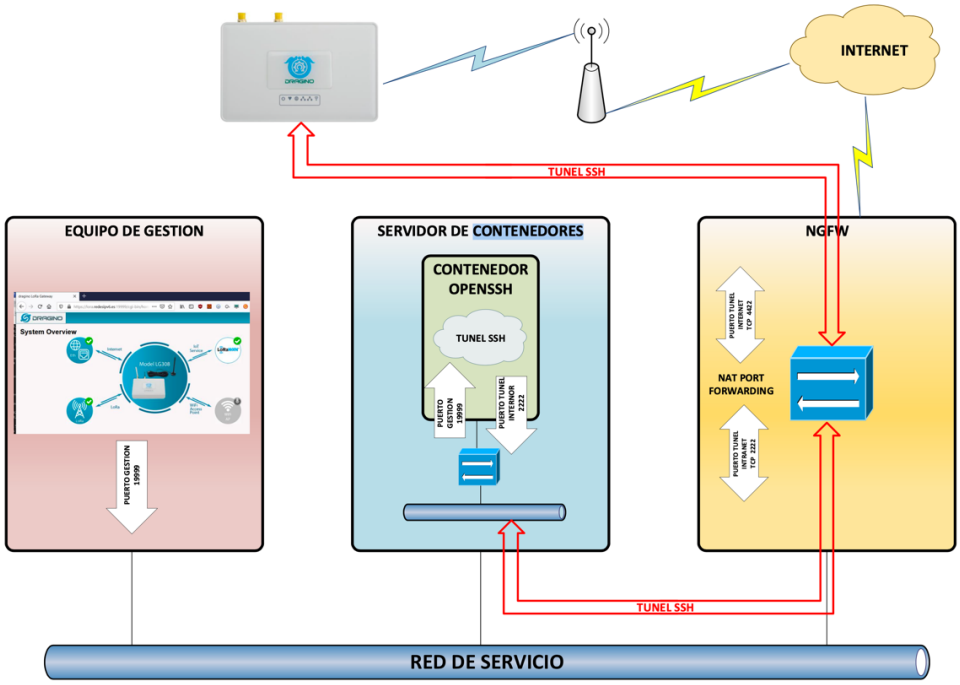


Figura 29: Flujo de trafico en túnel SSH [Elaboración propia]

Además para agregar una capa de seguridad más a la gestión remota de las pasarelas LoRaWAN, en la terminación de los túneles SSH utilizaremos un contenedor especifico para esta función. Así no se expondrá el servicio SSH del QNAP a Internet, aislando este trafico dentro del contenedor indicado para este fin. La autentificación que utilizaremos para perimir la conexión de la pasarela LoRaWAN al servicio SSH del contenedor se realizara mediante clave publica/privada RSA.

La ventaja que tiene establecer el túnel SSH con respecto de otros tipos de tecnologías de tunelización como IPSEC es:

- El consumo de recursos locales es bajo, dado que no requiere instalar ningún servicio adicional al SSH ya existente.
- Es más sencillo de configurar, sin mermar la seguridad de la conexión.
- Muchas redes con CG-NAT tienen problemas o limitaciones con el trafico IPSEC. Siendo casi obligatorio el uso de NAT-T.

Esta configuración nos abre un dilema, como antes hemos comentado, que es el tener que abrir un puerto de administración en Internet que es un riesgo que debemos sopesar. Esta claro que cualquier servicio en Internet esta expuesto a ataques del tipo de DoS, pero en nuestro caso, no somos una entidad suficientemente relevante como para ser un objetivo de este tipo de ataques. Otra cosa son los ataques de acceso por fuerza bruta para encontrar servicios de administración, correo electrónico (SMTP) y otros fácilmente vulnerables y poder acceder a ellos para cualquier tipo de acción maliciosa.

Ante este tipo de ataques, tenemos dos métodos de mitigación que pondremos en practica, uno será el uso de puertos no estándar para la publicación de servicios en Internet, de esta manera el servicio SSH del contenedor en vez de usar y publicar el puerto estándar TCP 22 del SSH, lo configuraremos para publicar el puerto TCP 2222 en el contenedor y el puerto TCP 24587 en Internet a través del NGFW, el segundo método es limitar el acceso al puerto TCP 24587 del NGFW geográficamente, de esta manera, acotamos los rangos IP que tienen la posibilidad de conectarse a nuestro NGFW, estamos sacando de la ecuación a la mayoría de los países [38] que son utilizados por grupos de cibercriminales como origen para realizar sus ataques. Esto lo realizaremos cuando configuremos el NGFW en el siguiente punto de este proyecto.
El flujo de trafico que se observará en esta conexión será el siguiente:

- Tunel SSH:
    - La pasarela LoRaWAN inicia el establecimiento del túnel hacia la IP:Puerto externo del NGFW.
    - El NGFW realiza un NAT Port Forwanding para ocultar el puerto interno del servicio SSH del contenedor. IP Externa:Puerto Externo → IP Interna:Puerto Interno
    - El trafico llega a la IP del servidor de contenedores y al puerto publicado por el contenedor OpenSSH para el establecimiento de tuneles.
    - El contenedor establece el túnel con la pasarela LoRaWAN y publica un puerto local para realizar el enrutamiento a través del túnel.
- Trafico gestión:
    - El equipo de gestión se conecta a la IP del servidor de contenedores y al puerto publicado por el contenedor SSH para enrutar el trafico al túnel.
    - Cuando el contenedor recibe trafico en el puerto local publicado para el túnel, realiza el reenvió de dicho trafico hacia el túnel SSH.
    - El trafico sale por el túnel con el puerto de túnel interno y el NGFW reenvía este trafico a la IP Publica y puerto externo de la pasarela LoRaWAN.

Hemos seleccionado un contenedor Docker con OpenSSH[33] disponible en GitHub. Generamos un fichero de Docker Compose [Anexo B.2] para el despliegue del contenedor y lo ejecutamos en el servidor de contenedores.
Para configurar esta funcionalidad desde la pagina de administrador "**Remoto Mgmt**" [Anexo C.6], configuramos los parámetros de conexión [Anexo C.7] y generaremos las claves (privada y publica) para la conexión SSH [Anexo C.8]. Como hemos indicado antes, el servicio SSH publicado en Internet por lo NGFW no será el puerto estándar SSH, por lo que deberemos adaptar el script de funcionamiento del servicio SSH de la pasarela LoRaWAN. Para ello añadiremos el parámetro "-p 24587" al final de cada línea de comando del servicio SSH [Anexo B.3] del script ***/etc/init.d/DR_enable_rssh.sh*** de arranque del servicio de "**Remote Mgmt**" para indicar el puerto publicado en le NGFW. También

[33] https://github.com/linuxserver/docker-openssh-server

cambiaremos en la misma línea de comando de este script el puerto de reenvió local, para que el trafico del túnel se envíe al puerto TCP 443 de administración HTTPS del LG-308.

Para autorizar el acceso de conexión de la pasarela al equipo servidor QNAP, descargaremos la clave publica de la pasarela y la incluiremos en el fichero ***autorizated_key*** del servidor QNAP. Modificamos el fichero de script de arranque del servicio SSH del servidor QNAP para permite los túneles SSH modificando el fichero ***/etc/init.d/login.sh*** y añadiendo dos líneas de comando para el script lo añada al fichero de configuración del servicio SSH del servidor QNAP [Anexo B.4] . Realizado esto debemos reiniciar el servicio SSH para que acepte los cambios configurados y empiece a funcionar el modo túnel.

Para probar su funcionamiento, en la pasarela LoRaWAn, en la pagina de "**Remote Mgmt**", pulsamos en el botón "**Connect**" y observamos que se establece el túnel SSH correctamente .

Ahora podemos a proceder a comprobar que desde el equipo de gestión en la red de servicio, podemos conectarnos a la pasarela LoRaWAN utilizando la dirección IP y el puerto TCP publicados localmente en el contenedor OpenSSH, para ello abrimos un navegador en el equipo y apuntamos a la URL https://lora.redesipv6.es:19999 (Hemos publicado en el DNS interno de la red de servicio el registro A "LORA" para que se resuelva a la IP de equipo contenedor) y observamos que se conecta perfectamente al equipo remoto [Anexo C.9] a través de HTTPS.

Una vez que disponemos de gestión remota mediante el túnel SSH, desactivaremos la gestión por HTTP y SSH en el interfaz WAN desactivando las opciones "***Enable HTTP Forward***" y "***Enable SSH Forward***" en la pagina "***System General***" [Anexo C.10].

Y por último nos queda evitar ciertos posibles ataques, como algunos de tipo "*Man-in-the-Midle*" que hemos comentado en el punto 2.2, por lo que hemos revisando varios estudios [18] [39] [40] [41] que tratan este problema, y hemos podido llegar a la conclusión que mediante el uso y configuración del protocolo LoRaWAN en la versión 1.1 en modo OTAA, y gracias a la implementación de un "*Join Server*" que realizada esta versión, nos permitirá securizar y garantizar la seguridad del protocolo LoRaWAN en redes no confiables como Internet para evitar varios tipo de ataques en comparación con versiones anteriores del protocolo LoRaWAN que si son vulnerables. Esta implementación la realizaremos en el siguiente punto de este proyecto a la hora de configurar el entorno de los servidores de gestión de la red LoRAWAN.

4.3. Despliegue seguro de entorno de contenedores

Para un despliegue seguro de un entorno Docker, empezaremos por securizar el equipo sobre el cual ejecutaremos los contenedores, que en nuestro caso es el QNAP TS-451B.

Lo primero que realizaremos será la securización del entorno de gestión del equipo, y para ello realizaremos las siguientes tareas:

- Desactivación de acceso Web mediante puertos no seguros y dejaremos habilitado solo el puerto HTTPS.
- Desactivaremos el puerto de gestión remota inseguro como telnet y dejaremos solo activados los puertos de gestión SSH.
- Generaremos un certificado digital para el equipo.
- Activaremos la autentificación de doble factor para los usuarios de administración.
- Desactivaremos los servicios no esenciales.
- Configuraremos las políticas de red para el control de acceso al equipo.

Actualmente para disponer de un certificado digital firmado por una autoridad de certificación (CA) reconocida no cabe más remedio que realizar un desembolso de dinero periódicamente, dado que cada vez que caduca el certificado hay que generar uno nuevo.

Ha habido algunos intentos de promover el uso de certificado gratuitos como fue el intento de la empresa Thawte a través de su servicio “Web of Trust”, basado en la verificación entre personas (notarios) de la cadena de confianza de los certificados pero desgraciadamente terminaron cerrando el proyecto y no prospero. Pero otro proyecto de iguales características funciona en la actualizada que es CAcert pero sigue teniendo una gran limitación con lo que seria una entidad de certificación que es la no inclusión de sus certificados raíz en los navegadores actuales [42], lo que obliga a la instalación manual estos certificados raíz en los equipos para poder garantizar el correcto funcionamiento de la cadena de validación. Pero el gran salto en el mundo de los certificados digitales gratuito supuso la creación de la entidad de certificación Let’s Encrypt[34]. Let’s Encrypt es una entidad de certificación abierta y gratuita, promovida por Internet Security Research Group (ISRG), basada en el protocolo ACME[35] para la obtención automática de certificados, una opción gratuita para disponer de certificados digitales confiables. En 14 de septiembre de 2015 ISRG envió los certificados raíz de Let’s Encrpyt para su inclusión en la mayoría de los navegadores y sistemas operativos.

El servidor QNAP trae incorporado un cliente ACME para la generación y actualización de los certificados digitales del equipo mediante Let’s Encrypt. Por esto podemos utilizar el servidor QNAP para generar un certificado digital para nuestras aplicaciones Web. Solo debemos tener en cuenta a la hora de generar el certificado en el servidor QNAP, es el incorporar los otros nombres alternativos que deseamos en el certificado al nombre principal del certificado para así además poder utilizar este certificado en otros equipos. De esta manera, una vez obtenido el certificado para el servidor QNAP,

[34] https://letsencrypt.org/about/
[35] https://tools.ietf.org/html/rfc8555

este mismo nos servirá para las pasarelas LoRaWAN, el NGFW y los contenedores Docker que despleguemos y que necesiten un certificado digital.

El único inconveniente que debemos tener en cuenta al utilizar los certificados emitidos por Let's Encrypt es que su caducidad es de 90 días, por lo que hay que establecer una política de mantenimiento adecuada para realizar el proceso de renovación del certificado periódicamente.

Una vez realizada la configuración básica del equipo, que es la dirección IP y la inicialización del almacenamiento, nos podemos conectarnos al equipo a través de su pagina web de administración [Anexo C.11]. A partir de aquí, configuraremos en la sección panel de control los siguientes parámetros de seguridad:

- Permitir solo conexiones de la lista [Anexo C.12]. Se deben añadir todas las redes que deseamos que tengan conexión con el equipo incluidas las redes de los contenedores Docker.
- Bloquear las conexiones fallidas al equipo durante un tiempo determinado [Anexo C.13]. Se bloquearán durante 1 minuto, los 5 intentos de conexión fallidos a los servicios SSH y HTTPS. El resto de los servicios serán desactivados.
- Bloquear las autentificaciones de usuarios fallidas [Anexo C.14]. Se bloqueará el acceso a los usuarios durante 1 minuto, cuando realicen los 5 intentos de conexión fallidos a los servicios SSH y HTTPS. El resto de los servicios serán desactivados.
- Configuraremos las políticas de contraseña con un modelo robusto de seguridad en consonancia con el resto de los equipos de nuestra arquitectura [Anexo C.15].
- Desactivaremos el servicio Telnet y solo dejaremos configurado el servicio SSH en el puerto TCP 57648 [Anexo C.16].
- Cambiaremos los puertos de administración Web GUI (HTTP y HTTPS) y forzaremos la redirección de HTTP a HTTPS, dado que no se puede desactivar el protocolo HTTP en la pagina de administración Anexo [C.17].
- Activaremos el doble factor de autentificación mediante OTP para el usuario administrador del equipo a través de Web GUI. Para ello seguimos el siguiente procedimiento (sirve para cualquier usuario):
    - Una vez iniciada sesión con el usuario que deseamos activar el doble factor OTP, en la esquina superior derecha de la pagina de administración, pulsamos sobre el nombre de usuario y seleccionamos "**Opciones"** [Anexo C.18].
    - Dentro de "**Opciones**", seleccionamos la pestaña "**Verificación en 2 pasos**" [Anexo C.19].
    - Pulsamos el botón de "**Empezar**" y se nos mostrara un código QR que deberemos escanear con nuestra aplicación OTP [Anexo C.20]. En nuestro caso, nos hemos descargamos la aplicación "Network Tools" para IPhone de la empresa americana Hurricane Electrics, que dispone de entre otras utilidades de un OTP integrado.
    - En el apartado OTP de la aplicación, seleccionamos "**Add account**" [Anexo C.21] y seleccionamos la opción de "**Scan Barcode**" [Anexo C.22].

- Escaneamos el código QR de la pagina del QNAP [Anexo C.23] y se nos añade automáticamente una nueva cuenta para generación de OTP [Anexo C.24].
- Ahora debemos verificar en el QNAP que el OTP esta funcionando correctamente por lo cual al pulsar continuar nos aparece una pantalla de Verificación, donde deberemos introducir uno de los códigos generados por la aplicación y pulsaremos "**Verificar**" [Anexo C.25].
- Si todo el proceso ha sido correcto en el siguiente pagina se nos solicita un método alternativo de verificación en caso de no disponer o perder el OTP. Podemos elegir entre una pregunta de seguridad o el envió de un correo electrónico [Anexo C.26].
- Pulsamos "**Finalizar**" y el usuario ya tiene configurado el doble factor de autentificación. Cuando el usuario de nuevo ingrese su usuario y contraseña, se le mostrara una segunda pantalla para introducir el código OTP generado por la aplicación móvil [Anexo C.27].

Realizadas las tareas de securización del servidor QNAP, solo nos queda generar el certificado digital para el equipo y el resto de los componentes del sistema que necesiten un certificado digital. Para la generación del certificado, seguiremos los siguientes pasos:

- En el panel de control, accedemos a las opciones de "**Seguridad**".
- En seguridad accedemos a la pestaña de "Certificado & clave privada" [Anexo C.28].
- Pulsamos la opción de "**Reemplazar certificado**" y en las opciones elegimos "**Obtener desde Let's Encrypt**" [Anexo C.29].
- Al pulsar "**Siguiente**" [Anexo C.30] se nos muestra las opciones para la generación del certificado que son:
  - Nombre de Dominio: Es el CN del certificado a generar.
  - Correo electrónico: Dirección de correo electrónico.
  - Nombre alternativo: Aquí deberemos introducir todos los nombres de dominio que deseamos que soporte el certificado separados por comas.
- Al pulsar "**Aplicar**" se nos mostrara una advertencia indicando que necesitamos que el puerto 80 del servidor Web del QNAP este visible desde Internet para que se pueda realizar la inscripción del certificado. Así mismo es necesario que los nombres de dominio que deseamos que tenga el certificado este publicados por DNS en Internet y apuntando a la dirección IP Publica de nuestro NGFW, donde estará publicado el puerto TCP 80 hacia la dirección IP del QNAP para que se puede realizar la verificación de todos los nombres de dominio del certificado.
- Si todo el proceso se realiza correctamente, se nos mostrara la pagina de "**Certificado & clave privada**" el estado y los datos del certificado [Anexo C.31].

Después de este proceso, ya disponemos de un certificado multidominio para ser utilizando tanto por el QNAP, los contenedores y el NGFW. La clave publica y privada del certificado se encuentran en el fichero /**etc/stunnel/stunnel.pem.**

En este punto, con la seguridad aplicada al equipo que ejecutara los contenedores Docker, procederemos a securizar cada uno de los contenedores necesarios en la arquitectura.

Para la ejecución del entorno LoRaWAN utilizaremos el contenedor Docker de TTN Stack que se compone de los siguientes elementos:

- Gateway Server
- Network Server
- Join Server
- Identity Server
- Application Server

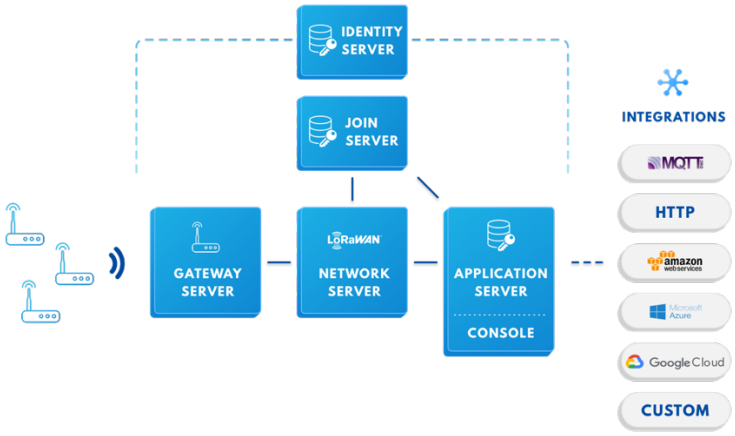


Figura 30: Arquitectura Docker TTN Stack 3.0 [43]

Para la visualización de los datos, utilizaremos los contenedores Node-Red para el tratamiento de los datos. Como base de datos para guardar toda la información recibida de los sensores utilizaremos MySQL y phpMyAdmin para la gestión de esta base de datos.

Por ultimo, utilizaremos un contenedor de Grafana como Front-End para la visualización de los datos de los sensores.

Para la ejecución de estos contenedores utilizaremos un único script de Docker Compose para la configuración automática de toda la aplicación, excepto el contenedor de túneles SSH que por razones de seguridad se mantendrá en un contenedor aislado del resto de aplicactivos. El script del de Docker Compose para se incluye en el Anexo B.1

Todos estos contenedores de la red LoRaWAN estarán conectados internamente mediante una única red aislada del servidor QNAP. El contenedor de túneles SSH dispondrá de su propia red privada y aislada del resto. Solo estarán publicados los siguientes puertos de los contenedores mediante NAT para acceso de los servicios externos:

- UDP 1700 para la recepción de mensajes desde las pasarelas LoRaWAN.
- TCP 24587 para los túneles SSH de administración de las pasarelas LoRaWAN
- TCP 1080 para la administración de la base de datos mediante phpMyAdmin.
- TCP 1880 para la gestión de Node-Red
- TCP 4000 para la pagina Web de Grafana.

Antes de configurar HTTPS en los contenedores debemos obtener el certificado generado anteriormente a través de Let's Encrypt para poder utilizarlo en los contenedores.
Para ello debemos seguir los siguientes pasos:

- Accedemos por SSH al QNAP.
- El certificado digital en formato PEM se encuentra en el fichero /etc/stunnel/stunnel (clave privada y publica).
- Copiamos la parte de la clave privada en un fichero llamado key.pem.
- Copiamos la parte de la clave publica en un fichero llamado cert.pem
- Utilizamos estos dos ficheros para cargar el certificado los contenedores.

Para la securización de cada contenedor seguiremos los siguientes pasos:

- Contenedor TTN Stack
    - Desactivar HTTP: Eliminamos del script el puerto 1885 del script **docker-compose.yml** [Anexo B.1].
    - Cargar certificados para HTTPS: Configuramos la ubicación de los certificados digitales en el fichero de configuración **ttn-lw-stack-docker.yml** [Anexo B.5].
    - Cambiar contraseña por defecto de administración: Durante el despliegue inicial, seleccionaremos una contraseña robusta acorde con las políticas de contraseñas ya utilizadas.
    - Desactivar servicios externos no esenciales: Eliminaremos del script **docker-compose.yml** toda la referencia de puertos que no sean los estrictamente necesarios.
    - Cambiamos todas las contraseñas por defecto del fichero de configuración **ttn-lw-stack-docker.yml** aunque no sean necesarias.
    - Comprobamos que accedemos a la pagina HTTPS de TTN Stack y su autentificación Anexo [C.32].
- Contenedor Node-Red
    - Activamos HTTPS: Añadimos al fichero de configuración **settings.js** los parámetros de ubicación de los ficheros de certificados digitales para HTTPS [Anexo B.6].
    - Activamos la autentificación para el acceso a la administración Web GUI: Generamos el hash de la contraseña del administrador a través de la consola del contenedor con el comando **node -e "console.log(require('bcryptjs').hashSync(process.argv[1], 8));"** ***password*** , configuramos y agregamos este hash en el fichero de configuración.
    - Observamos que la pagina inicial de Node-Red funciona solo con HTTPS y que nos solicita la autentificación para poder acceder [Anexo C.33].
- Contenedor MySQL
    - Contraseña por defecto: Debemos cambiar la contraseña del usuario **root**, por una contraseña robusta según las políticas de seguridad anteriormente descritas.
    - Proteger usuario **root**: Configuraremos un secreto Docker para la contraseña del usuario **root** de MySQL en el fichero **docker-compose.yml**.

- Contenedor PHPmyAdmin
    - Debido a que la versión oficial del contenedor no soporta HTTPS, deberemos reconfigurar la imagen del contenedor oficial de phpMyAdmin para que soporte HTTPS. Para ello encontramos una guía [44] que nos muestra como configurar un Dockerfile [Anexo B.7] para incorporar en la imagen los parámetros necesarios para que funcione HTTPS. Después de realizar la configuración de HTTPS, podemos comprobar que nos conectamos al servidor Web de phpMyAdmin por HTTPS en el puerto 5443 [Anexo C.34].
    - Desactivar HTTP: Eliminamos el puerto 1080 del script **docker-compose.yml.**
    - Configuramos **Docker secret** para la conexión a la base de datos: Agregamos en el fichero **docker-compose.yml** un secreto para la conexión al MySQL.
    - Activamos el doble factor de autentificación: Para el acceso a phpMyAdmin del usuario ***root*** activaremos el doble factor de autentificación en la pagina de "**Configuración → Autentificación de dos factores**" y seleccionando en "**Solicitud de autentificación (2FA)**" [Anexo C.35]. Escaneamos con la aplicación OTP que ya hemos utilizado anteriormente el código QR y verificamos el correcto funcionamiento introduciendo en "**Authentication code**" un código OTP generado por la aplicación móvil [Anexo C.36]. Después comprobamos que al iniciar sesión en phpMyAdmin se nos solicita un código OTP para validar el inicio de sesión [Anexo C.37].
- Contenedor Grafana
    - Contraseña por defecto: Al iniciar por primera vez la aplicación Grafana, debemos cambiar la contraseña del usuario **admin**, por una contraseña robusta según las políticas vistas anteriormente.
    - Activamos HTTPS: En el fichero de configuración **grafana.ini**, activamos el protocolo HTTPS en lugar del HTTP y añadimos los parámetros de ubicación de los ficheros de certificados digitales para HTTPS [Anexo B.8]. Al acceder a la URL mediante HTTPS, podemos comprobar que la pagina funciona correctamente [Anexo C.38]
- Contenedor OpenSSH
    - Acceso solo mediante clave SSH: Configuramos el fichero **docker-compose.yml** [Anexo B.9] del contenedor para solo aceptar autentificación mediante clave publica.

El resto de los contenedores, Redis y CockroachDB como son de uso interno y no están expuesto a Internet no realizaremos su configuración segura, aunque siempre será recomendable hacerla.
En este momento podemos decir, que todos los elementos que componen la gestión de los datos de nuestra red LoRaWAN están protegidos y adecuadamente securizados.

4.4. Implementación segura de NGFW en Internet

Para proteger la red de servicio de las amenazas de Internet, se deberá configurar las políticas de seguridad en el NGFW para garantizar un entorno seguro y confiable.

Dividiremos las políticas de seguridad según la zona o interface sobre la que se aplique. Si la políticas afecta a la red de servicio, las políticas de seguridad se aplicaran a la zona INTERNA o al interfaz LAN. Si la política de seguridad debe controlar el trafico que se recibe desde Internet y que fluye hacia la red de servicio, se aplicara a la zona DMZ o al interfaz WAN.

La primera fase de la securización es realizar el bastionado del propio NGFW, dado que su exposición a Internet le hace objeto de posibles ataques y amenazas.

Damos por realizado la configuración inicial del equipo, pudiendo acceder a su gestión a través del interfaz LAN.

La primera tarea será actualizar la versión del firmware a la ultima versión recomendada para cumplir con los requisitos de seguridad FIPS-CC si deseamos aplicar este requerimiento, sino se debe instalar la ultima versión de firmware recomendada por el fabricante que este libre de vulnerabilidades. Para ello nos descargaremos la versión 5.6.7-fips disponible en la pagina de soporte del fabricante para el equipo y procedemos con los siguientes pasos:

- Nos conectamos al interfaz grafico de gestión Web a través de la IP del interfaz LAN que es el único que tiene permitido el acceso de administración Web.
- En el menu **System** seleccionamos la opción sobre **Firmware** y seleccionamos el fichero para actualizar el equipo y pulsamos **backup config and upgrade** [Anexo C.39].
- El equipo se actualizará y se reiniciará en la nueva versión seleccionada.

En la guía de empleo seguro de Cortafuegos Fortigate del CCN que seguiremos para bastionar el equipo, se indica la necesidad de configurar la operación en modo seguro FIPS-CC, para lo cual es necesario que la primera configuración que se realice sea esta, dado que algunas configuraciones podrán ser borradas. Para activar el modo FIPS-CC en el NGFW haremos lo siguiente:

- Accedemos por SSH a la gestión mediante línea de comando (CLI).
- Entramos en modo configuración global.
- Activamos FIPS-CC.
- Nos solicita una nueva contraseña de administrador.
- El equipo se reiniciará y funcionará en modo seguro FIPS-CC.

A partir de este punto, ya podemos realizar el resto de configuración del equipo para su correcto funcionamiento.

Antes de configurar HTTPS en el equipos deberemos cargar un nuevo certificado SSL para sustituir al certificado autofirmado incorporado de fabrica.

Desgraciadamente, FortiNet actualmente no incluye soporte para la obtención de certificados de Let's Encrypt, pero utilizaremos el certificador generado por el servidor QNAP y manualmente lo podemos instalar en el NGFW.

Los pasos para instalar un certificado digital en el NGFW son los siguientes:

- Accedemos por la pagina de administración Web al menú **System** y pulsamos en la opción **Certificate**.
- Pulsamos sobre la opción **Import** y seleccionamos **Local certificate**.
- En tipo de importación seleccionamos **Certificate**, añadimos los ficheros con la clave publica y privada y aceptamos para que se carguen los certificados [Anexo C.40].
- Accedemos mediante una URL segura la NGFW para comprobar que funciona correctamente [Anexo C.41]

Ahora que ya disponemos de un certificado digital, empezaremos por configurar los interfaces de red. El interfaz LAN aunque previamente ha sido configurado para poder gestionar inicialmente el equipo, comprobaremos dicha configuración y la adaptaremos para un modo seguro, desactivando todos los servicios de gestión y dejando únicamente la gestión mediante HTTP y SSH [Anexo C.42].

En la configuración del interfaz WAN desactivaremos todos los servicios de gestión, para evitar exponer estos a Internet y activamos el bloqueo de conexiones de redes Botnet [Anexo C.43] y configuraremos su asignación de IP mediante PPPoE. Para ello previamente en el router FFTH se ha configurado en modo bridge (algunos operadores lo permiten desde el router FFTH [45] o conectando directamente el ONT al NGFW) . Esto permitirá que la IP Publica este asignada en el propio interfaz WAN del Router para un mayor control (evitamos los ataques al router FTTH).

Configuraremos los parámetros generales del equipo de gestión. Para una políticas segura de gestión y de contraseñas, configuraremos los siguientes parámetros [Anexo C.44] en el menú **System → Settings**:

- Longitud mínima de las contraseñas: 10 caracteres.
- Histórico de contraseña : 5 contraseñas anteriores.
- Complejidad: Como mínimo una letra minúscula, una mayúsculas, un numero y un carácter especial.
- Cambio de contraseña: Cada 60 días.
- Cambiamos puerto por defecto HTTPS: Configuramos el puerto 44300
- Cambiamos puerto por defecto SSH: Configuramos el puerto 10220
- Cambiamos puerto por defecto Telnet: Configuramos el puerto 20230

En la guía de uso seguro del CCN, se recomienda configurar la autentificación de administración con un certificado de cliente para disponer de un doble factor de autentificación, pero en este caso configuraremos un doble factor de autentificación por OTP que es más sencillo y seguro. Para ello desde el menú **System → Administrators**, seleccionamos el usuario administrador que queremos configurar, le activamos el doble factor de autentificación y seleccionamos el FortiToken que le asignamos al usuario [Anexo C.45]. Este FortiToken es una licencia que permite al usuario poder utilizar OTP. En nuestro NGFW incorporara dos FortiToken gratuitos. FortiNet incluye su propia aplicación para gestionar y generar los códigos OTP llamada FortiToken. Recibimos un correo con el código QR para escanearlo con la aplicación y ya tenemos disponibles los códigos OTP para el inicio

de sesión [Anexo C.46]. Ahora cuando introducimos la contraseña de inicio de sesión del usuario **admin** además se nos solicita el código OTP [Anexo C.47].

El resto de las configuraciones de seguridad las realizaremos por CLI según se muestra en el Anexo B.10.

En la segunda fase crearemos las políticas de seguridad necesarias para el correcto funcionamiento del entorno de sensores.

- Políticas de seguridad de red de servicio.
    - LAN → INTERNET
        - Acceso QNAP a Internet (HTTP, HTTPS, DNS , NTP) [Anexo C.48]. Esta política permite al QNAP acceder a Internet para servicios básicos como la sincronización de la hora, consultas de nombres de dominio y actualización de firmwave.
- Políticas de seguridad de Internet.
    - INTERNET → LAN
        - Acceso pasarela LoRaWAN a Gateway Server: Esta política permite a las pasarelas LoRaWAN acceder al Gateway Server para trasmitir las tramas LoRaWAN. Para ello debemos crear una política Virtual IP para el reenvió de trafico de Internet a la red interna [Anexo C.49] y asociarlo a una política que permita el trafico entrante [Anexo C.50].
        - Acceso frontal Web Grafana: Permite la conexión a los usuario para consultar los datos de localización de los sensores desde Internet. También debemos crear un Virtual IP [Anexo C.51] y una política asociada a ella [Anexo C.52].
        - Acceso túneles SSH: Mediante esta política, se permite que las pasarelas LoRaWan se conecten al contenedor OpenSSH de terminación de túneles SSH. Creamos un Virtual IP para redirigir el puerto externo al puerto interno del contenedor SSH [Anexo C.53] y creamos la política de seguridad para permitir el trafico entrante [Anexo C.54].
    - Restricción de acceso por Geolocalización: Modificaremos ambas política para indicar que solo se permitirá la conexión a través de ellas a los rangos IP's que se ubiquen en España. Primero configuraremos un objeto de tipo **Address** y de tipo **Geography**, y en Country seleccionaremos **Spain** [Anexo C55]. Ahora modificamos las dos políticas anteriores, cambio el **Source → all** por **Source → SPAIN_COUNTRY** [Anexo C.56].
    - Protección contra intrusos IPS: Para proteger los dos servicios publicados en Internet (SSH y HTTPS) de posibles ataques, crearemos dos políticas de IPS y se asignara cada una a su correspondiente política de seguridad según su protocolo. La primera política IPS será para el protocolo SSH utilizado para el túnel de gestión remota de las pasarelas LoRaWAN [Anexo C.57] y la segunda para proteger el servidor Web de visualización de datos [Anexo C.58]. Una vez creadas la política IPS, deberemos crear

una política de inspección SSL/SSH dado que será necesario romper los túneles cifrados para que el NGFW pueda ver el trafico sin cifrar y buscar entre las firmas de IPS posibles ataques. En las políticas de **SSL/SSH inspection** crearemos un perfil con el certificado del NGFW para inspección SSL y ajustamos los puertos HTTPS y SSH a los de nuestra arquitectura [Anexo C.59]. Después a las políticas de seguridad las asignaremos a su correspondiente política de IPS y de inspección SSL/SSH [Anexo C.60].

- Políticas de seguridad de VPN.
    - INTERNET → VPN
        - Acceso a red de servicio: Para poder acceder a los datos de la base de datos, será necesario conectarse mediante VPN al NGFW para de esta manera tener acceso a la red de servicio. Primero configuramos el acceso mediante VPN SSL, donde configuraremos el puerto 4430 para el acceso VPN SSL y evitar conflictos con el puerto 443 del servicio Web de Grafana [Anexo C.61], luego creamos los usuarios VPN y les asignamos una autentificación de doble factor [Anexo C.62] y por último creamos la políticas de seguridad [Anexo C.63]. Para conectarnos a la VPN necesitamos el cliente VPN de Fortinet disponible gratuitamente. Configuramos la conexión del cliente con la URL **vpn.redesipv6.es** y el puerto 4430. Realizamos una prueba de la conexión y podemos observar como después de introducir el usuario y contraseña de acceso se nos solicita el código OTP [Anexo C.64], que obtenemos mediante la aplicación FortiToken, conseguimos conexión con la red de servicio y tenemos acceso al servidor Web phpMyAdmin para la gestión de la base de datos de los sensores.
- Políticas contra denegación de acceso (DoS)
    - Política de protección contra ataques de Internet: para evitar ataques de DoS en nuestro interfaz de red WAN, configuraremos una política de DoS, ajustando los parámetros de limite a un 20% de los valores por defecto [Anexo C.65].
- Parámetros generales que se habilitaran en todas las políticas:
    - Opciones de registros: Se habilitará el registro de todo el trafico que pase por las políticas y se generará un log cuando la sesión se inicie [Anexo C.66].

Una vez configuradas las políticas de seguridad, nuestro NGFW se encuentra en un modo de funcionamiento seguro y ofreciendo una protección a las conexiones de red que fluyen a través de él.

5. Conclusiones y líneas de futuro

5.1. Conclusiones

Durante el desarrollo de este proyecto, hemos podido constatar la gran cantidad de opción de seguridad que se van incorporando a aplicaciones y sistemas, incluso en equipos de gama baja, pero siendo necesario un esfuerzo inicial en si aplicación, dado que muchas de estas mejoras no son operativas por defecto y teniendo que se configuradas adicionalmente.

De esta manera, la incorporación de sistemas de doble factor de autentificación es una mejora muy buena con respecto a la seguridad, al separar el proceso de autentificación en dos partes independientes, una lógica o mental como es recordar la contraseña y otra física como es el disponer de una aplicación independiente que genera códigos de verificación, pero que observamos nunca son obligatorios sino que son configuraciones opcionales con lo que se delega los criterios de seguridad al buen hacer de los ingenieros y/o profesionales que despliegan estos sistemas.

Lo mismo sucede con el uso de HTTPS para el cifrado de la información que circula por redes no seguras y donde hemos constatado que siempre es una opción configurable y muy pocas veces activada por defecto.

Pero aun así, también podemos concluir que la seguridad esta muy presente en todos los sistemas utilizados en este proyecto, solo siendo necesario un buen conocimiento de los requisitos de seguridad necesarios para desplegar redes seguras y un buen conocimiento de los productos para su correcta configuración.

5.2. Líneas de futuro

Como líneas de futuro trabajos, podíamos ahondar más en los trabajos de securización dentro de los propios contenedores y en como establecer mecanismos de confianza con los repositorios de imágenes de contenedores basándonos en blockchain.

Bibliografía

[1] «Waterfall: metodología para el desarrollo secuencial de tarea,» 2017. [En línea]. Available: https://www.ticportal.es/glosario-tic/waterfall-metodologia-desarrollo-secuencial.

[2] SWD(2016)110, «Advancing the Internet of Things in Europe,» 2016. [En línea]. Available: https://data.consilium.europa.eu/doc/document/ST-8100-2016-ADD-1/en/pdf.

[3] Accent System, «Las “vacas conectadas” son el futuro de la industria ganadera,» 28 03 2018. [En línea]. Available: https://accent-systems.com/es/blog/vacas-conectadas-futuro-ganaderia/.

[4] «Coverage SigFox,» 2020. [En línea]. Available: https://www.sigfox.com/en/coverage.

[5] Vodafone, «Cobertura NB-IoT,» Vodafone, 2020. [En línea]. Available: https://www.vodafone.es/conocenos/es/vodafone-espana/mapa-de-cobertura/consulta-de-cobertura-movil?page_label=co_cobertura_y_tiendas_home.

[6] «Subvenciones destinadas a financiar proyectos empresariales dirigidos a favorecer la incorporación de las tecnologías de la información y la comunicación en las PYMES,» Junta de Castilla y Leon, 2016. [En línea]. Available: https://www.tramitacastillayleon.jcyl.es/web/jcyl/AdministracionElectronica/es/Plantilla100DetalleFeed/1251181050732/Ayuda012/1284618770219/Propuesta.

[7] IBM Cloud Education, «Containerization,» 15 05 2019. [En línea]. Available: https://www.ibm.com/cloud/learn/containerization.

[8] K. Clark y C. Jackson , «The true benefits of moving to containers, Part 1,» 10 09 2020. [En línea]. Available: https://developer.ibm.com/articles/true-benefits-of-moving-to-containers-1/.

[9] INCIBE, «Seguridad en la instalación y uso de dispositivos IoT,» 2020. [En línea]. Available: https://www.incibe.es/sites/default/files/contenidos/guias/doc/guia-de-seguridad-iot.pdf.

[10] Firmware, Intro to Hardware Hacking - Dumping your First, «Nvisium,» 2019. [En línea]. Available: https://blog.nvisium.com/intro-to-hardware-hacking-dumping-your-first-firmware.

[11] «Nebra IP67 Waterproof/Weatherproof Enclosure,» Pi Supply, 2020. [En línea]. Available: https://uk.pi-supply.com/products/die-cast-outdoor-weatherproof-enclosure.

[12] High-Performance Potting Compound Chemistries for Electronics Protection, «Electrolube,» 2018. [En línea]. Available: https://www.electrolube.com.au/technical-articles/high-performance-potting-compound-chemistries-for-electronics-protection/.

[13] Hardware Security in IoT Devices with Emphasis on Hardware Trojans, «Journal of Sensor and Actuator Networks — Open Access Journal,» 2019. [En línea]. Available: https://www.mdpi.com/2224-2708/8/3/42/htm.

[14] S. J. Johnston, M. Scott y S. Cox, «Recommendations for securing Internet of Things devices using commodity hardware,» de *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, Reston, 2016.

[15] LoRa Alliance, «LoRaWAN What is it?,» 2015. [En línea]. Available: https://lora-alliance.org/sites/default/files/2018-04/what-is-lorawan.pdf.

[16] J. Navarro-Ortiz, N. Chinchilla-Romero y J. J. Ramo, «Improving Hardware Security for LoRaWAN,» de *IEEE Conference on Standards for Communications and Networking (CSCN)*, Granada, 2019.

[17] LoRa Alliance, «About LoRaWan,» [En línea]. Available: https://lora-alliance.org/about-lorawan.

[18] J. Thomas, S. Cherian, S. Chandran y V. Pavithran, «Man in the Middle Attack Mitigation in LoRaWAN,» de *Fifth International Conference on Inventive Computation Technologies (ICICT-2020)*, London, UK, 2020.

[19] N. Fishbein y M. Kajilot, «Watch Your Containers: Doki Infecting Docker Servers in the Cloud,» 06 07 2020. [En línea]. Available: https://www.intezer.com/container-security/watch-your-containers-doki-infecting-docker-servers-in-the-cloud/.

[20] Intezer, «Best Practices for Securing a Docker Runtime Environment,» 17 06 2020. [En línea]. Available: https://www.intezer.com/blog/container-security-blog/best-practices-for-securing-a-docker-runtime-environment/.

[21] M. Souppaya, J. Morello y K. Scarfone, «Application Container Security Guide,» 2017. [En línea]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-190.pdf.

[22] D. Mulgund, «8 Best Practices for Application Container Security,» 12 06 2019. [En línea]. Available: https://securityintelligence.com/posts/8-best-practices-for-application-container-security/.

[23] WIkipedia, «Principio del mínimo privilegio,» [En línea]. Available: https://es.wikipedia.org/wiki/Principio_de_m%C3%ADnimo_privilegio.

[24] «Next-Generation Firewall (NGFW),» FortiNet, 2020. [En línea]. Available: https://www.fortinet.com/products/next-generation-firewall#overview1.

[25] Wikipedia, «Zero-day (computing),» 16 09 2020. [En línea]. Available: https://en.wikipedia.org/wiki/Zero-day_(computing).

[26] Microchip, «megaAVR Data Sheet,» 2020. [En línea]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061B.pdf.

[27] Microchip, «8-bit Microcontroller with 16/32/64KB In-System Programmable Flash,» 05 2020. [En línea]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/ATmega640-1280-1281-2560-2561-Datasheet-DS40002211A.pdf.

[28] ESP32 Series Datasheet, «ESP32 Series Datasheet,» 2020. [En línea]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf.

[29] «Secure Boot & Flash Encryption,» 2016. [En línea]. Available: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/security/secure-boot-v1.html#secure-boot-flash-encryption.

[30] «Hardening of your OpenWrt device,» 2019. [En línea]. Available: https://openwrt.org/docs/guide-user/security/openwrt_security.

[31] Gartnet, «Magic Quadrant for Network Firewalls,» 17 09 2019. [En línea]. Available: https://www.gartner.com/doc/reprints?id=1-64R1AGJ&ct=190118&st=sb.

[32] Skerrett, Ian;, «Open Source IoT @ Eclipse,» Eclipse Foundation , 2017.

[33] K. Gabrial, «TTGO T-Beam Tracker for The Things Network,» 2020. [En línea]. Available: https://github.com/kizniche/ttgo-tbeam-ttn-tracker.

[34] «ESP-IDF Programming Guide,» Espressif System, 2020. [En línea]. Available: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/index.html.

[35] «OpenSSL,» OpenSSL Software Foundation., 2020. [En línea]. Available: https://www.openssl.org/.

[36] «Qué es CGNAT y por qué se utiliza,» MásMóvil, 2020. [En línea]. Available: https://blog.masmovil.es/que-es-tecnologia-cgnat-masmovil/.

[37] «Remote Tunnels,» SSH Communications Security Corp., 2011. [En línea]. Available: https://www.ssh.com/manuals/server-zos-admin/61/Remote_Tunnels.html.

[38] «Estado de Internet / Seguridad 2019,» Akamia, 2019. [En línea]. Available: https://www.akamai.com/es/es/multimedia/documents/state-of-the-internet/soti-security-a-year-in-review-report-2019.pdf.

[39] M. Eldefrawy, I. Butun, N. Pereira y M. Gidlund, «Formal Security Analysis of LoRaWAN,» 2018. [En línea]. Available: http://miun.diva-portal.org/smash/get/diva2:1269488/FULLTEXT01.pdf.

[40] G. Bernardinetti, F. Mancini y G. Bianchi, «Disconnection Attacks Against LoRaWAN 1.0.XABP Devices,» de *Mediterranean Communication and Computer Networking Conference* , Online Conference, 2020.

[41] S. Wesemeyer, I. Boureanu, Z. Smith y H. Treharne, «Extensive Security Verification of the LoRaWAN Key-Establishment:Insecurities & Patches,» de *IEEE European Symposium on Security and Privacy*, Virtual Event, 2020.

[42] «CAcert.org,» Wikipedia, 2020. [En línea]. Available: https://es.wikipedia.org/wiki/CAcert.org.

[43] «Components TTN Stack,» The Things Network, 2020. [En línea]. Available: https://thethingsstack.io/reference/components/.

[44] V. Thomas Pronold, «phpMyAdmin Docker Image with SSL/TLS,» 23 02 2020. [En línea]. Available: https://blog.zotorn.de/phpmyadmin-docker-image-with-ssl-tls/.

[45] «Tutorial para configurar HGU ASKEY en modo BRIDGE,» Movistar, 2016. [En línea]. Available: https://comunidad.movistar.es/t5/Soporte-Fibra-y-ADSL/Tutorial-para-configurar-HGU-ASKEY-en-modo-BRIDGE/td-p/2803022.

Anexo

Anexo A: Glosario

ABS: Acrylonitrile Butadiene Styrene

ACME: Automated Certificate Management Environment

CCN: Centro Criptologico Nacional

CEO: Chief Executive Officer

CG-NAT: Carrier Grade - Network Address Translation

CPU: Central process Unit

DoS: Denial of Service

DMZ: Demilitarized Zone.

DNS: Domain Name System

FIPS-CC: Federal Information Processing Standard – Common Criteria

FTTH: Fiber To The Home

GB: Gigabytes

GPS: Global Positioning System

GUI: Graphics User Interface

HDD: Hard Disk Driver

HTTP: Hypertext Transfer Protocol.

HTTPS: Hypertext Transfer Protocol Security

IP: Internet Protocol

IPSEC: Intenet Protocol Security

LAN: Local Area Network

LPWAN: Low Power Wide Area Network

MAC: Media Access Control

NAT: Network Address Translation

NAT-T: Network Address Translation Traversal

NGFW: New Generation FireWall

NTP: Network Time Protocol

MCU: Micro Controller Unit

MQTT: Message Queing Telemetry Transport

ONT: Optical Network Terminal

OTP: One Time Password

SMB: Server Message Block

SoC: System on Chip

SOC: Security Operations Center

SSH: Secure Shell

SSL: Secure Sockets Layer

TB: Terabyte

TCP: Transmission Control Protocol

TIC: Tecnologías de la Información y la Comunicación

TLS: Transport Layer Security

TTL: Time To Live

UDP: User Datagram Protocol

URL: Uniform Resource Locator

UV: Ultraviolet

VPN: Virtual Private Network

WAN: Wide Area Network

Anexo B: Entregables del proyecto

1. Fichero docker-compose.yml del contenedor principal.

```
version: '3.7'
services:
  cockroach:
    image: cockroachdb/cockroach:latest
    command: start-single-node --http-port 26256 --insecure
    restart: unless-stopped
    volumes:
      - ${DEV_DATA_DIR:-.env/data}/cockroach:/cockroach/cockroach-data
    ports:
      - "127.0.0.1:26257:26257" # Cockroach
  redis:
    image: redis:latest
    command: redis-server --appendonly yes
    restart: unless-stopped
    volumes:
      - ${DEV_DATA_DIR:-.env/data}/redis:/data
    ports:
      - "127.0.0.1:6379:6379"
  stack:
    image: thethingsnetwork/lorawan-stack:latest
    entrypoint: ttn-lw-stack -c /config/ttn-lw-stack-docker.yml
    command: start
    restart: unless-stopped
    depends_on:
      - redis
      - cockroach
    volumes:
      - ./blob:/srv/ttn-lorawan/public/blob
      - node-red-data:/data
      - '/share/MD0_DATA/Container/container-station-data/config:/config:ro'
    environment:
      TTN_LW_BLOB_LOCAL_DIRECTORY: /srv/ttn-lorawan/public/blob
      TTN_LW_CONSOLE_OAUTH_TOKEN_URL: http://ttn.redesipv6.es:8885/oauth/token
      TTN_LW_REDIS_ADDRESS: redis:6379
      TTN_LW_IS_DATABASE_URI: postgres://root@cockroach:26257/ttn_lorawan?sslmode=disable
    ports:
      - "443:8885"
      - "1700:1700/udp"
    secrets:
       - ./cert/ttn/cert.pem
       - ./cert/ttn/key.pem
  db:
    image: mysql
    command: --default-authentication-plugin=mysql_native_password
    restart: always
    environment:
```

```
      MYSQL_ROOT_PASSWORD: /run/secrets/secret_mysql
    secrets:
      - secret _mysql
  phpmyadmin:
    image: phpMyAdmin
    build:
      context: .
      dockerfile: Dockerfile
    restart: always
    ports:
      - 5443:443
    environment:
      - PMA_ARBITRARY=1
      - PMA_PASSWORD_FILE=/run/secrets/db_root_pass
    volumes:
      - ./cert:/cert
    secrets:
       - db_root_pass
  node-red:
    image: nodered/node-red:latest
    environment:
      - TZ=Europe/Amsterdam
    ports:
      - "1880:1880"
    links:
      - db
    volumes:
      - node-red-data:/data
  grafana:
    image: grafana/grafana:latest
    user: "$UID:$GID"
    ports:
      - 4000:4000
    volumes:
      - grafana:/var/lib/grafana
      - grafana_conf:/etc/Grafana
volumes:
  grafana:
  grafana_conf:
  node-red-data:
    secrets:
      file: ./ca.pem
      file: ./cert.pem
      file: ./key.pem
secrets:
   secret_mysql:
     file: ./secret/mysql.txt
   db_root_pass:
     file: ./secret/phpmyysql.txt
    cert.pem:
```

```
      file: ./secret/ert.pem
    key.pem:
      file: ./secret/key.pem
```

2. Fichero docker-compose.yml del contenedor OpenSSH

```
version: '3'
services:
  openssh-server:
    image: ghcr.io/linuxserver/openssh-server
    container_name: openssh-server
    hostname: openssh-server #optional
    environment:
      - PUID=1000
      - PGID=1000
      - TZ=Europe/Madrid
      - PUBLIC_KEY_FILE=./ect/ssh_host_keys/ssh_host_rsa_key.pub #optional
      - SUDO_ACCESS=false #optional
      - PASSWORD_ACCESS=false #optional
      - USER_PASSWORD=password #optional
      #- USER_PASSWORD_FILE=/path/to/file #optional
      - USER_NAME=admin
    volumes:
      - ./etc:/config
    ports:
      - 2222:2222            # Puerto túnel red de servicio
      - 19999:19999          # Puerto interno administración pasarela LORA-01
      - 19998:19998          # Puerto interno administración pasarela LORA-02
      - 19997:19997          # Puerto interno administración pasarela LORA-03
      - 19996:19996          # Puerto interno administración pasarela LORA-04
   restart: unless-stopped
```

3. Fichero /etc/init.d/DR_enable_rssh.sh

```
#!/bin/sh /etc/rc.common
START=99
start()
{
        host_id="$(uci -q get rssh.rssh.host_id)"
        host_addr="$(uci -q get rssh.rssh.host_addr)"
        host_port="$(uci -q get rssh.rssh.host_port)"
        killall ssh
        sleep 2
        # Connect to server
        # Modificamos el Puerto de reenvió del túnel y de conexión remota
```

```
        ssh -y -i /etc/dropbear/id_dropbear -o "ExitOnForwardFailure yes" -f -N -T -R
$host_port:localhost:443 -K 60 $host_id@$host_addr -p 24587 > /dev/null
        # Check for successful connection
        sleep 2
        check_connect=$(ps | grep -c -e "ssh -y -i /etc/dropbear/id_dropbear")
        manual_connect=$(uci get rssh.rssh.manual_connect)

        # If connect failed, and it is an auto connect, start background task to
        # wait for server to drop prior connection, then connect.
        if [ $check_connect != "2" ] && [ $manual_connect == "0" ]; then
                # Modificamos el Puerto de reenvió del túnel y de conexión remota
                (sleep 300; \
                ssh -y -i /etc/dropbear/id_dropbear -o "ExitOnForwardFailure yes" -f -N -T -R
$host_port:localhost:443 -K 60 $host_id@$host_addr -p 24587 > /dev/null) &
        fi
        # Reset manual connect flag
        uci set rssh.rssh.manual_connect="0"
        uci commit rssh
}
stop()
{
echo "Manual disconnect" > /tmp/date.txt
        uci set rssh.rssh.manual_connect="0"
        uci commit rssh
        killall ssh
}
```

4. Fichero /etc/init.d/login.sh

```
SSH=/usr/sbin/sshd
TELNET=/bin/utelnetd
SSHD_CONF=/etc/config/ssh/sshd_config
SSHD_CONF_DEFAULT=/etc/ssh/sshd_config
/sbin/test -f $SSHD || exit 0
/sbin/test -f $TELNET || exit 0
[ -f "/bin/cmp" ] || ln -sf /bin/busybox /bin/cmp
SSH_PORT=`/sbin/getcfg LOGIN "SSH Port" -d 22`
DEAFULT_TELNET_PORT=`/sbin/getcfg -f /var/default LOGIN "TELNET Port" -d 13131`
TELNET_PORT=`/sbin/getcfg LOGIN "TELNET Port" -d $DEAFULT_TELNET_PORT`
SSHKEY_CONFIG_DIR=/etc/config/ssh
BOOT_CONF=`/bin/cat /etc/default_config/BOOT.conf`
generte_ssh_key()
{
       [ -d $SSHKEY_CONFIG_DIR ] || /bin/mkdir $SSHKEY_CONFIG_DIR
       if [ -f /usr/bin/ssh-keygen ]; then
              if [ ! -f ${SSHKEY_CONFIG_DIR}/ssh_host_rsa_key ] || [ ! -f ${SSHKEY_CONFIG_DIR}/ssh_host_rsa_key.pub ]; then
                     /bin/rm -f ${SSHKEY_CONFIG_DIR}/ssh_host_rsa_key*
                     /usr/bin/ssh-keygen -t rsa -f ${SSHKEY_CONFIG_DIR}/ssh_host_rsa_key -N ""
                     /bin/touch /etc/config/ssh_key.fla
                     /bin/sync
              fi
              if [ ! -f ${SSHKEY_CONFIG_DIR}/ssh_host_dsa_key ] || [ ! -f ${SSHKEY_CONFIG_DIR}/ssh_host_dsa_key.pub ]; then
                     /bin/rm -f ${SSHKEY_CONFIG_DIR}/ssh_host_dsa_key*
                     /usr/bin/ssh-keygen -t dsa -f ${SSHKEY_CONFIG_DIR}/ssh_host_dsa_key -N ""
                     /bin/touch /etc/config/ssh_key.fla
```

```
                        /bin/sync
                fi
                /bin/cmp /etc/ssh/ssh_host_rsa_key ${SSHKEY_CONFIG_DIR}/ssh_host_rsa_key 1>>/dev/null 2>>/dev/null
                retrsa1=$?
                /bin/cmp /etc/ssh/ssh_host_dsa_key.pub ${SSHKEY_CONFIG_DIR}/ssh_host_dsa_key.pub 1>>/dev/null 2>>/dev/null
                retrsa2=$?
                /bin/cmp /etc/ssh/ssh_host_dsa_key ${SSHKEY_CONFIG_DIR}/ssh_host_dsa_key 1>>/dev/null 2>>/dev/null
                retdsa1=$?
                /bin/cmp /etc/ssh/ssh_host_dsa_key.pub ${SSHKEY_CONFIG_DIR}/ssh_host_dsa_key.pub 1>>/dev/null 2>>/dev/null
                retdsa2=$?
                [ $retrsa1 -eq 0 ] || /bin/cp -a ${SSHKEY_CONFIG_DIR}/ssh_host_rsa_key /etc/ssh/
                [ $retrsa2 -eq 0 ] || /bin/cp -a ${SSHKEY_CONFIG_DIR}/ssh_host_rsa_key.pub /etc/ssh/
                [ $retdsa1 -eq 0 ] || /bin/cp -a ${SSHKEY_CONFIG_DIR}/ssh_host_dsa_key /etc/ssh/
                [ $retdsa2 -eq 0 ] || /bin/cp -a ${SSHKEY_CONFIG_DIR}/ssh_host_dsa_key.pub /etc/ssh/
                if [ -d /etc/config/ssh ]; then
                        /bin/rm -rf /root/.ssh 1>>/dev/null 2>>/dev/null
                        /bin/ln -sf /etc/config/ssh /root/.ssh
                        [ -f /etc/config/ssh/id_rsa ] || /bin/ln -sf ssh_host_rsa_key /etc/config/ssh/id_rsa
                        [ -f /etc/config/ssh/id_rsa.pub ] || /bin/ln -sf ssh_host_rsa_key.pub /etc/config/ssh/id_rsa.pub
                fi
                if [ "x${BOOT_CONF}" = "xTS-NASX86" ] && [ ! -f /etc/config/ssh_key.fla ]; then
                        /bin/rm -f ${SSHKEY_CONFIG_DIR}/ssh_host_rsa_key*       2>>/dev/null
                        /bin/rm -f ${SSHKEY_CONFIG_DIR}/ssh_host_dsa_key* 2>>/dev/null
                        /usr/bin/ssh-keygen -t rsa -f ${SSHKEY_CONFIG_DIR}/ssh_host_rsa_key -N ""
                        /usr/bin/ssh-keygen -t dsa -f ${SSHKEY_CONFIG_DIR}/ssh_host_dsa_key -N ""
                        /bin/touch /etc/config/ssh_key.fla
                fi
        fi
}
update_sshd_config()
{
    /sbin/mksshdconf
        ENABLED_SFTP=`/sbin/getcfg LOGIN "SFTP Enable" -u -d TRUE`

        if [ "x${ENABLED_SFTP}" = "xTRUE" ]; then
                /bin/grep "/usr/libexec/sftp-server" ${SSHD_CONF} > /dev/null
                if [ $? != 0 ]; then
                        /bin/sed '$ a\Subsystem sftp \/usr\/libexec\/sftp-server' ${SSHD_CONF} > ${SSHD_CONF}.tmp
                        /bin/cp -f ${SSHD_CONF}.tmp ${SSHD_CONF}
                fi
        else
                /bin/grep "/usr/libexec/sftp-server" ${SSHD_CONF} > /dev/null
                if [ $? = 0 ]; then
                        /bin/sed "/\/usr\/libexec\/sftp-server/d" ${SSHD_CONF} > ${SSHD_CONF}.tmp
                        /bin/cp -f ${SSHD_CONF}.tmp ${SSHD_CONF}
                fi
        fi
        #Set PermitRootLogin yes
        OPTION="PermitRootLogin"
        if [ -z "`grep .*${OPTION}.* ${SSHD_CONF}`" ]; then
            echo "${OPTION} yes" > ${SSHD_CONF}
        else
        sed -i "s/^#\s\?${OPTION}\s\?[yesno]\{1,3\}.*/${OPTION} yes/g" ${SSHD_CONF}
        fi
       ##############################################################################################
       ###     AÑADIMOS AL FICHERO DE CONFIGURACION LOS PARAMETROS PARA PERMITIR TUNELES SSH.     ####
       ##############################################################################################
        # Añadimos AllowTcpForwarding →yes
        OPTION="AllowTcpForwarding"
        if [ -z "`grep .*${OPTION}.* ${SSHD_CONF}`" ]; then
            echo "${OPTION} yes" > ${SSHD_CONF}
        else
        sed -i "s/AllowTcpForwarding no/AllowTcpForwarding yes/g" ${SSHD_CONF}
        fi
        # Añadimos GatewayPorts →yes
        OPTION="GatewayPorts"
        if [ -z "`grep .*${OPTION}.* ${SSHD_CONF}`" ]; then
            echo "${OPTION} yes" > ${SSHD_CONF}
        else
        sed -i "s/GatewayPorts no/GatewayPorts yes/g" ${SSHD_CONF}
        fi
        # Añadimos PermitTunnel →yes
        OPTION="PermitTunnel"
        if [ -z "`grep .*${OPTION}.* ${SSHD_CONF}`" ]; then
            echo "${OPTION} yes" > ${SSHD_CONF}
```

```
        else
        sed -i "s/PermitTunnel no/PermitTunnel yes/g" ${SSHD_CONF}
        fi
        ##############################################################################################
}
sshd_privilege_separation()
{
        local sshd_home="/var/empty"
        local uid=110
        local grp=guest
        local passwd="/etc/config/passwd"
        local sshd_existed=`grep "^\[sshd\]:" ${passwd}`
        local gid=65534

        mkdir -p ${sshd_home}

        if [ -z "${sshd_existed}" ]; then
                /bin/sed -i "/^.*:.*:${uid}:.*:.*:.*:/d" ${passwd}
                if [ -f "/usr/local/bin/useradd" ] ;then
                        local useradd_cmd="/usr/local/bin/useradd -u ${uid} -g ${gid} -c \"SSHD Privilege Separation\" -d ${sshd_home} -M [sshd] -s /bin/sh"
                        eval ${useradd_cmd}
                else
                        local add_sshd_cmd="busybox adduser -D -u ${uid} -H -h \"${sshd_home}\" -g \"SSHD Privilege Separation\" -G ${grp} [sshd]"
                        eval ${add_sshd_cmd}
                fi
        fi

        sshd_exist_and_sid_110=`grep "^\[sshd\]:.*:${uid}:.*:.*:.*:" ${passwd}`
        if [ -z "${sshd_exist_and_sid_110}" ]; then
                /bin/sed -i "s/^\[sshd\]:.*/\[sshd\]:x:${uid}:${gid}:SSHD Privilege Separation:\/var\/empty:\/bin\/sh/" ${passwd}
        fi
}

case "$1" in
    start)
        # for openssh 7.5p1 and later
        sed -i '/^UsePrivilegeSeparation .*/d' ${SSHD_CONF_DEFAULT}

        if [ ! -f "${SSHD_CONF}" ]; then
                /bin/cp -f ${SSHD_CONF_DEFAULT} ${SSHD_CONF}
        fi
        if [ ! -f "${SSHD_CONF}" ]; then
                SSHD_CONF=${SSHD_CONF_DEFAULT}
        fi
        /bin/chmod 0400 /etc/config/shadow* /etc/default_config/shadow
        if [ `/sbin/getcfg LOGIN "SSH Enable" -u -d FALSE` != FALSE ]; then
                echo -n "Starting sshd service: "
                generte_ssh_key
                update_sshd_config
        sshd_privilege_separation
                /sbin/daemon_mgr sshd start "$SSH -f ${SSHD_CONF} -p $SSH_PORT"
                echo "OK"
                touch /var/lock/subsys/sshd
        fi

        if [ `/sbin/getcfg LOGIN "TELNET Enable" -u -d FALSE` != FALSE ]; then
                echo -n "Starting telnet service: "
                /sbin/daemon_mgr utelnetd start "$TELNET -p $TELNET_PORT &"
                echo "OK"
                touch /var/lock/subsys/utelnetd
        fi

        if [ `/sbin/getcfg "TFTP Server" "Enable" -u -d FALSE` != FALSE ]; then
                /etc/init.d/opentftp.sh start 1>/dev/null 2>&1
        fi

        ;;
    stop)
        echo -n "Shutting down sshd service: "
        /sbin/daemon_mgr sshd stop $SSH
        /usr/bin/killall sshd 2>/dev/null
        rm -f /var/lock/subsys/sshd
        echo "OK"

        echo -n "Shutting down telnet service: "
```

```
        /sbin/daemon_mgr utelnetd stop $TELNET
        rm -f /var/lock/subsys/utelnetd
        echo "OK"
        ;;

    restart)
        $0 stop
        $0 start
        ;;
    *)
        echo "Usage: /etc/init.d/login.sh {start|stop|restart}"
        exit 1
esac

exit 0
```

5. Fichero de configuración ttn-lw-stack-docker.yml

```
# Identity Server configuration
# Email configuration for "thethings.example.com"
is:
  email:
    sender-name: 'The Things Stack'
    sender-address: 'ttn@redesipv6.es'
    network:
      name: 'The Things Stack'
      console-url: 'https://ttn.redesipv6.es/console'
      identity-server-url: 'https://ttn.redesipv6.es/oauth'
  oauth:
    ui:
      canonical-url: 'https://ttn.redesipv6.es/oauth'
      #canonical-url: 'http://ttn.redesipv6.es:8885/oauth'
      is:
        base-url: 'https://ttn.redesipv6.es/api/v3'
        #base-url: 'http://ttn.redesipv6.es:8885/api/v3'
# HTTP server configuration
http:
  #listen: '1885'.    <- DESACTIVAMOS HTTP
  listen-tls: '8885'
  cookie:
    block-key: ''                # generate 32 bytes (openssl rand -hex 32)
    hash-key: ''                 # generate 64 bytes (openssl rand -hex 64)
  metrics:
    password: 'Jnds&5dsjsdb&gsah'               # ← GENERAMOS UNA CONTRASEÑA ROBUSTA
  pprof:
    password: 'Kjsd7dsad&dssJ3j'                 # ← GENERAMOS UNA CONTRASEÑA ROBUSTA
# If using custom certificates:
tls:
  source: file
  certificate: /config/cert.pem
  key: /config/key.pem
gs:
  mqtt:
    public-address: 'ttn.redesipv6.es:1882'
    public-tls-address: 'ttn.redesipv6.es:8882'
  mqtt-v2:
    public-address: 'ttn.redesipv6.es:1881'
    public-tls-address: 'ttn.redesipv6.es:8881'
```

```
gcs:
  basic-station:
    default:
      lns-uri: 'wss://ttn.redesipv6.es:8887'
  the-things-gateway:
    default:
      mqtt-server: 'mqtts://ttn.redesipv6.es:8881'
console:
  ui:
    canonical-url: 'https://ttn.redesipv6.es/console'
    is:
      base-url: 'https://ttn.redesipv6.es/api/v3'
    gs:
      base-url: 'https://ttn.redesipv6.es/api/v3'
    ns:
      base-url: 'https://ttn.redesipv6.es/api/v3'
    as:
      base-url: 'https://ttn.redesipv6.es/api/v3'
    js:
      base-url: 'https://ttn.redesipv6.es/api/v3'
    qrg:
      base-url: 'https://ttn.redesipv6.es/api/v3'
    edtc:
      base-url: 'https://ttn.redesipv6.es/api/v3'
  oauth:
    authorize-url: 'https://ttn.redesipv6.es/oauth/authorize'
    token-url: 'https://ttn.redesipv6.es/oauth/token'
    logout-url: 'https://ttn.redesipv6.es/oauth/logout'
    client-id: 'console'
    client-secret: 'nKHy77$ghas65Khas/'          # choose or generate a secret
as:
  mqtt:
    public-address: 'https://ttn.redesipv6.es:1883'
    public-tls-address: 'https://ttn.redesipv6.es:8883'
  webhooks:
    downlink:
      public-address: 'ttn.redesipv6.es:1885/api/v3'
dcs:
  oauth:
    authorize-url: 'https://ttn.redesipv6.es/oauth/authorize'
    token-url: 'https://ttn.redesipv6.es/oauth/token'
    logout-url: 'https://ttn.redesipv6.es/oauth/logout'
    client-id: 'device-claiming'
    client-secret: 'device-claiming'          # ← GENERAMOS UN SECRETO ROBUSTO
  ui:
    canonical-url: 'https://ttn.redesipv6.es/claim'
    as:
      base-url: 'https://ttn.redesipv6.es/api/v3'
    dcs:
      base-url: 'https://ttn.redesipv6.es/api/v3'
    is:
      base-url: 'https://ttn.redesipv6.es/api/v3'
    ns:
      base-url: 'https://ttn.redesipv6.es/api/v3'
```

6. Fichero de configuración settings.js

```
module.exports = {
    uiPort: process.env.PORT || 1880,
    mqttReconnectTime: 15000,
    serialReconnectTime: 15000,
    debugMaxLength: 1000,

    // The following property can be used to enable HTTPS
    // See http://nodejs.org/api/https.html#https_https_createserver_options_requestlistener
    // for details on its contents.
    // This property can be either an object, containing both a (private) key and a (public) certificate,
    // or a function that returns such an object:
    //// https object:
    https: {
      key: require("fs").readFileSync('/data/key.pem'),
      cert: require("fs").readFileSync('/data/cert.pem')
    },
     // The following property can be used to cause insecure HTTP connections to
    // be redirected to HTTPS.
    requireHttps: true,

    functionGlobalContext: {
        // os:require('os'),
        // jfive:require("johnny-five"),
        // j5board:require("johnny-five").Board({repl:false})
    },
    // By default, the property is set to false to avoid accidental exposure of
    // their values. Setting this to true will cause the keys to be listed.
    exportGlobalContextKeys: false,
    // Configure the logging output
    logging: {
        // Only console logging is currently supported
        console: {
          level: "info",
            metrics: false,
            audit: false
        }
    },
    editorTheme: {
        projects: {
            // To enable the Projects feature, set this value to true
            enabled: false
        }
    }
}
```

7. Fichero de configuración Dockerfile para phpMyAdmin

```
FROM phpmyadmin
RUN a2enmod ssl
RUN sed -ri -e 's,80,443,' /etc/apache2/sites-available/000-default.conf
RUN sed -i -e '/^<\/VirtualHost>/i SSLEngine on' /etc/apache2/sites-available/000-default.conf
RUN sed -i -e '/^<\/VirtualHost>/i SSLCertificateFile /cert/cert.pem' /etc/apache2/sites-available/000-default.conf
RUN sed -i -e '/^<\/VirtualHost>/i SSLCertificateKeyFile /cert/privkey.pem' /etc/apache2/sites-available/000-default.conf
RUN sed -i -e '/^<\/VirtualHost>/i SSLCertificateChainFile /cert/fullchain.pem' /etc/apache2/sites-available/000-
default.conf
EXPOSE 5443
```

8. Fichero de configuración grafana.ini

```
#################################### Server ####################################
[server]
# Protocol (http, https, h2, socket)
protocol = https

# https certs & key file
cert_file = /etc/grafana/cert.pem
cert_key = /etc/grafana/privkey.pem
```

9. Fichero de configuración docker-compose.yml del contenedor OpenSSH

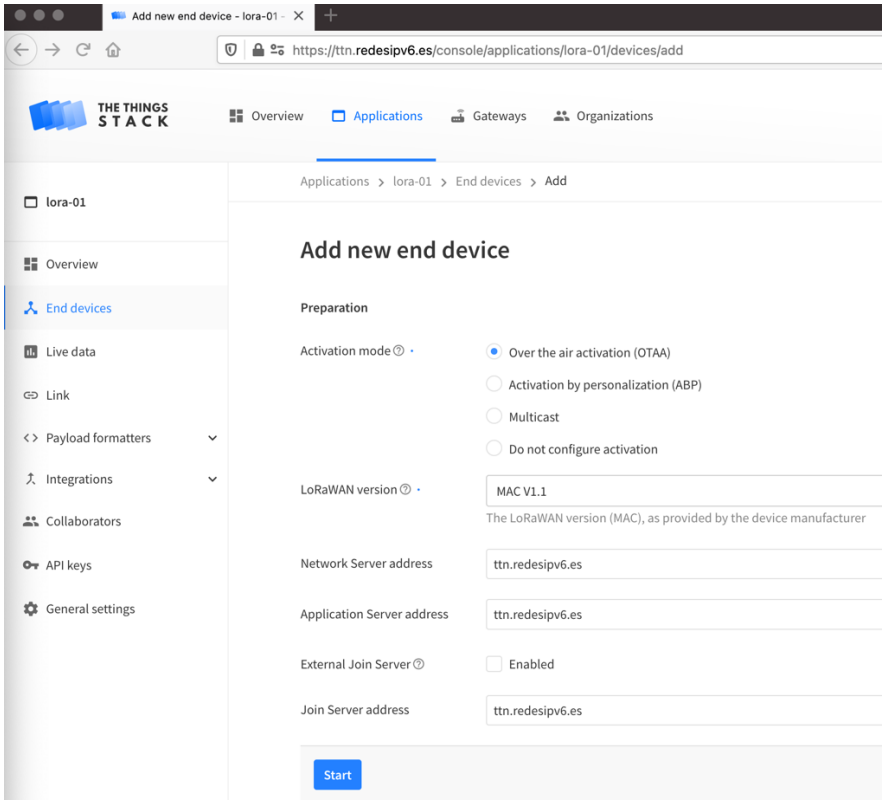
```
version: '3'
services:
  openssh-server:
    image: ghcr.io/linuxserver/openssh-server
    container_name: openssh-server
    environment:
      - PUID=1000
      - PGID=1000
      - TZ=Europe/Madrid
      - PUBLIC_KEY_FILE=./ect/ssh_host_keys/ssh_host_rsa_key.pub
      - USER_NAME=admin
    volumes:
      - ./etc:/config
    ports:
      - 2222:2222 # Puerto de escucha túnel SSH
      - 19999:19999 # Puerto túnel pasarela LoraWAN 1
      - 19998:19998 # Puerto túnel pasarela LoraWAN 2
      - 19997:19997 # Puerto túnel pasarela LoraWAN 3
      - 19996:19996 # Puerto túnel pasarela LoraWAN 4

    restart: unless-stopped
```

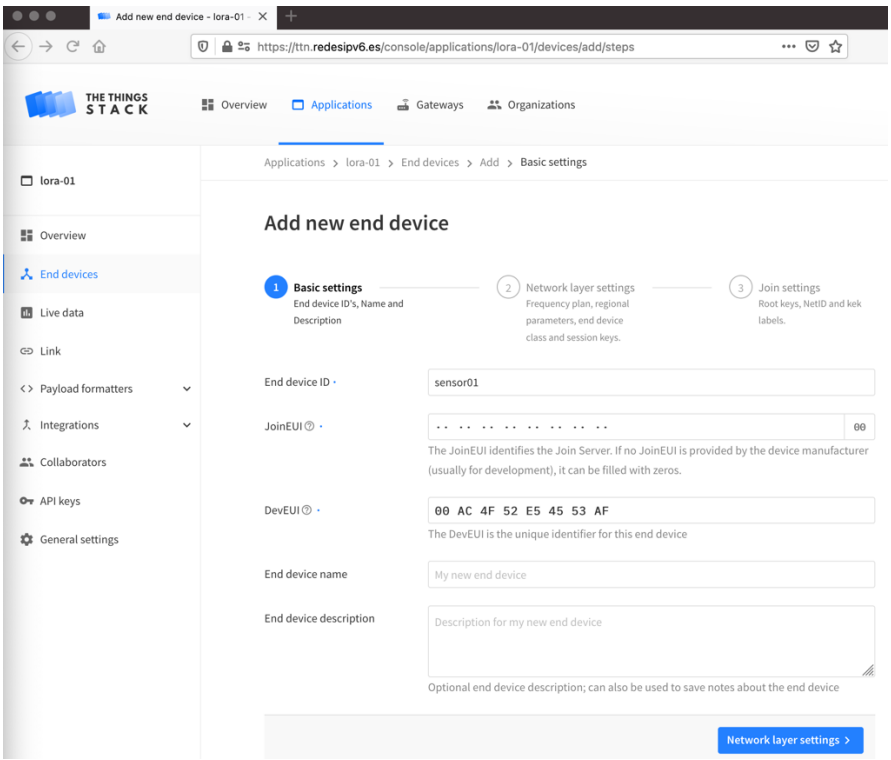
10. Configuración de seguridad NGFW mediante CLI

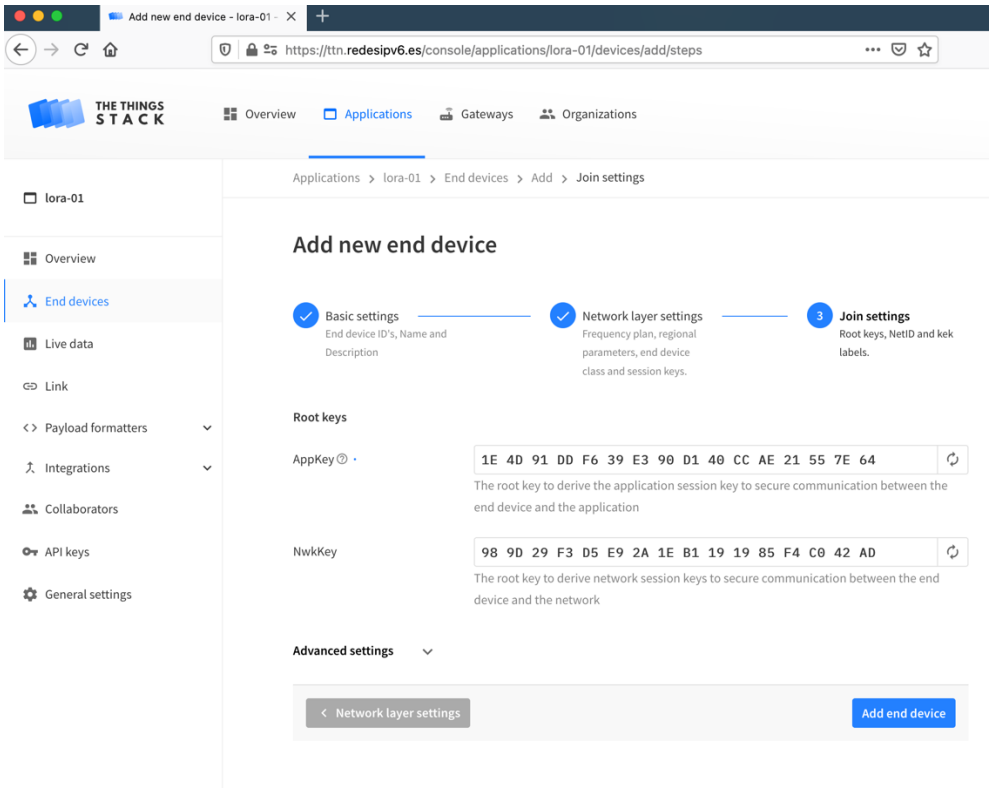
```
config system global
          set admin-https-ssl-version tlsv1-2
          set admin-ssh-v1 disable
          set admin-maintainer disable
          set pre-login-banner enable
          set allow-traffic-redirect disable
          set anti-replay strict
          set arp-max-entry 131072
          set auth-keepalive disable
end
```

Anexo C: Capturas de pantalla

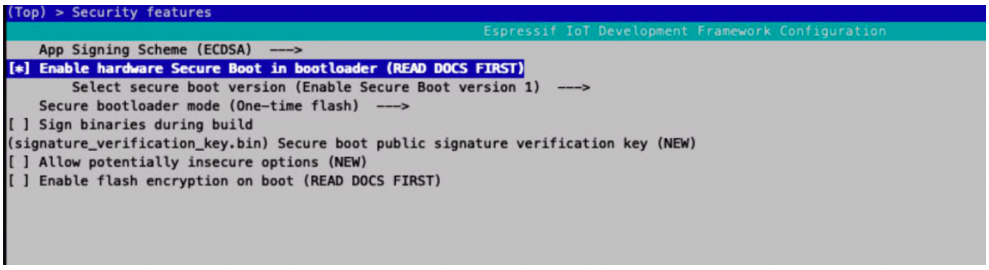
1. Configuración seguridad LoRaWAN



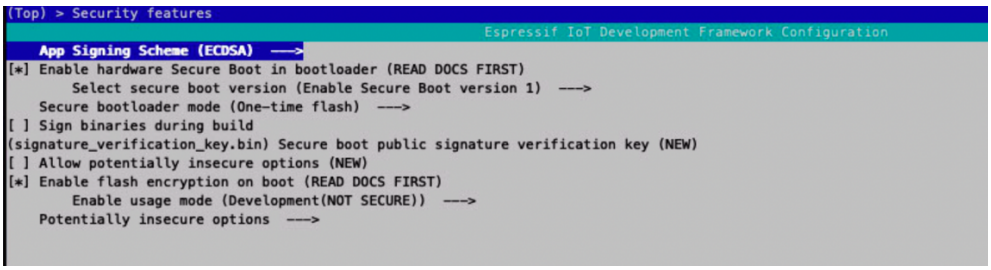




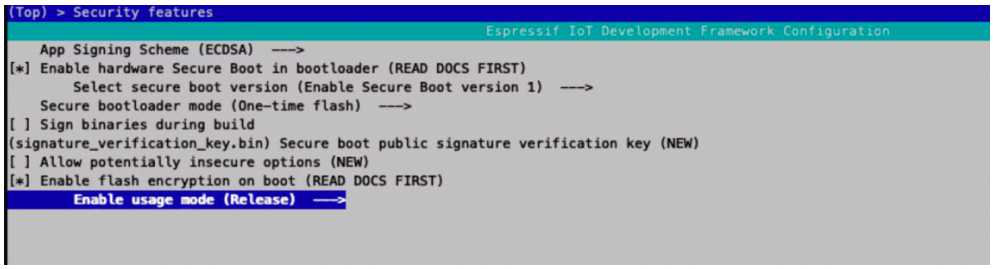
2. Configuración de Secure Boot en procesador ESP32



3. Encriptación de memoria Flash



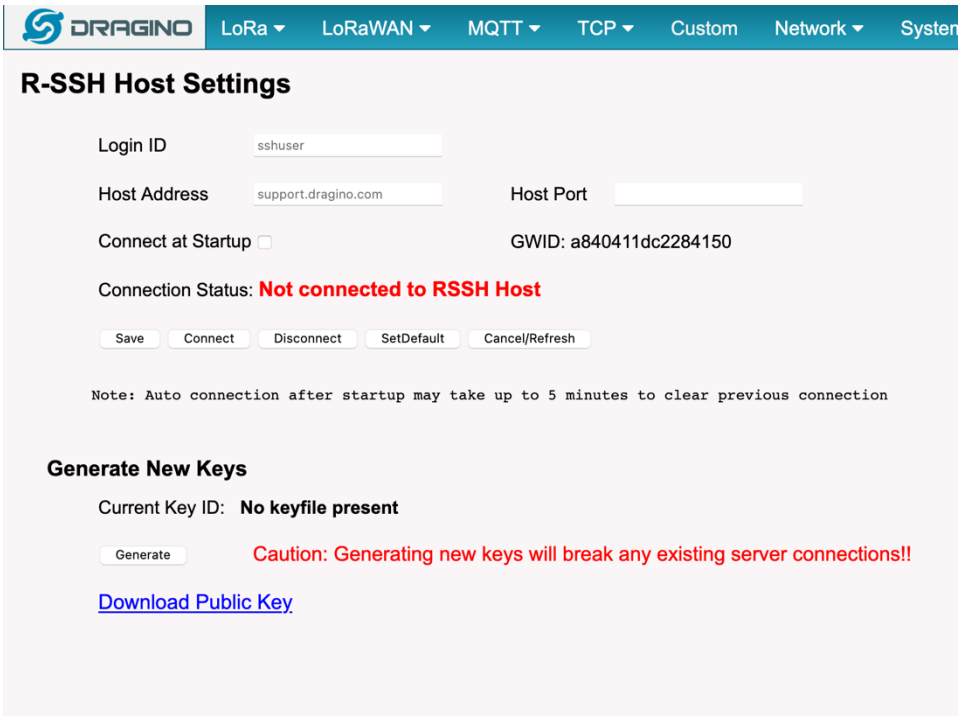
4. Cambio de modo de uso en la encriptación de la memoria Flash


(Top) > Security features
Espressif IoT Development Framework Configuration
App Signing Scheme (ECDSA) --->
[*] Enable hardware Secure Boot in bootloader (READ DOCS FIRST)
Select secure boot version (Enable Secure Boot version 1) --->
Secure bootloader mode (One-time flash) --->
[ ] Sign binaries during build
(signature_verification_key.bin) Secure boot public signature verification key (NEW)
[ ] Allow potentially insecure options (NEW)
[*] Enable flash encryption on boot (READ DOCS FIRST)
Enable usage mode (Release) -->

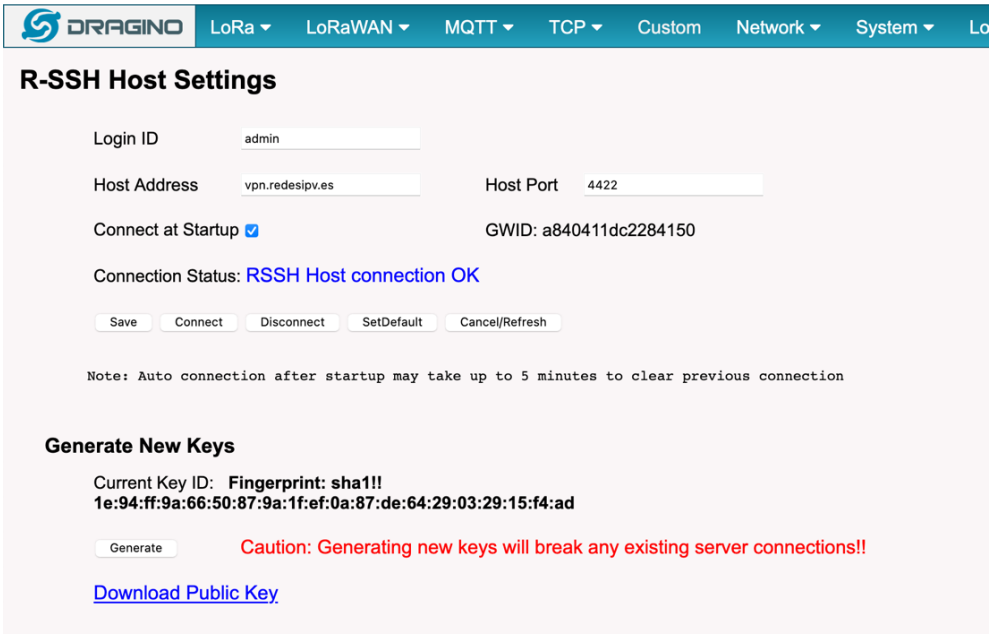
5. Procesos ejecutándose en la pasarela LG-308

```
root@lora:/# ps -w
  PID USER       VSZ STAT COMMAND
    1 root      1524 S    /sbin/procd
    2 root         0 SW   [kthreadd]
    3 root         0 SW   [ksoftirqd/0]
    5 root         0 SW<  [kworker/0:0H]
    6 root         0 SW   [kworker/u2:0]
    7 root         0 SW<  [lru-add-drain]
   47 root         0 SW   [oom_reaper]
   77 root         0 SW<  [writeback]
   79 root         0 SW<  [crypto]
   80 root         0 SW<  [bioset]
   82 root         0 SW<  [kblockd]
   96 root         0 SW<  [watchdogd]
  109 root         0 SW   [kworker/0:1]
  116 root         0 SW   [kswapd0]
  175 root         0 SW   [spi0]
  193 root         0 SW<  [bioset]
  198 root         0 SW<  [bioset]
  203 root         0 SW<  [bioset]
  208 root         0 SW<  [bioset]
  213 root         0 SW<  [bioset]
  218 root         0 SW<  [bioset]
  223 root         0 SW<  [bioset]
  301 root         0 SW<  [ipv6_addrconf]
  309 root         0 SW<  [kworker/0:1H]
  352 root         0 SW   [kworker/0:2]
  403 root         0 SWN  [jffs2_gcd_mtd4]
  463 root      1172 S    /sbin/ubusd
  471 root       896 S    /sbin/askfirst /usr/libexec/login.sh
  568 root         0 SW<  [bioset]
  571 root         0 SW<  [bioset]
  575 root         0 SW<  [bioset]
  579 root         0 SW<  [bioset]
  582 root         0 SW<  [bioset]
  585 root         0 SW<  [bioset]
  588 root         0 SW<  [bioset]
  591 root         0 SW<  [bioset]
  741 root         0 SW<  [cfg80211]
  957 root      1212 S    /sbin/logd -S 64
  983 root      1496 S    /sbin/rpcd
 1125 root      1688 S    /sbin/netifd
 1546 root      1196 S    udhcpc -p /var/run/udhcpc-eth1.pid -s /lib/netifd/dhcp.script -f -t 0 -i eth1 -x hostname:dragino-1dc228
 1567 root         0 SW   [spi1]
 1672 root      1200 S    /usr/sbin/crond -f -c /etc/crontabs -l 5
 1699 root      1056 S    /usr/sbin/dropbear -F -P /var/run/dropbear.1.pid -p 22 -K 300 -T 3
 1910 dnsmasq   1300 S    /usr/sbin/dnsmasq -C /var/etc/dnsmasq.conf.setup -k -x /var/run/dnsmasq/dnsmasq.setup.pid
 2004 root      1460 S    /usr/sbin/dbus-daemon --system
 2009 root       920 S    /usr/sbin/thd --socket /tmp/triggerhappy.socket --triggers /etc/triggerhappy/triggers.d/ --daemon /dev/in
 2073 root      1200 S<   /usr/sbin/ntpd -n -N -S /usr/sbin/ntpd-hotplug -p 0.openwrt.pool.ntp.org -p 1.openwrt.pool.ntp.org -p 2.o
 2122 root      1224 S    /bin/sh /usr/bin/iot_keep_alive.sh
 2145 root      1364 S    /bin/ash /etc/rc.common /etc/rc.d/S99get-page-data.sh boot
 2327 root      1124 S    /usr/sbin/dropbear -F -P /var/run/dropbear.1.pid -p 22 -K 300 -T 3
 2497 root      1200 S    -ash
 3570 root      4008 S    /usr/sbin/uhttpd -f -h /www -r lora.redesipv6.es -c /etc/config/http.conf -x /cgi-bin -u /ubus -t 120 -T
 6245 root         0 SW   [kworker/u2:1]
 6589 root      1196 S    sleep 15
 6798 root      1196 S    sleep 9
 6799 root      1196 R    ps -w
12501 root      3184 S    /usr/bin/lora_pkt_fwd
22751 root         0 SW   [kworker/u2:2]
root@lora:/#
```

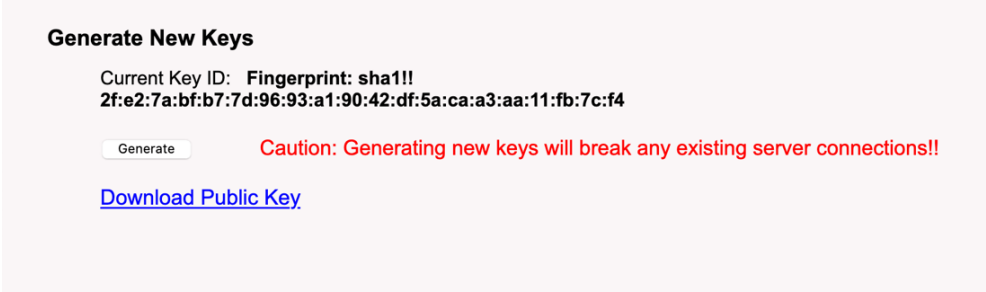
6. Pagina de Remote-Mgmt



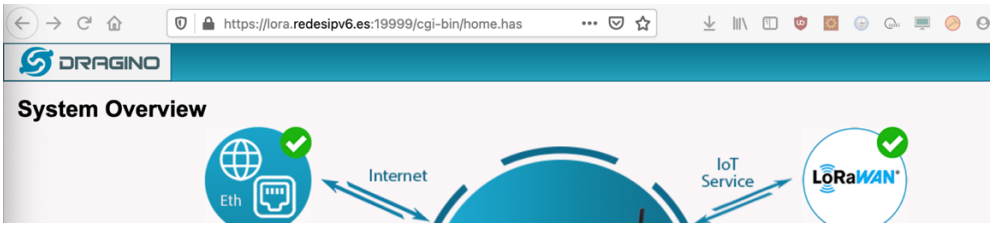
7. Parámetros de conexión



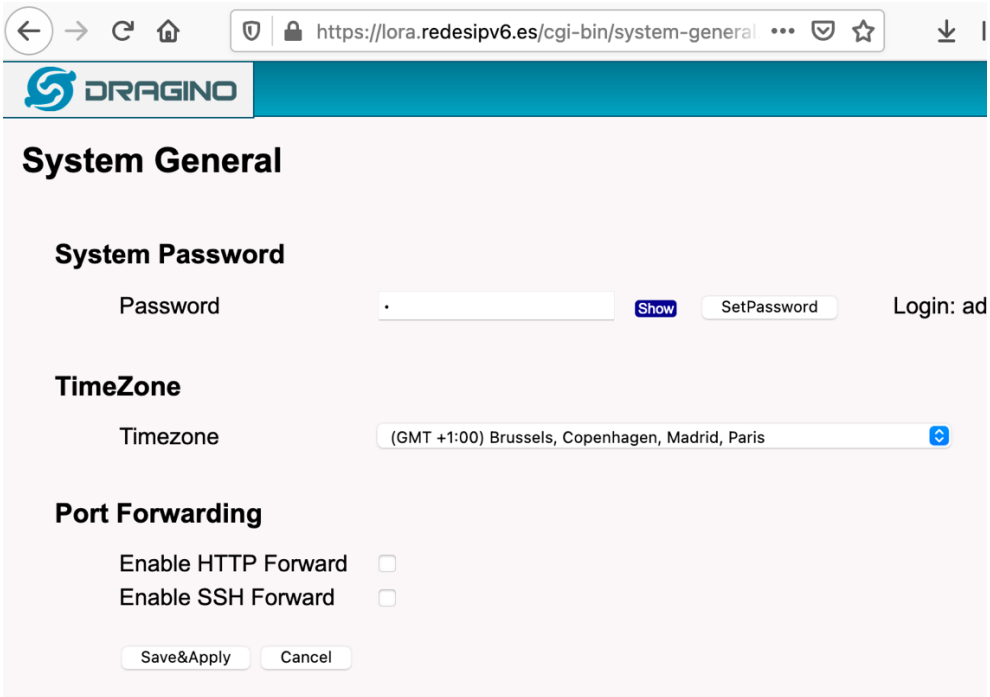
8. Generación de claves publica/privada SSH



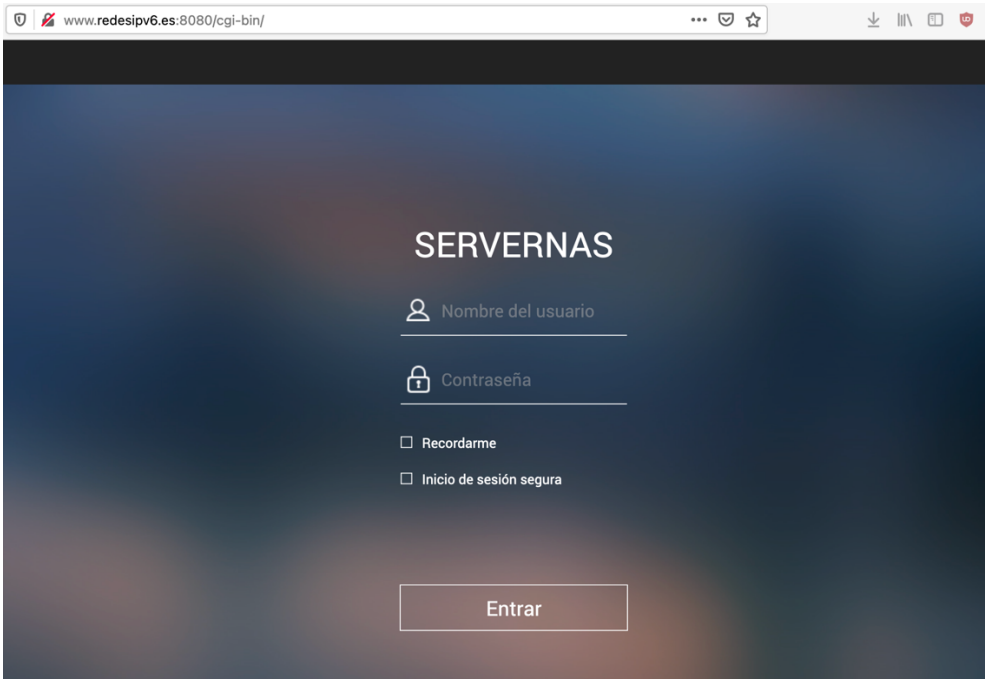
9. Conexión al LG-308 a través de túnel SSH



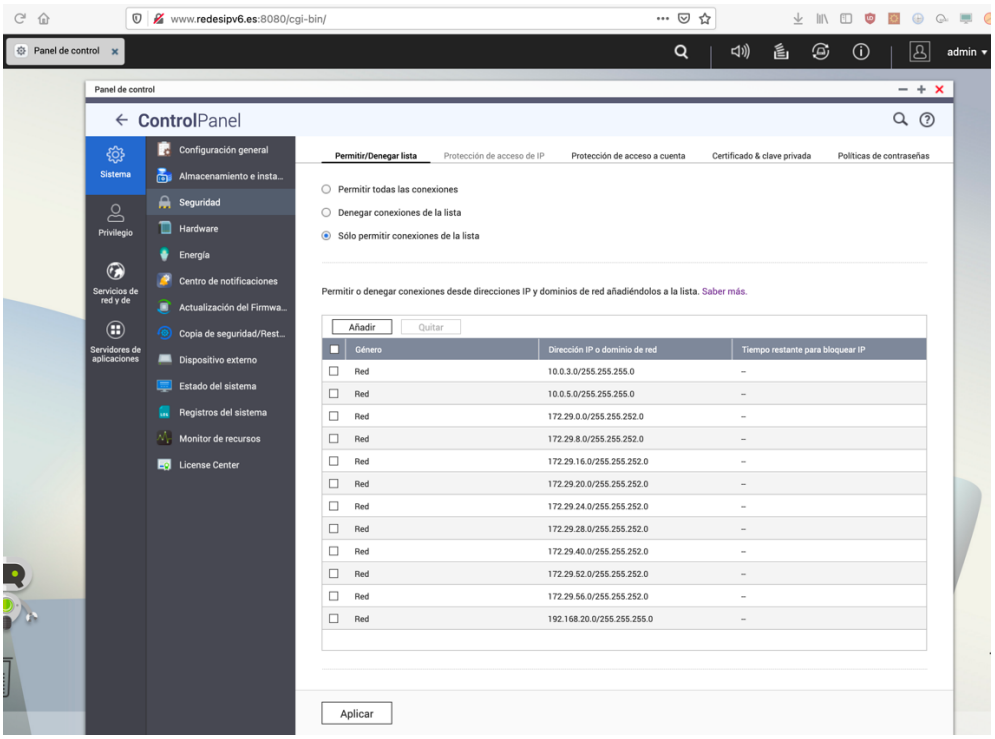
10. Desactivación de servicios HTTP y SSH



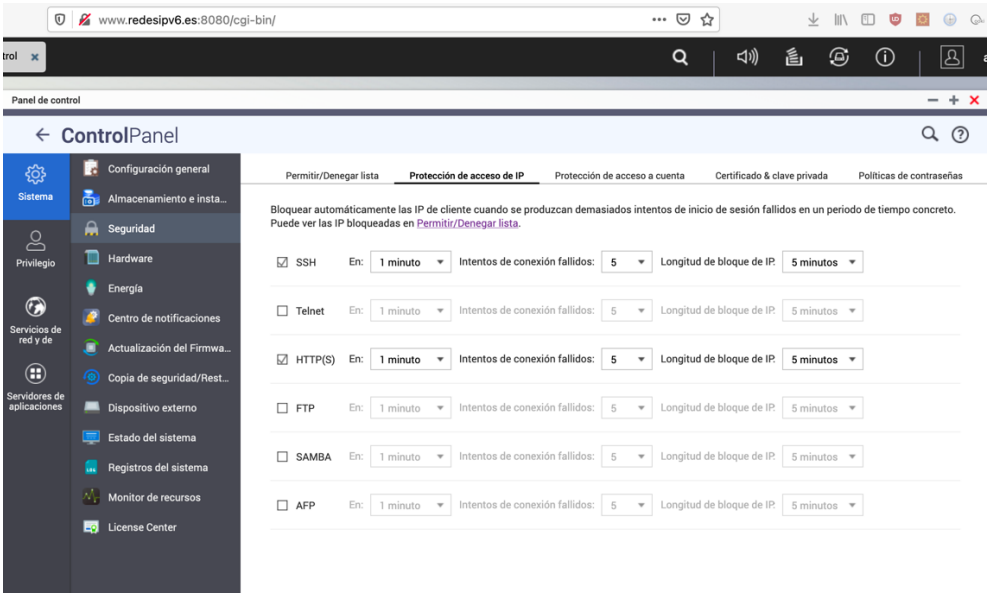
11. Pagina de administración de QNAP



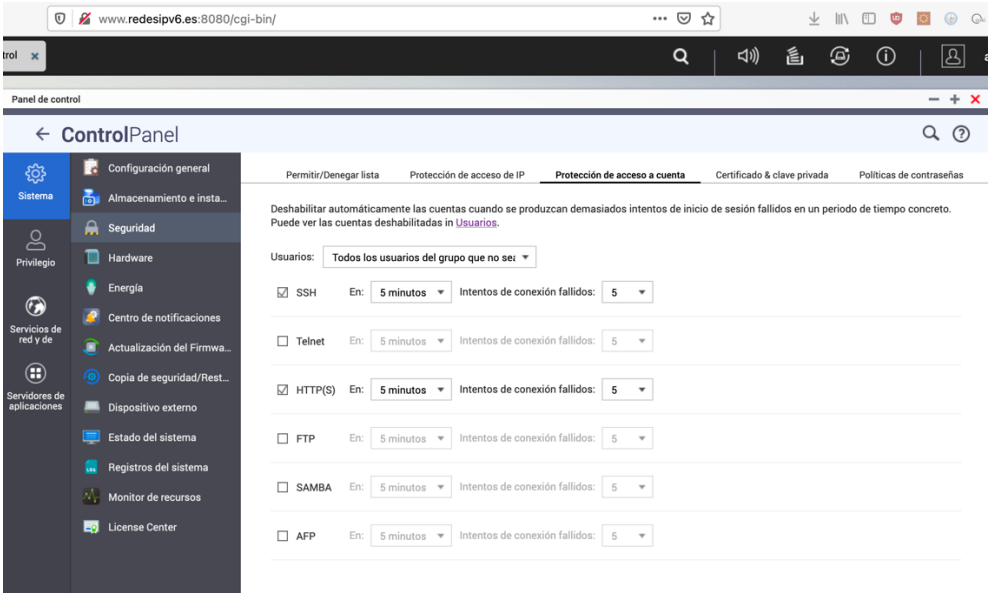
12. Panel de Control → Seguridad → Permitir/Denegar lista



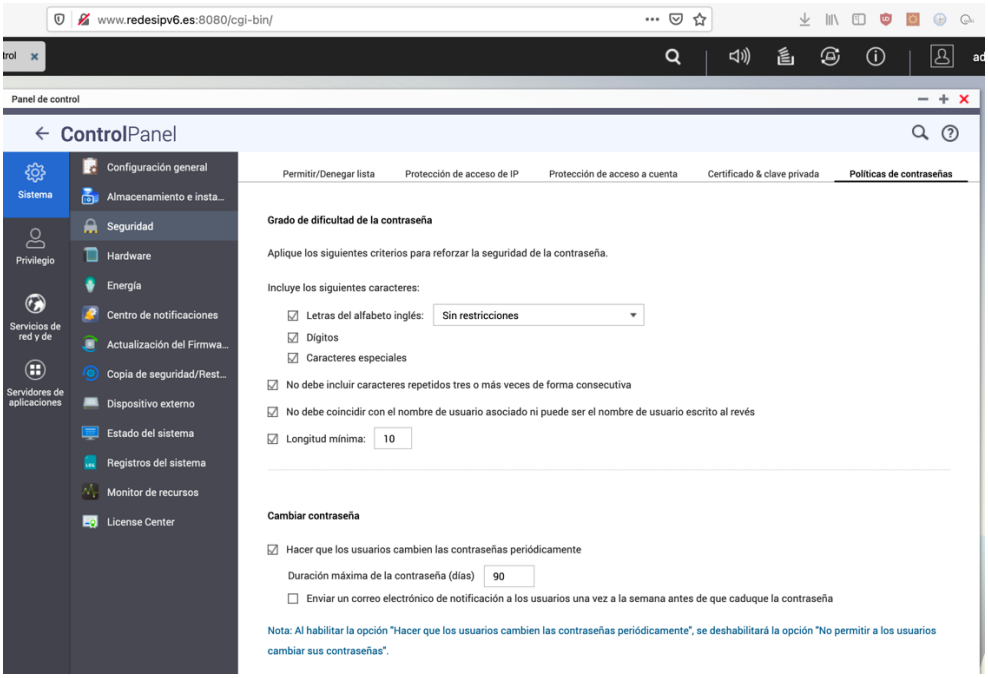
13. Panel de Control → Seguridad → Protección de acceso IP



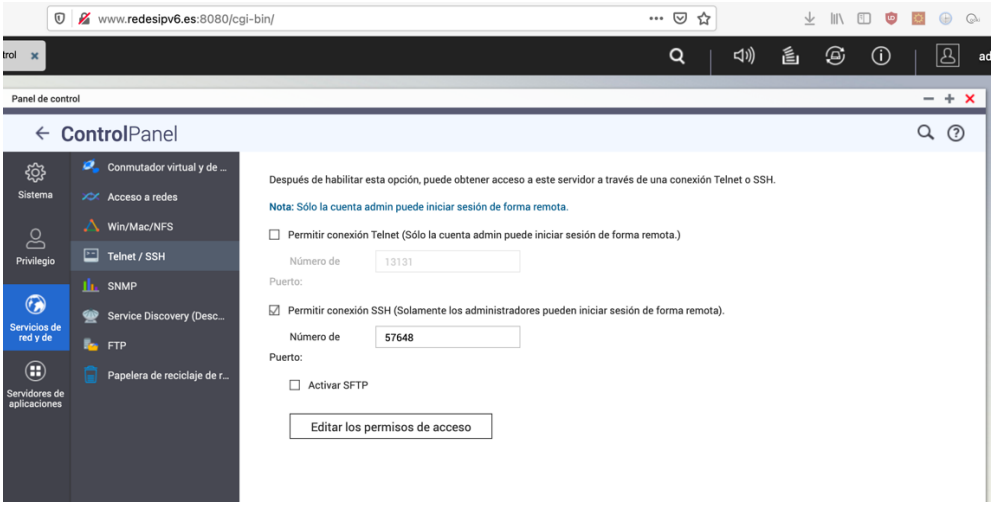
14. Panel de Control → Seguridad → Protección de acceso IP



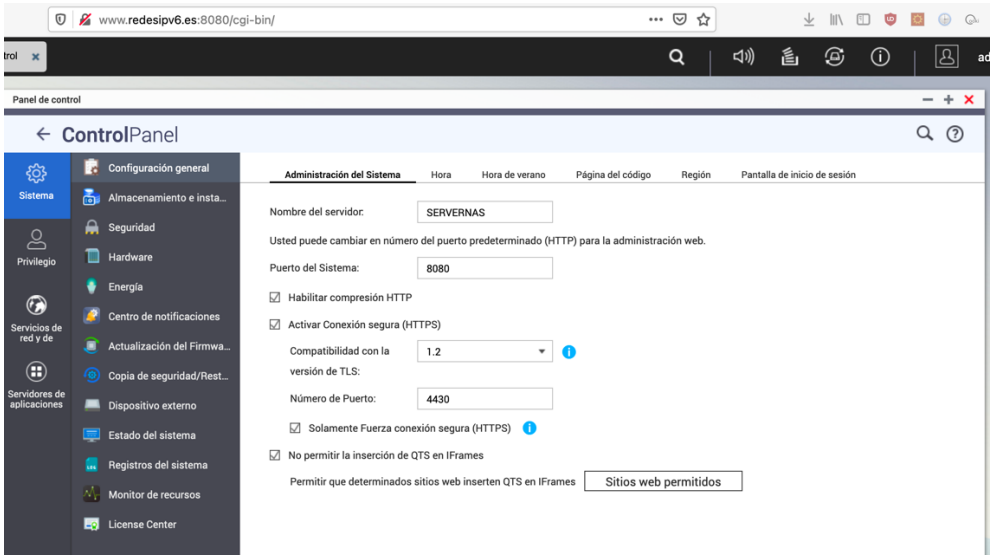
15. Panel de Control → Seguridad → Políticas de contraseñas



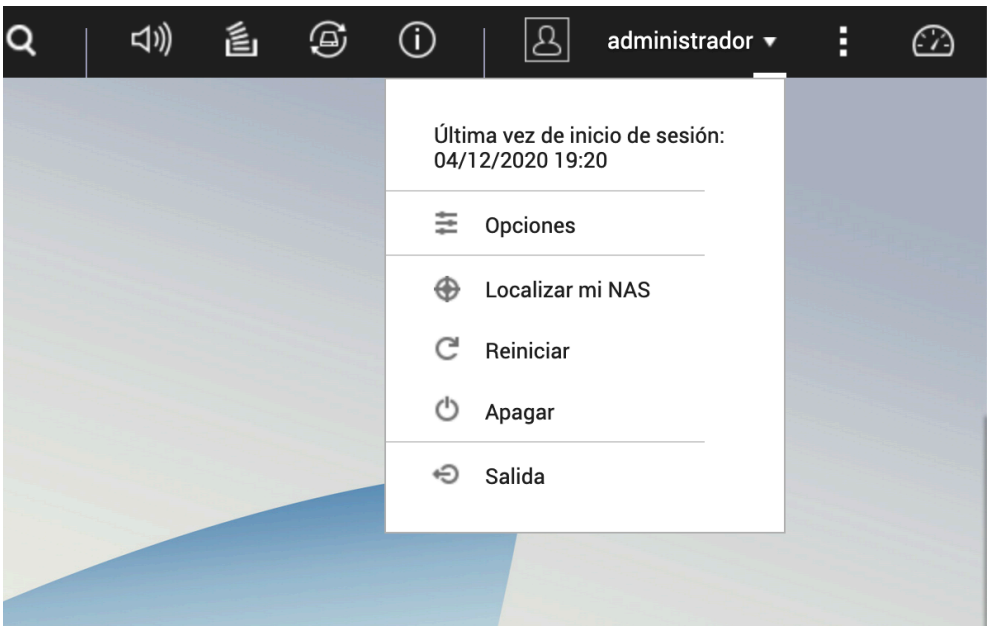
16. Panel de Control → Servicios de red → Telnet/SSH



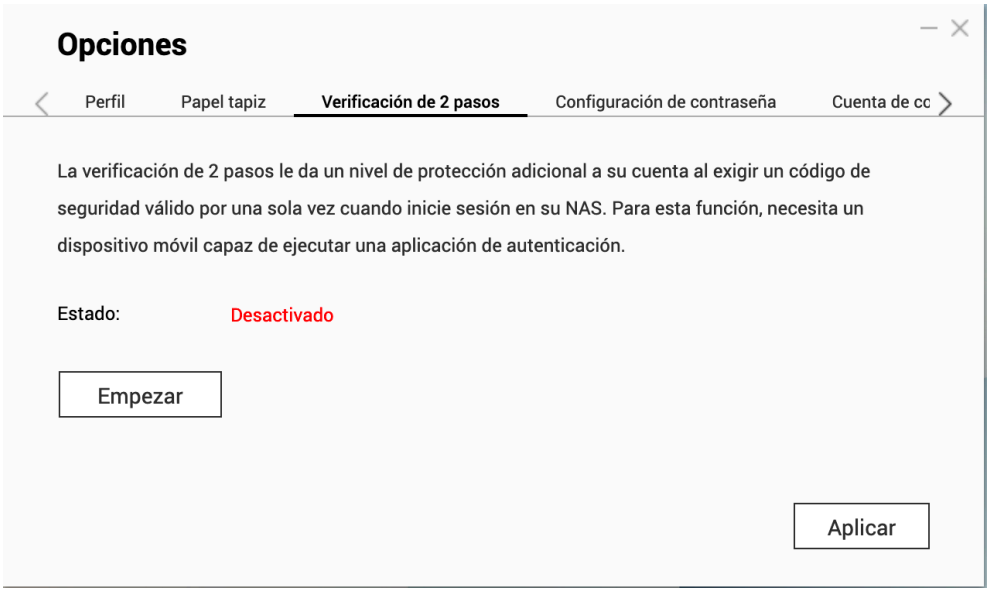
17. Panel de Control → Configuración general → Administración del Sistema



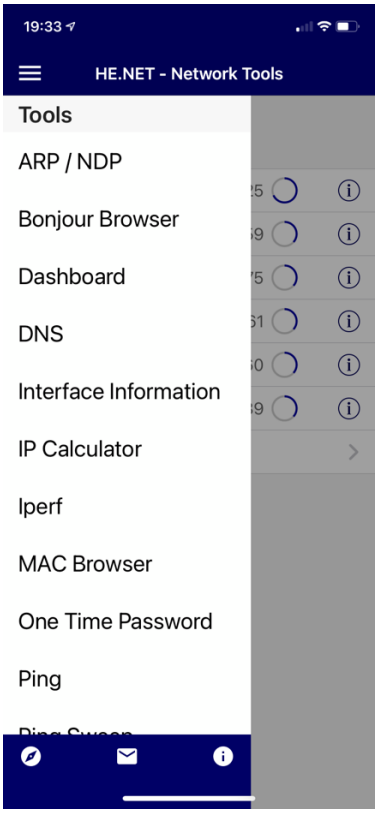
18. Opciones de usuario



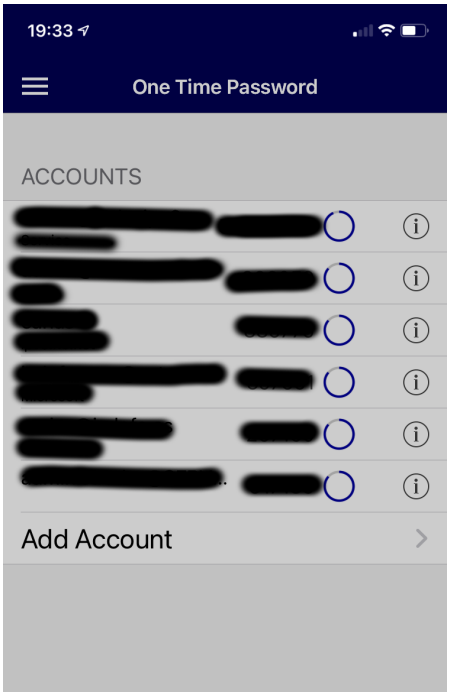
19. Verificación de 2 pasos → Inicio



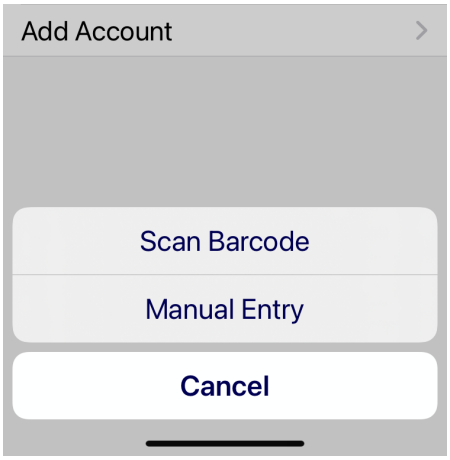
20. Aplicación móvil OTP



21. Añadir nueva cuenta OTP

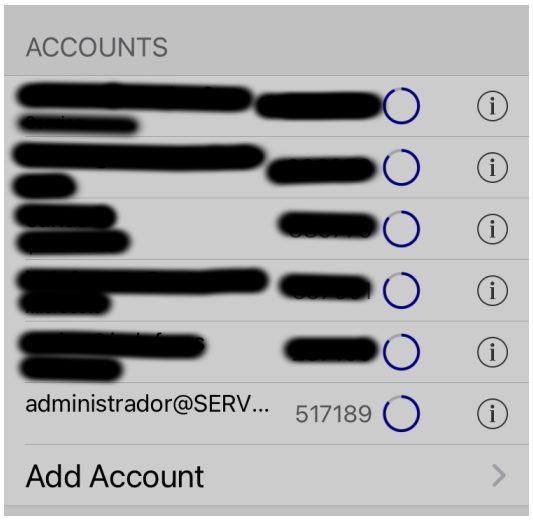


22. Escanear código QR

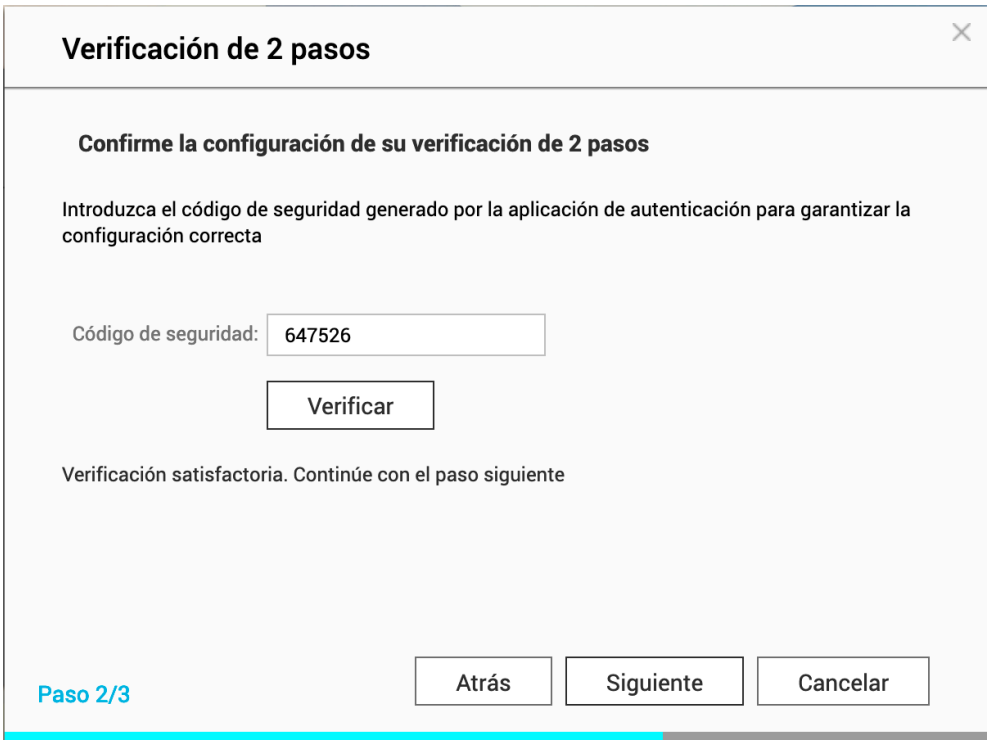


23. Verificación de 2 pasos → Código QR

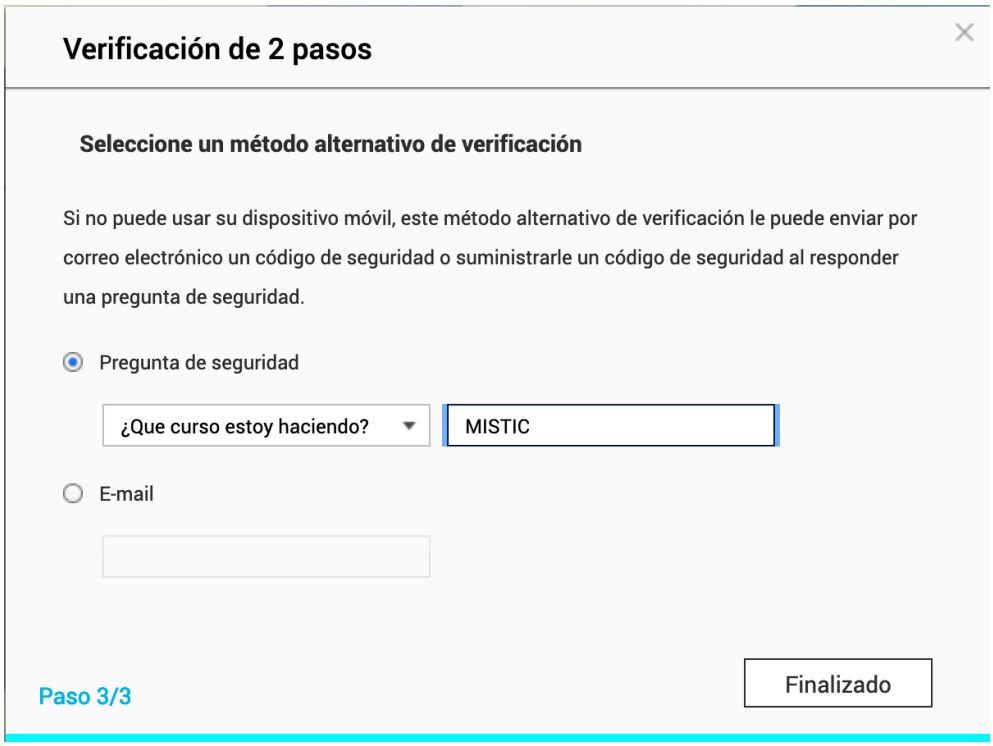
24. Cuenta añadida a la aplicación OTP



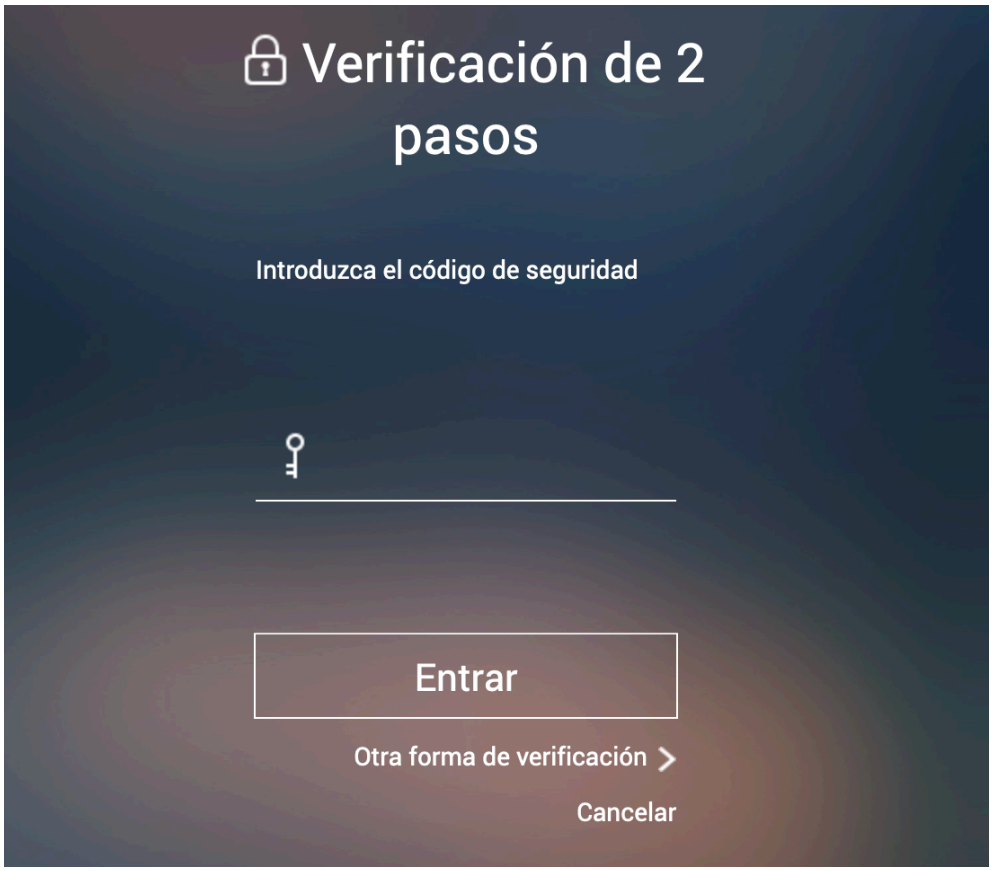
25. Verificación de 2 pasos → Verificación



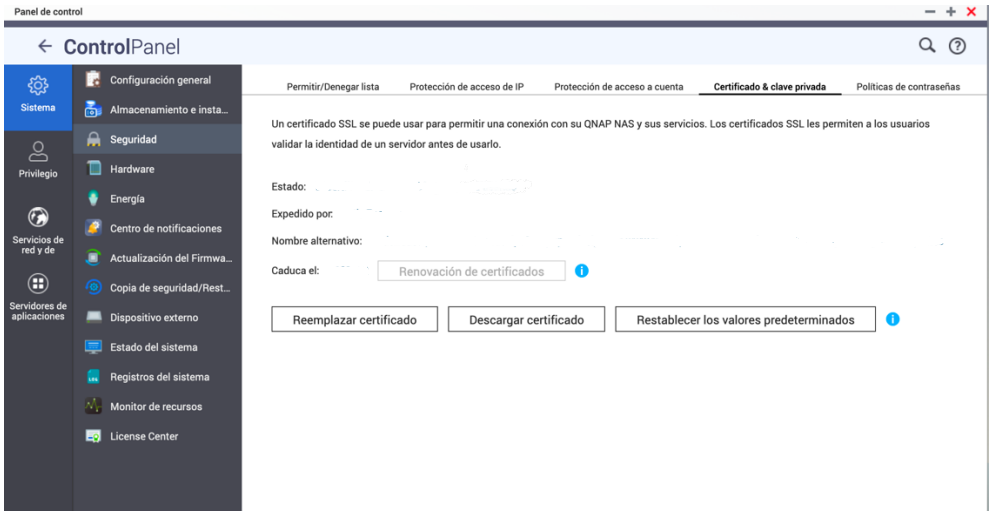
26. Verificación de 2 pasos → Método alternativo



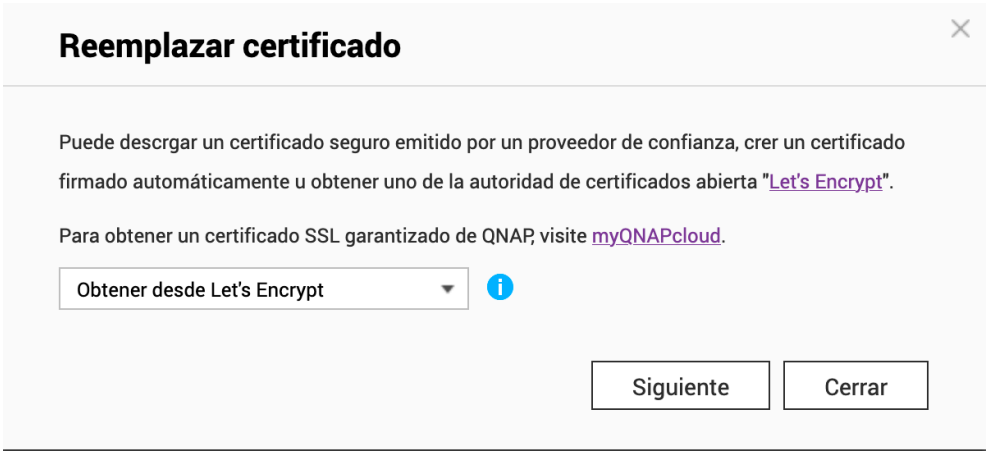
27. Solicitud de doble factor de autentificación



28. Panel de Control → Seguridad → Certificado & clave privada

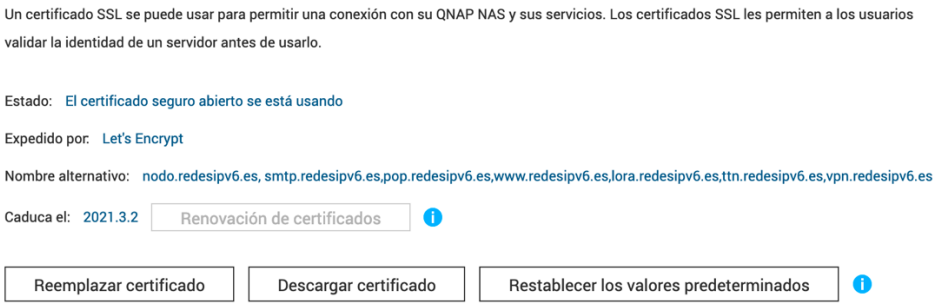


29. Reemplazar certificado

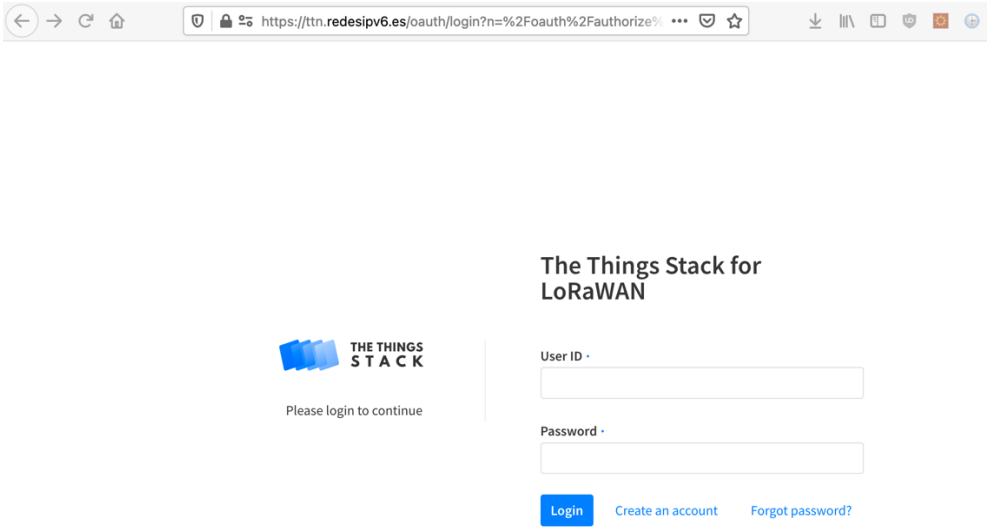


30. Obtener desde Let’s Encrypt

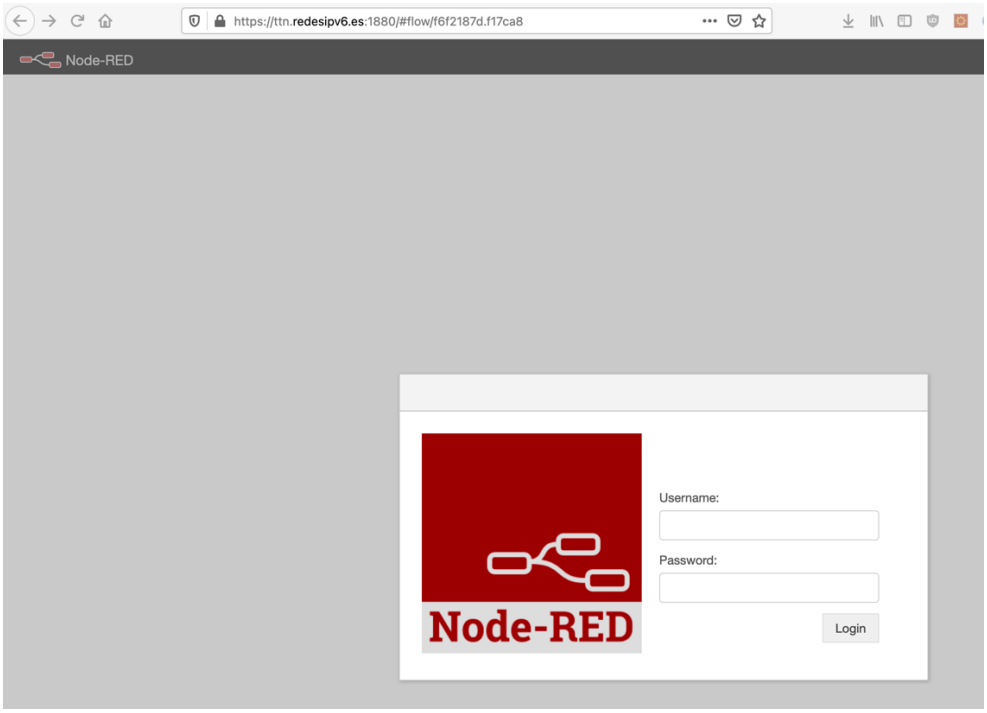
31. Certificado generado



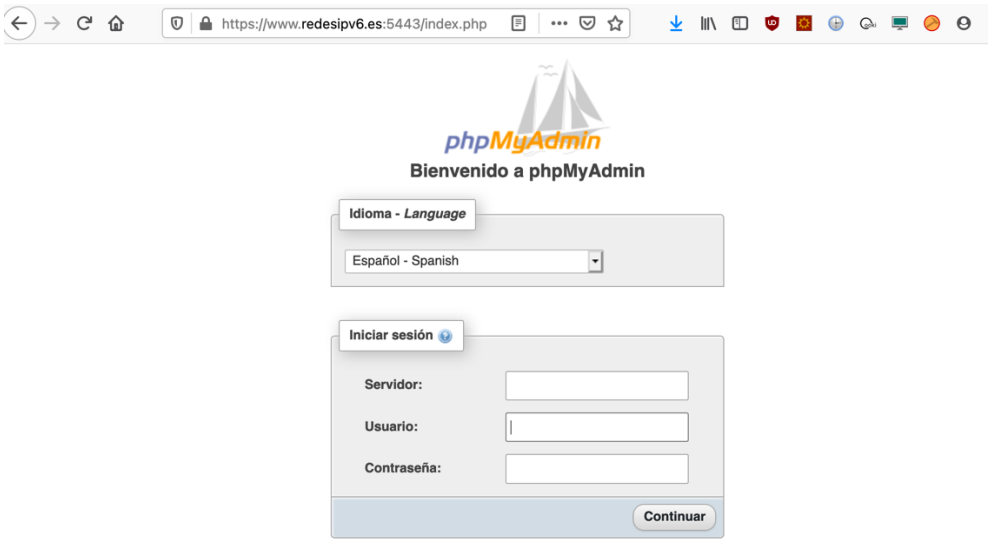
32. Acceso seguro a TTN Stack



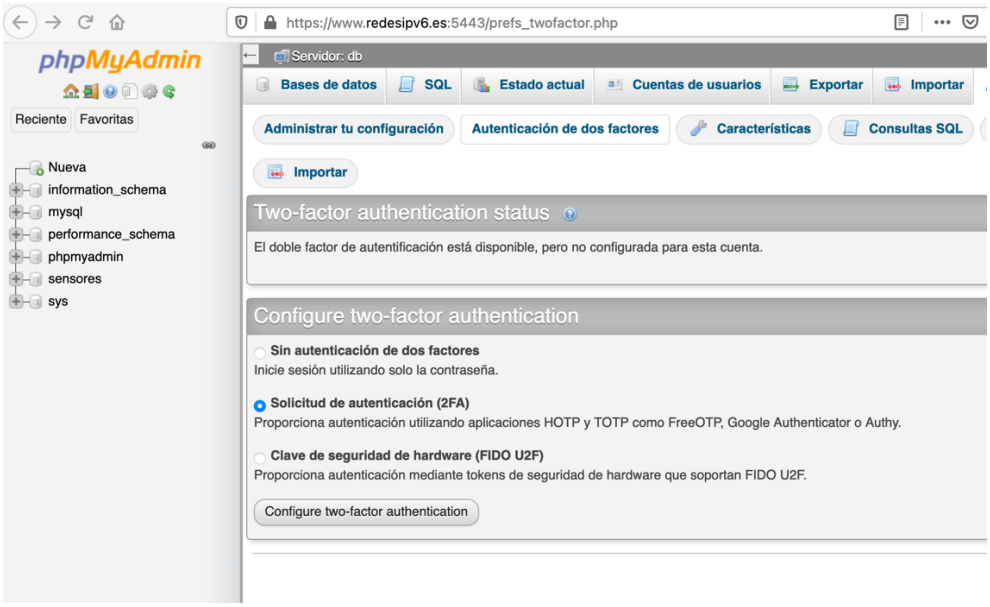
33. Acceso seguro a Node-Red



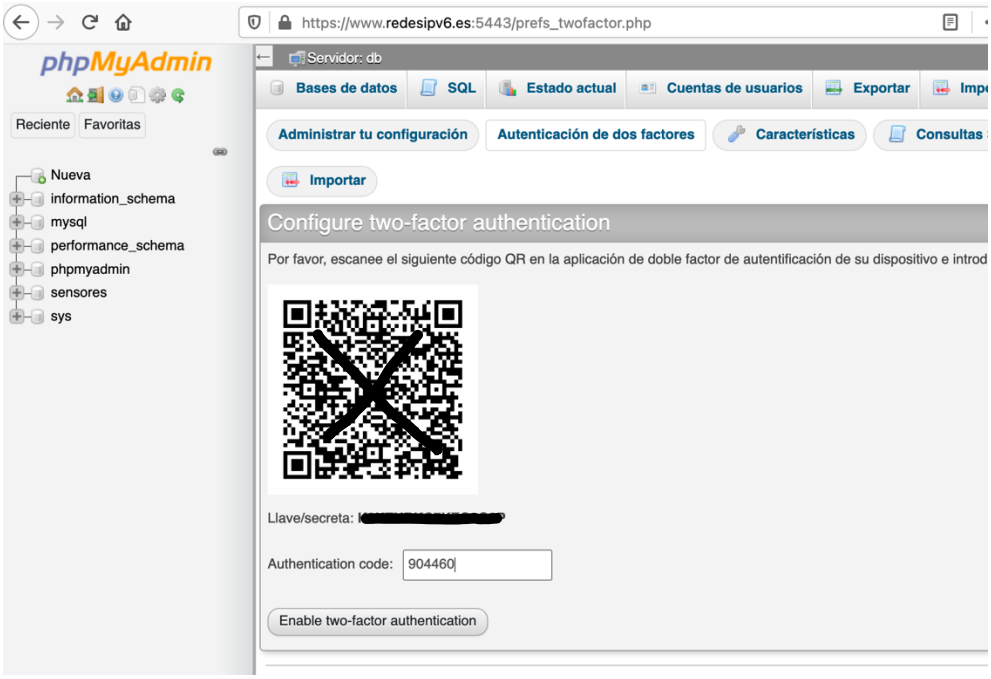
34. Acceso seguro a phpMyAdmin



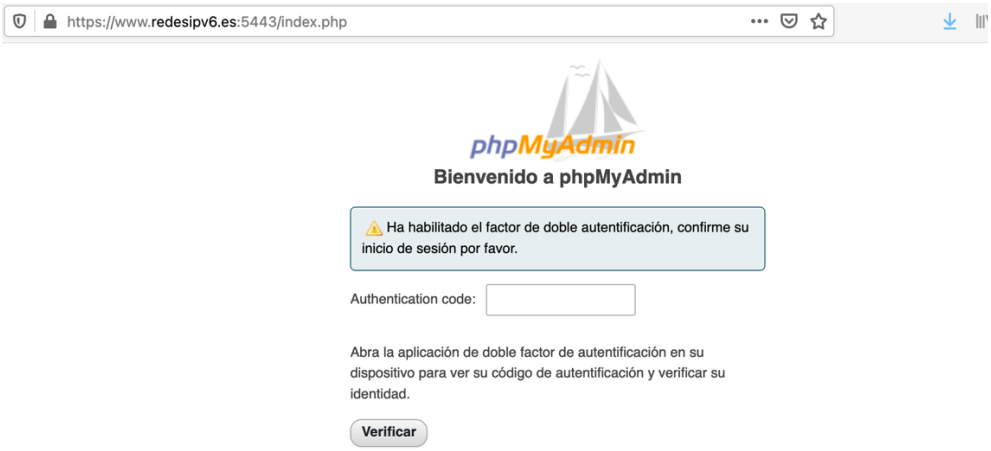
35. Configuración de doble facto de autentificación en phpMyAdmin



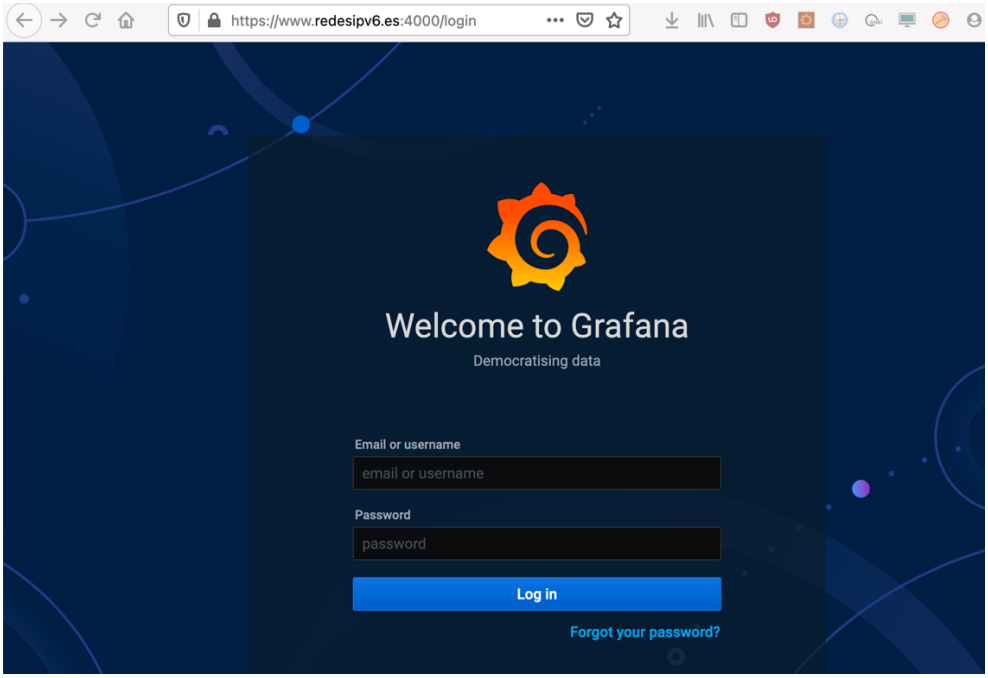
36. Verificación de configuración de doble factor de autentificación



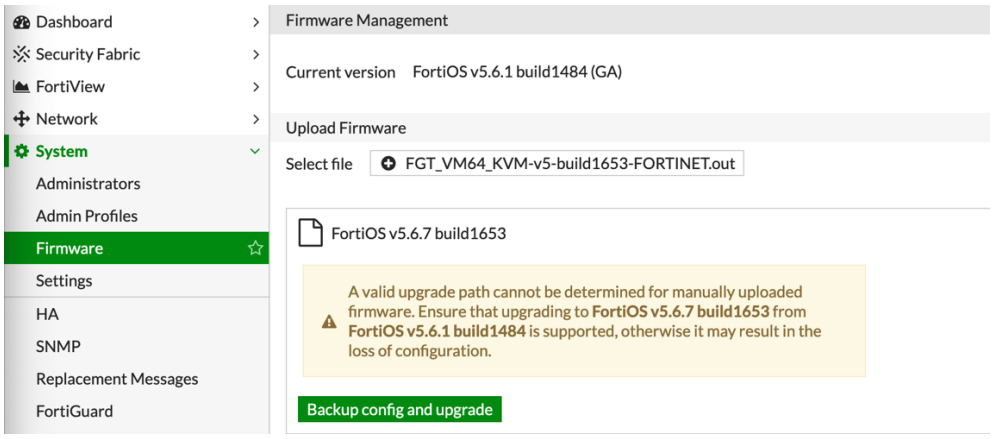
37. Solicitud de doble factor de autentificación



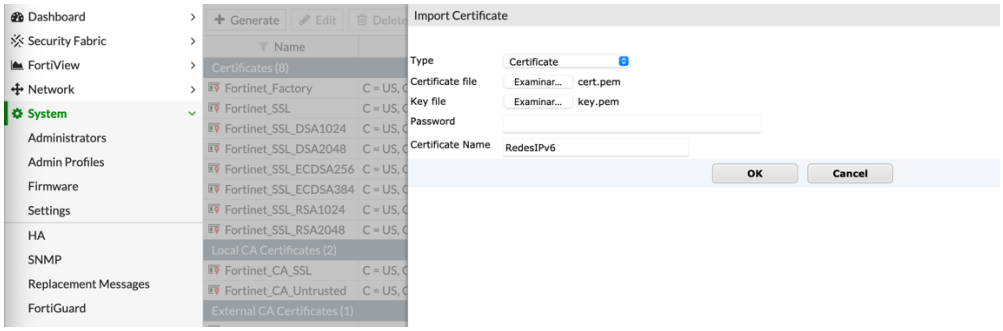
38. Acceso seguro a Grafana



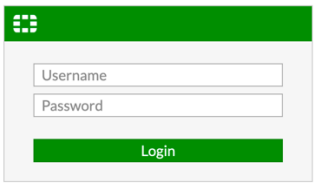
39. Actualización FortiGate



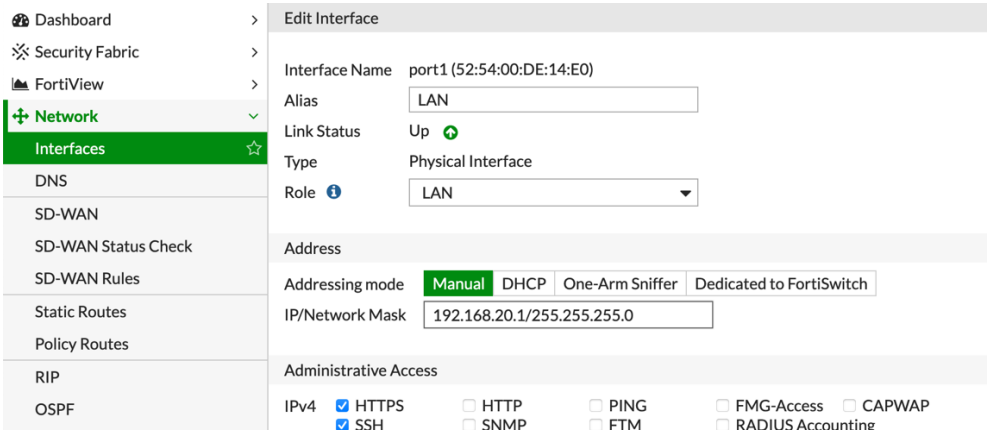
40. Certificado FortiGate



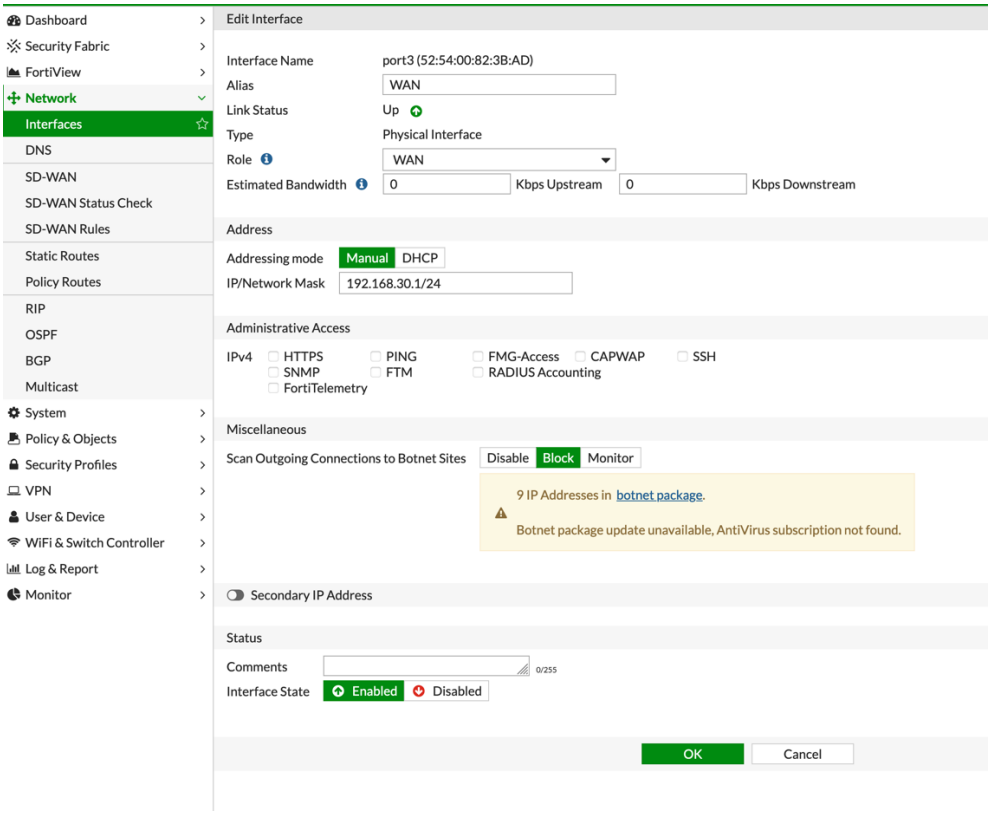
41. Acceso seguro al NGFW



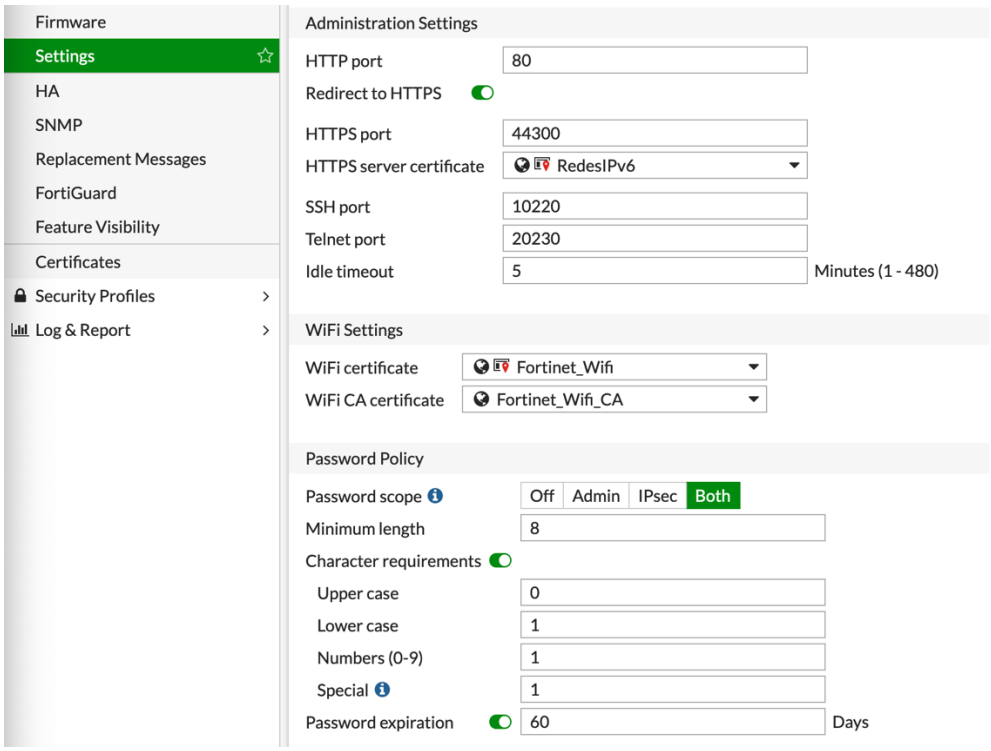
42. Configuración interfaz LAN



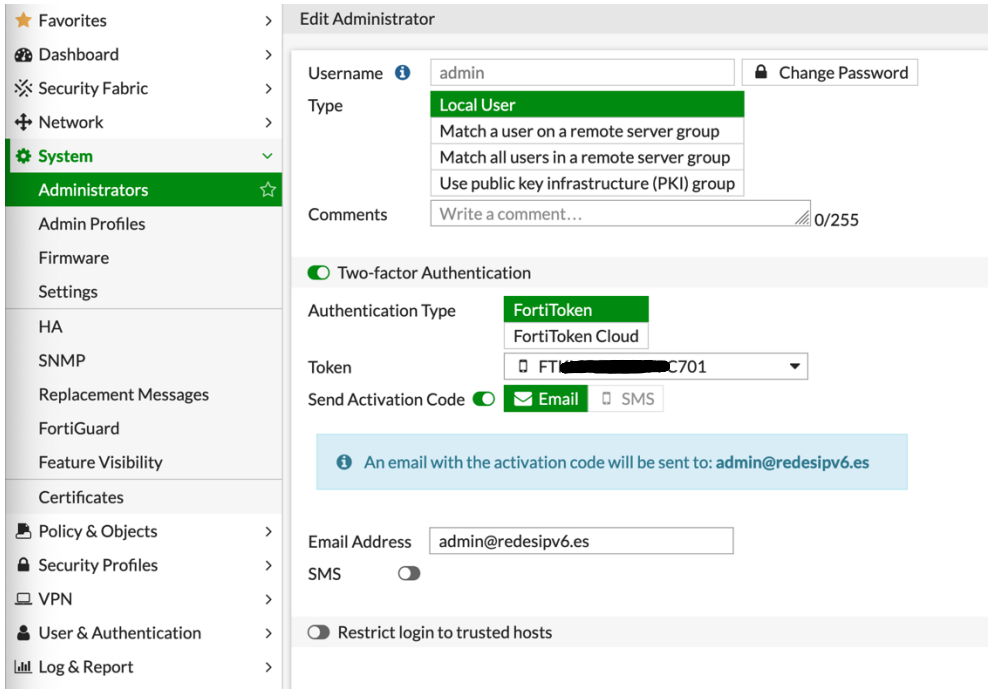
43. Configuración interfaz WAN



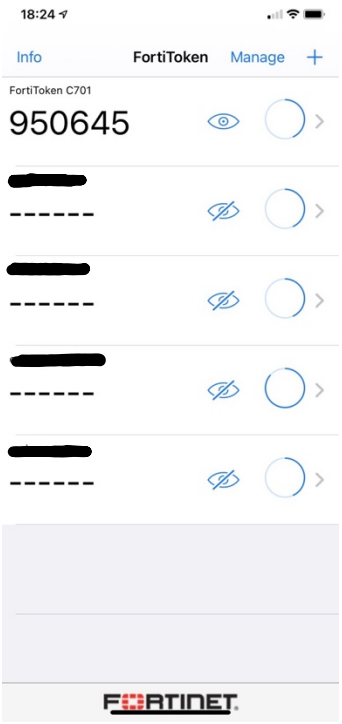
44. Configuración de gestión en NGFW



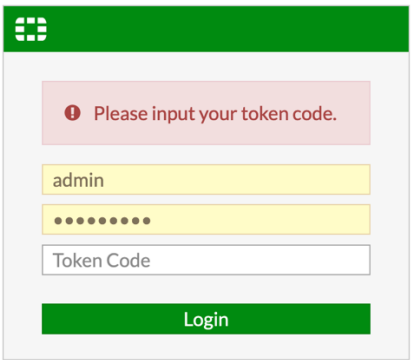
45. Configuración de doble factor de autentificación en NGFW



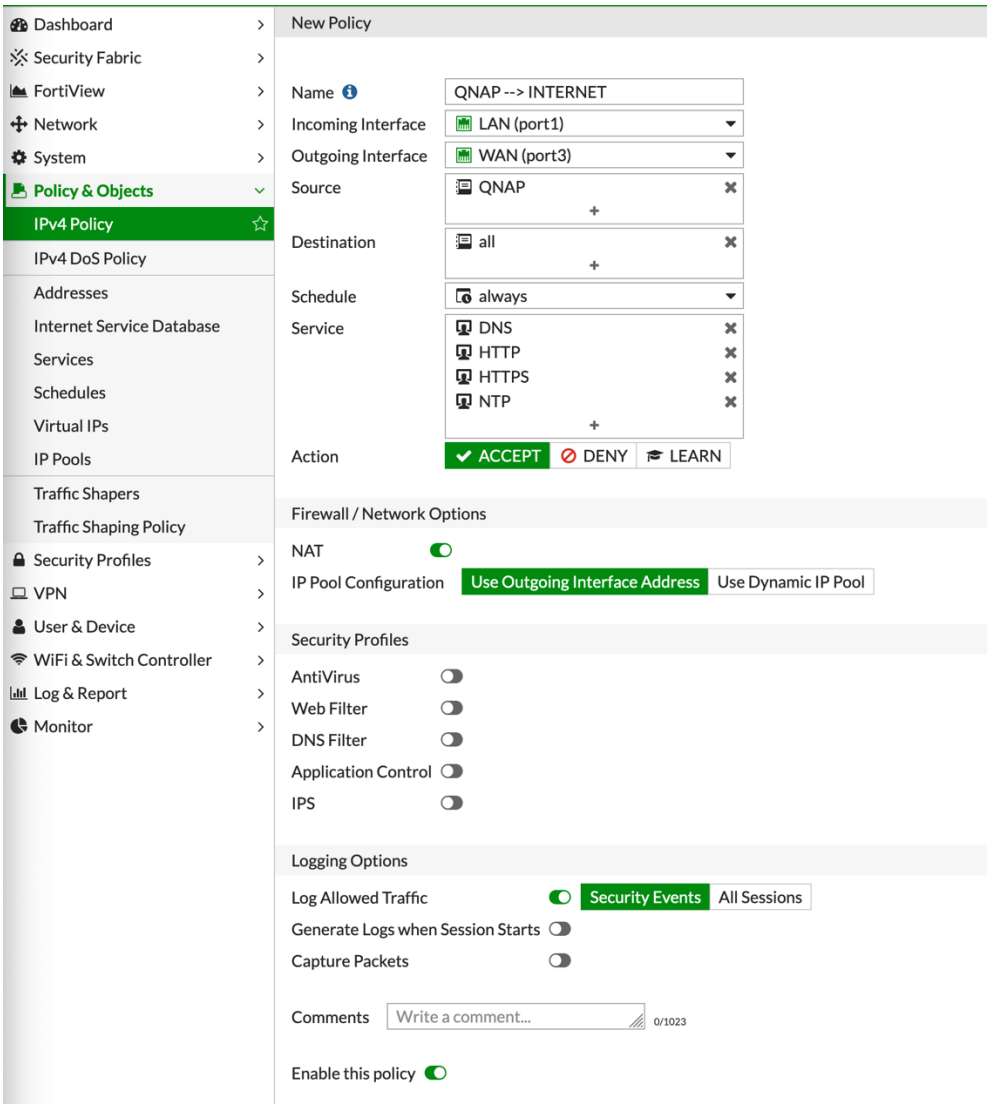
46. Código OTP mediante FortiToken



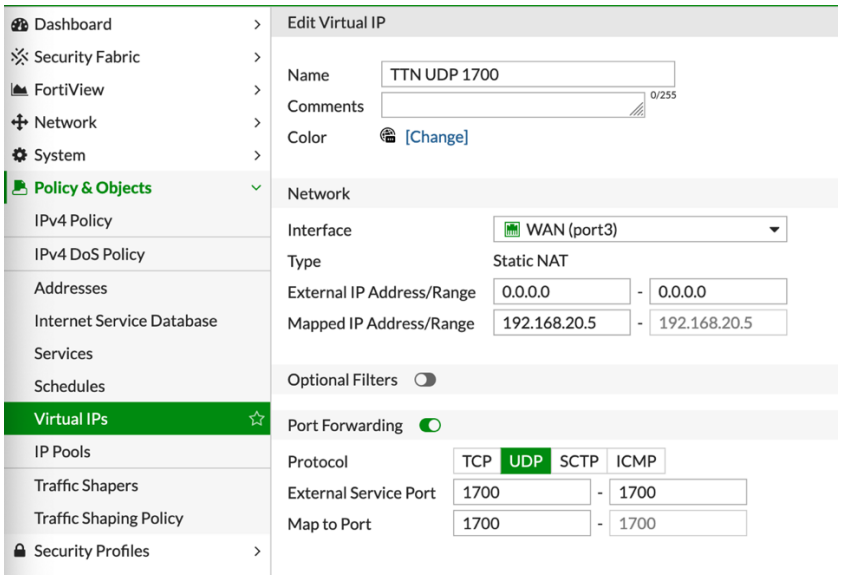
47. Inicio de sesión con OTP en NGFW



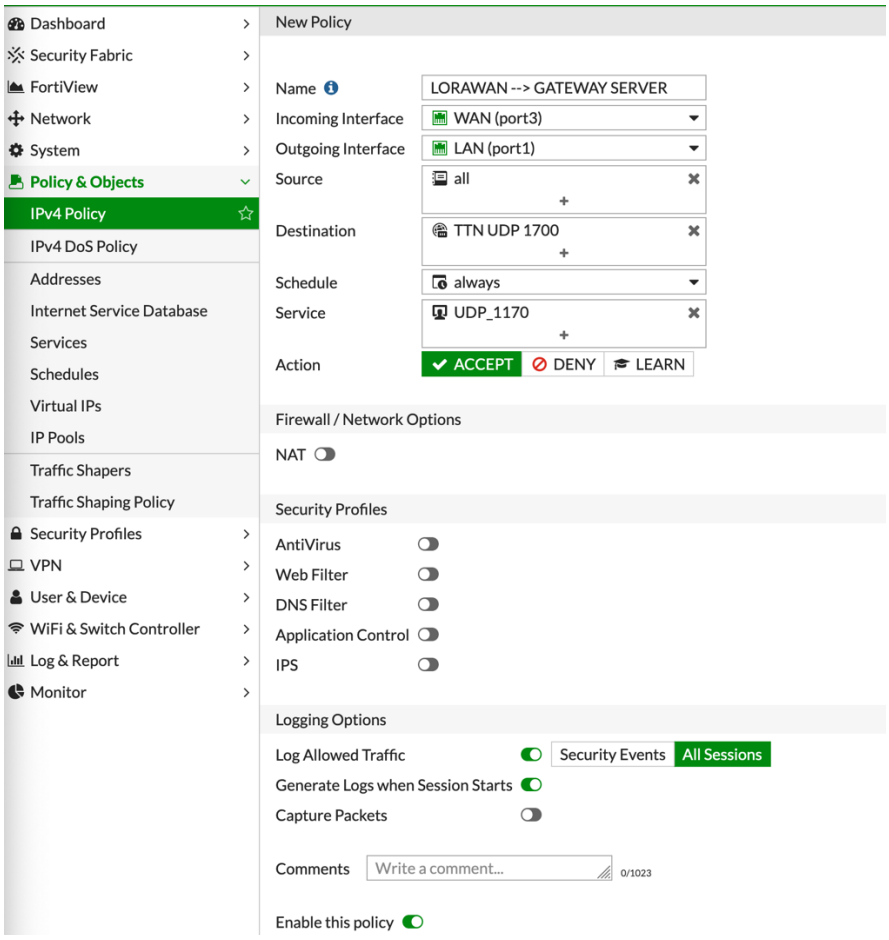
48. Política de seguridad QNAP → INTERNET



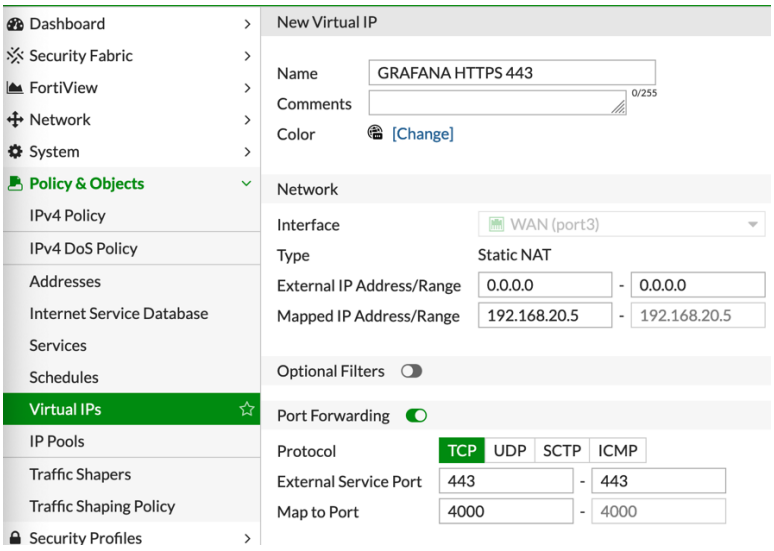
49. Virtual IP UDP 1700



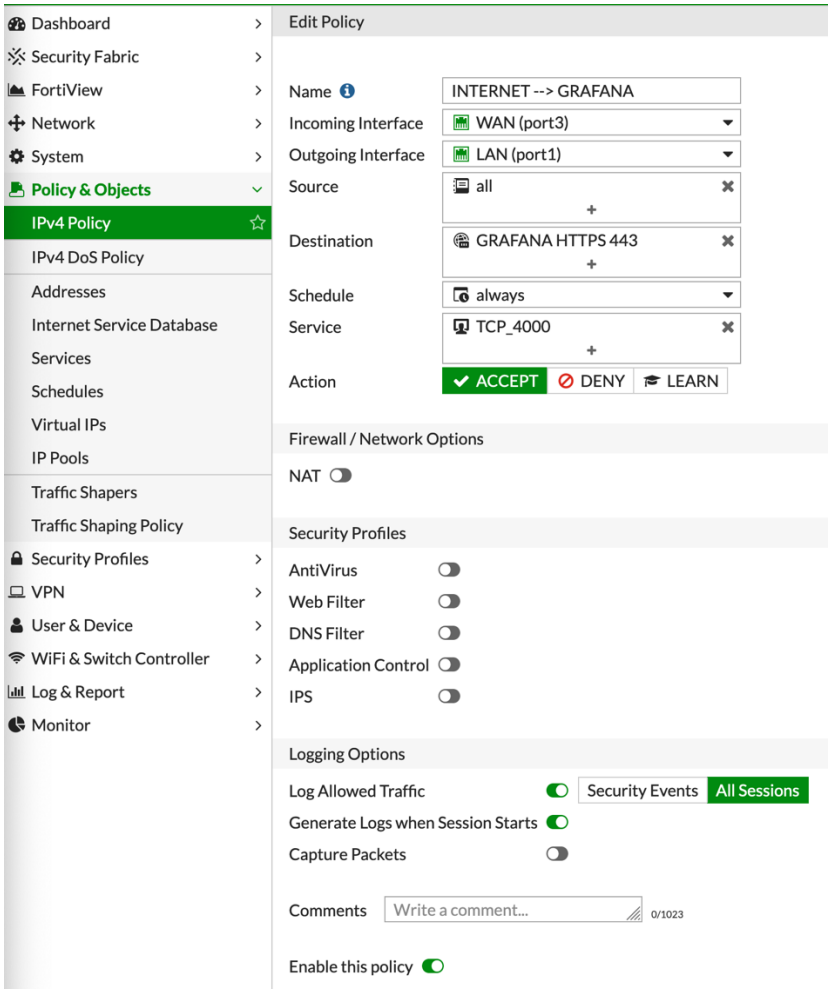
50. Política de seguridad LORAWAN → GATEWAY SERVER



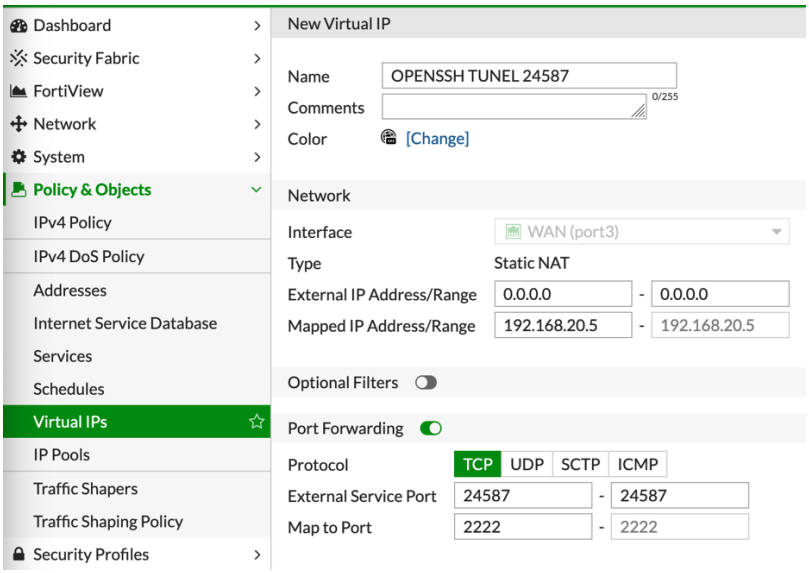
51. Virtual IP TCP 443 → TCP 4000



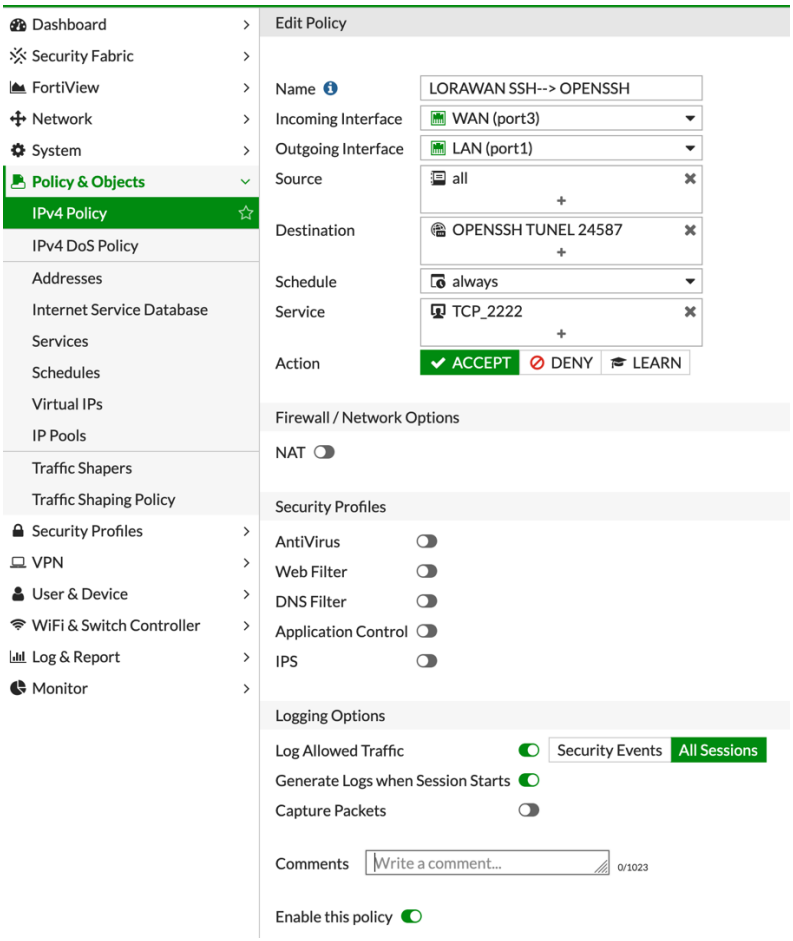
52. Política de seguridad INTERNET → GRAFANA



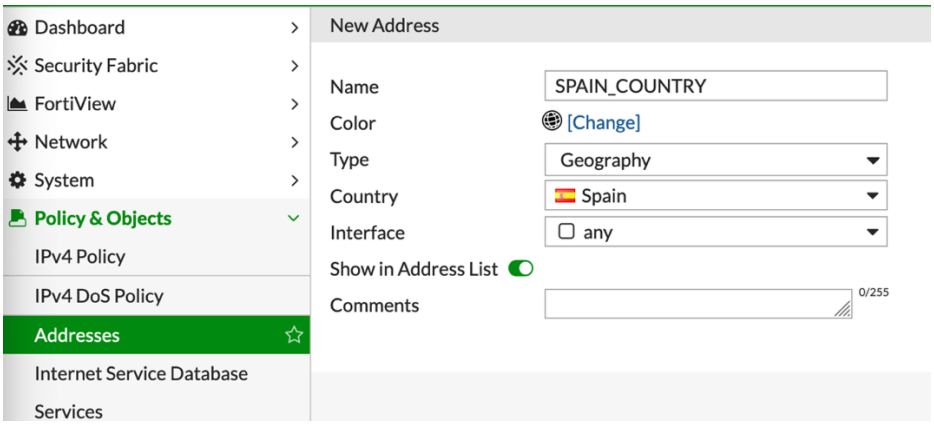
53. Virtual IP TCP 24587 → TCP 2222



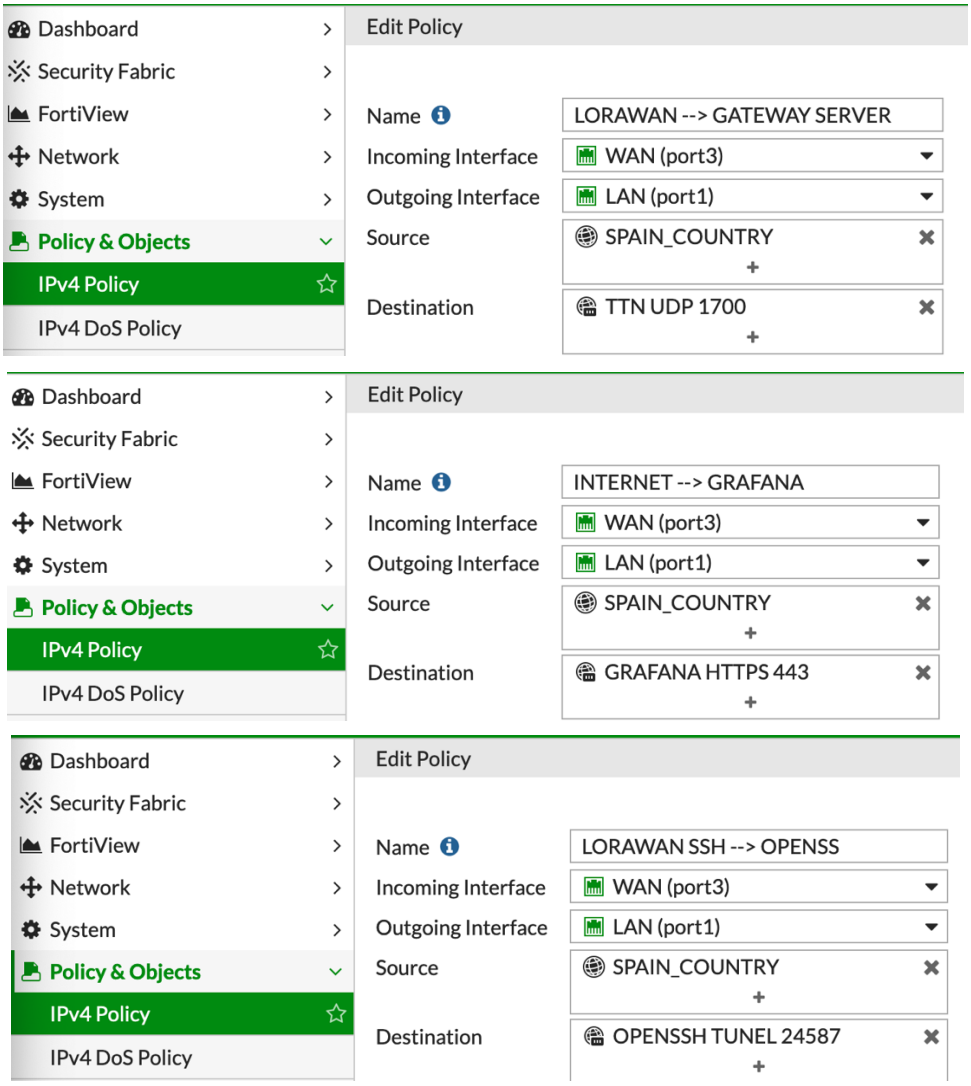
54. Política de seguridad INTERNET → OPENSSH



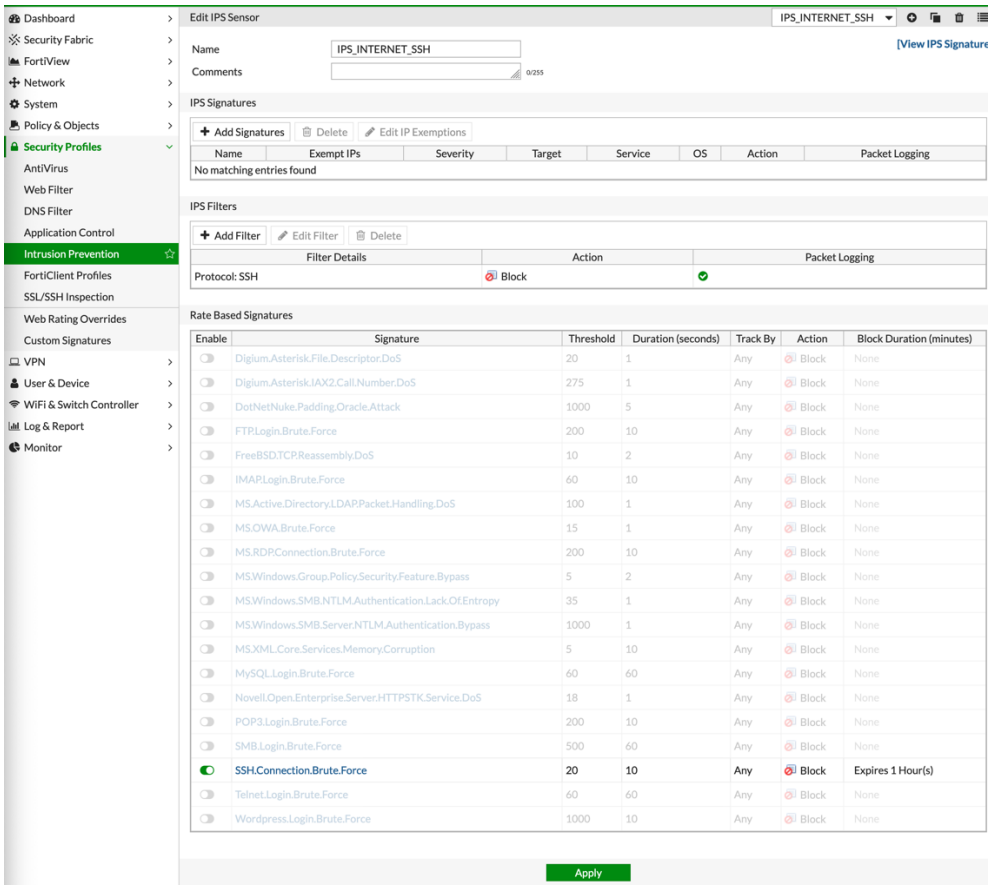
55. Objeto IP geográfico



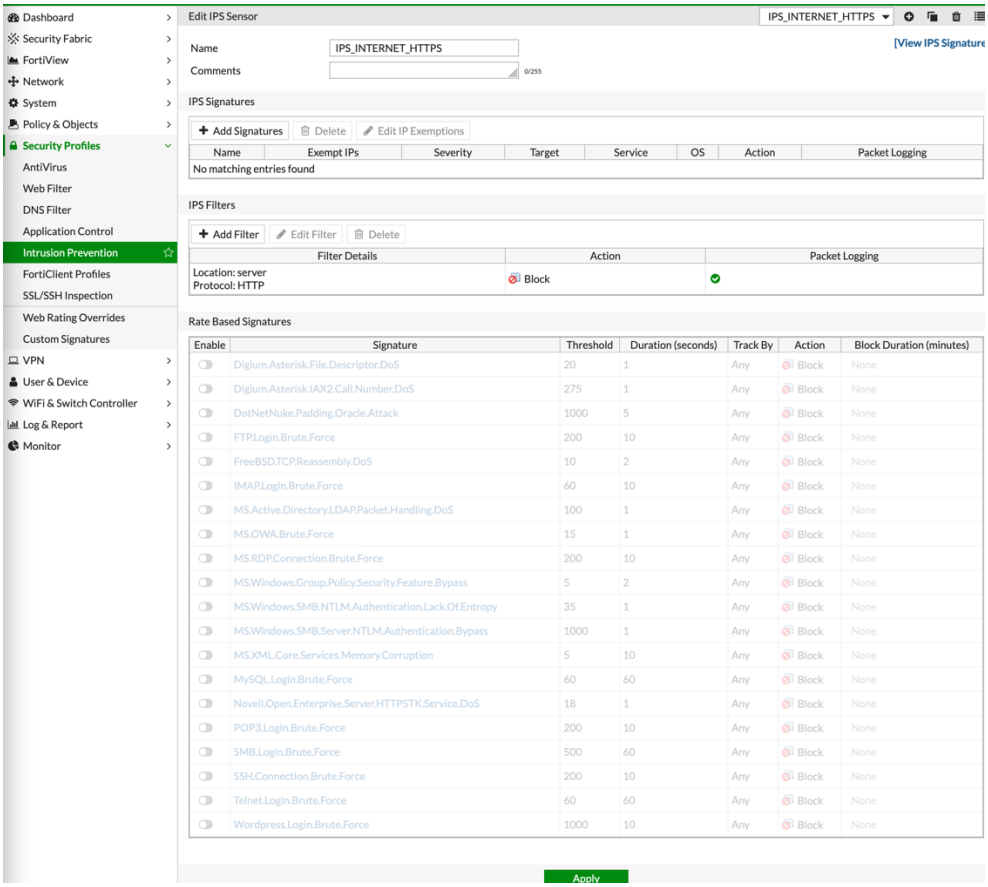
56. Políticas con **Source** geográfico



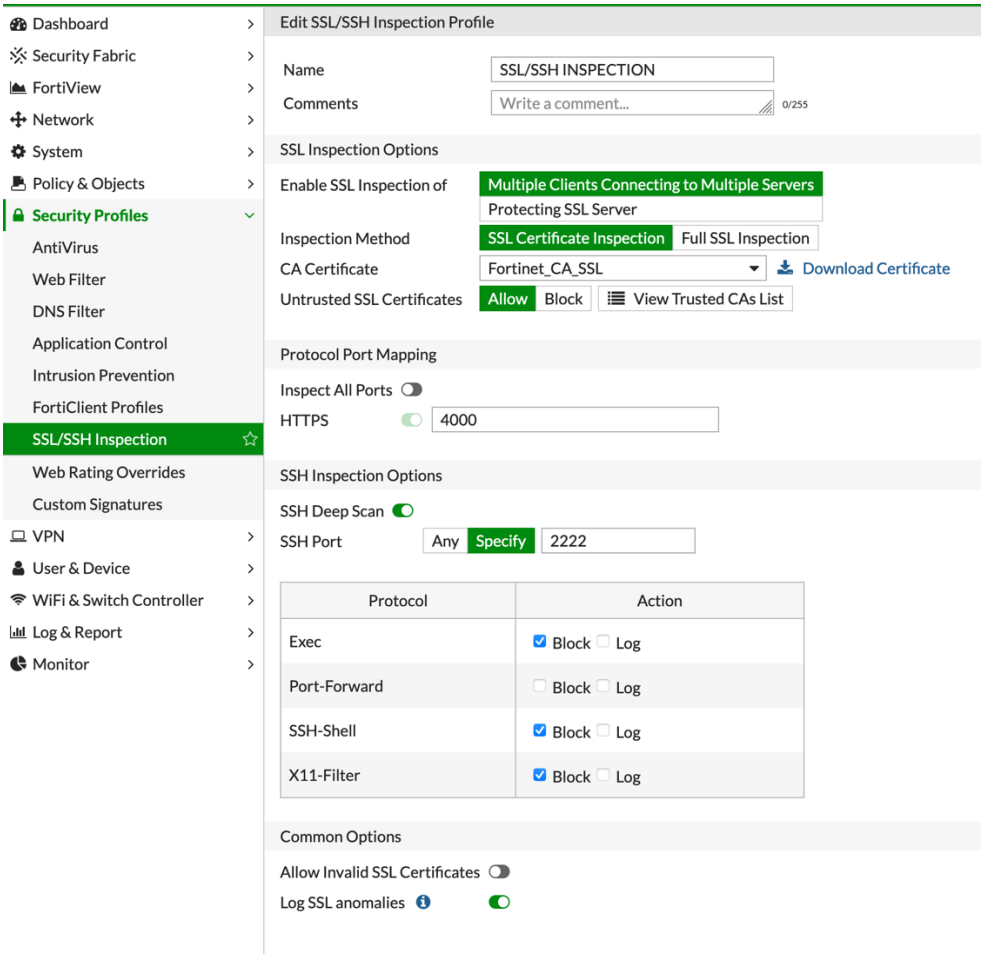
57. Política IPS para SSH



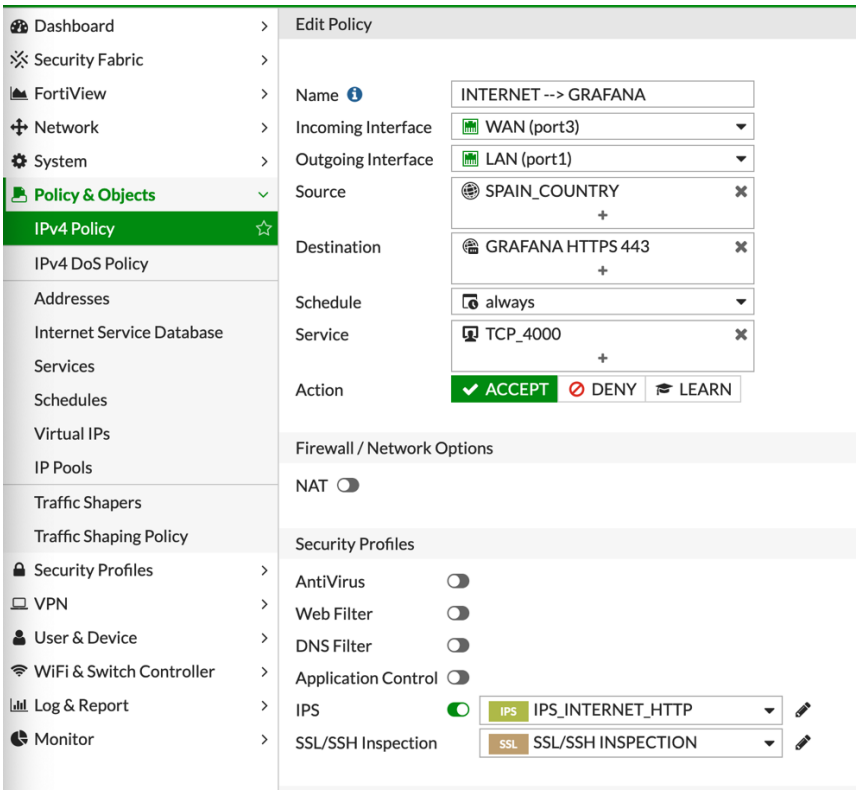
58. Política IPS para HTTP

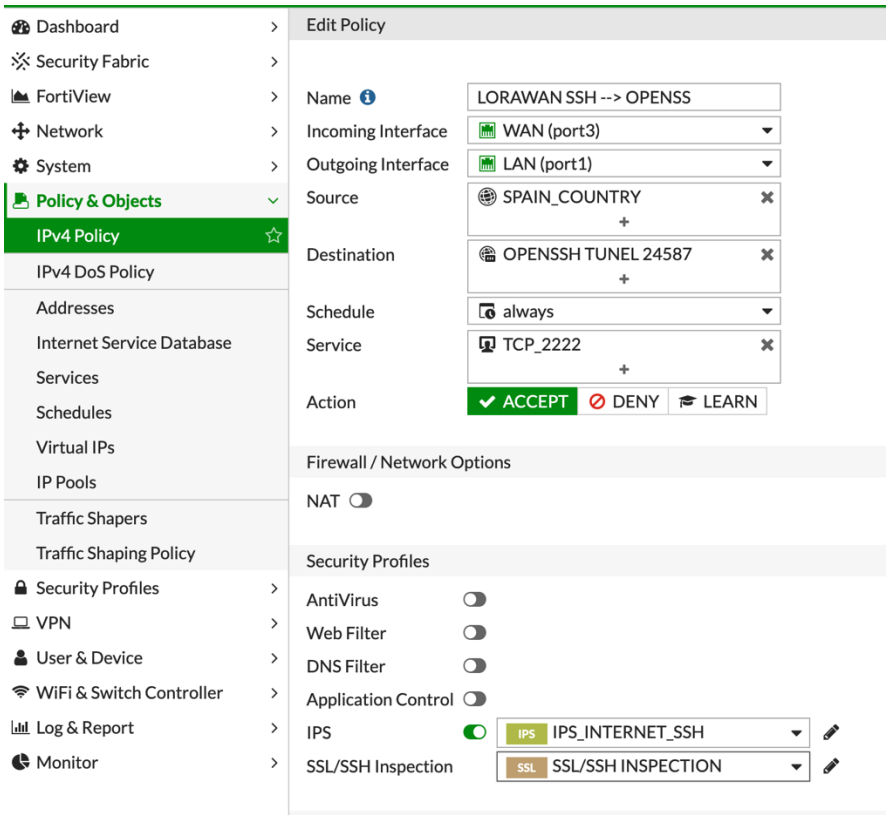


59. Políticas de inspección SSL/SSH

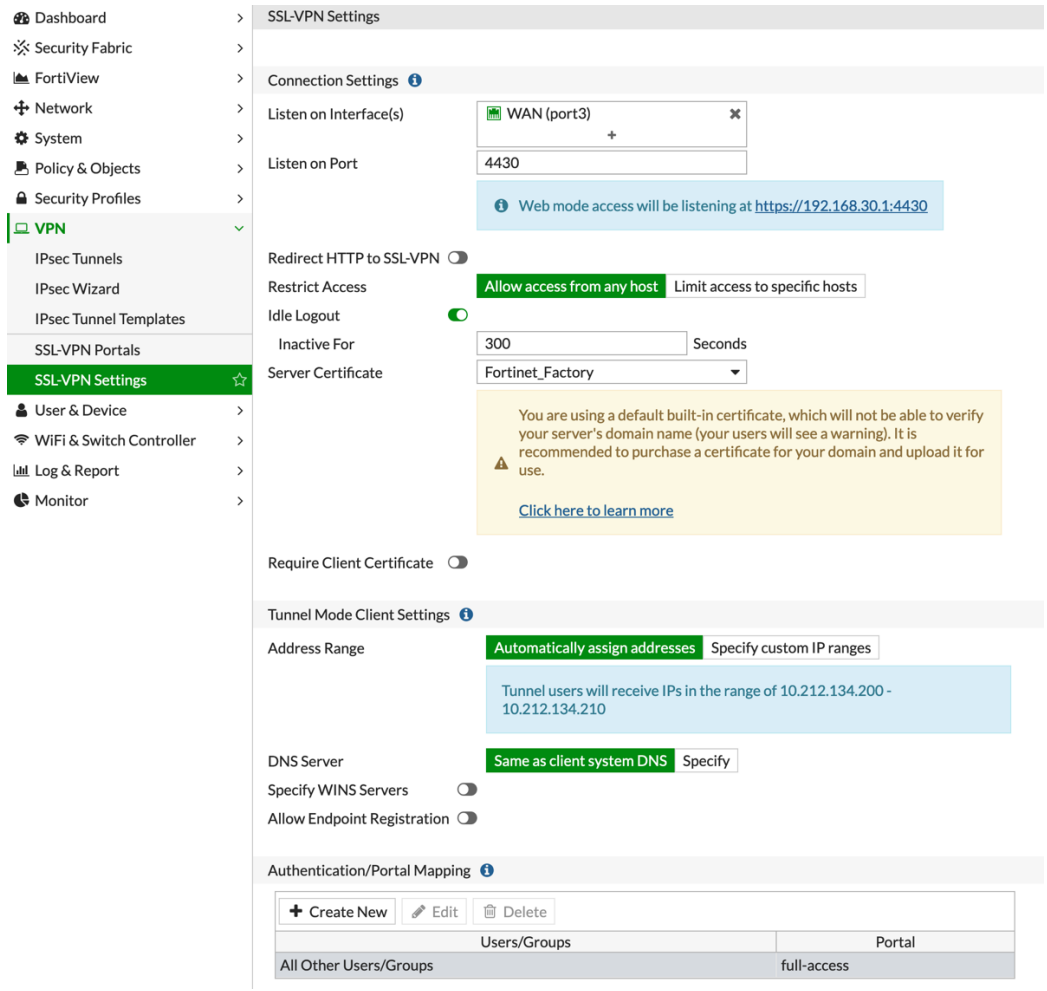


60. Asignación de políticas de inspección SSL/SSH e IPS

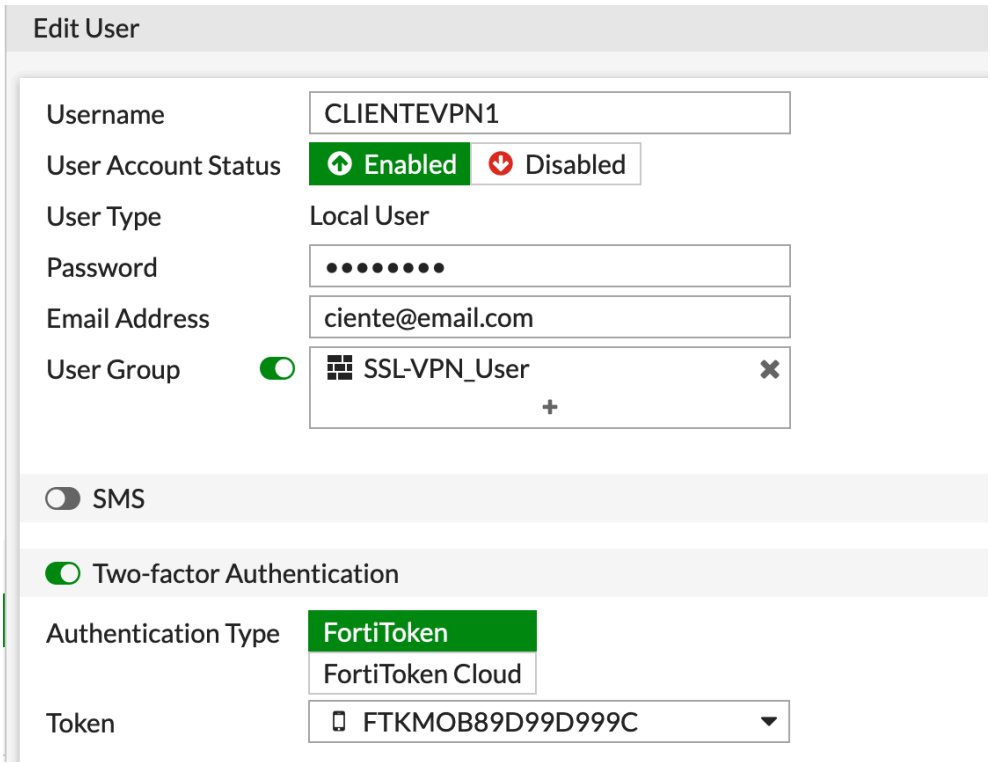




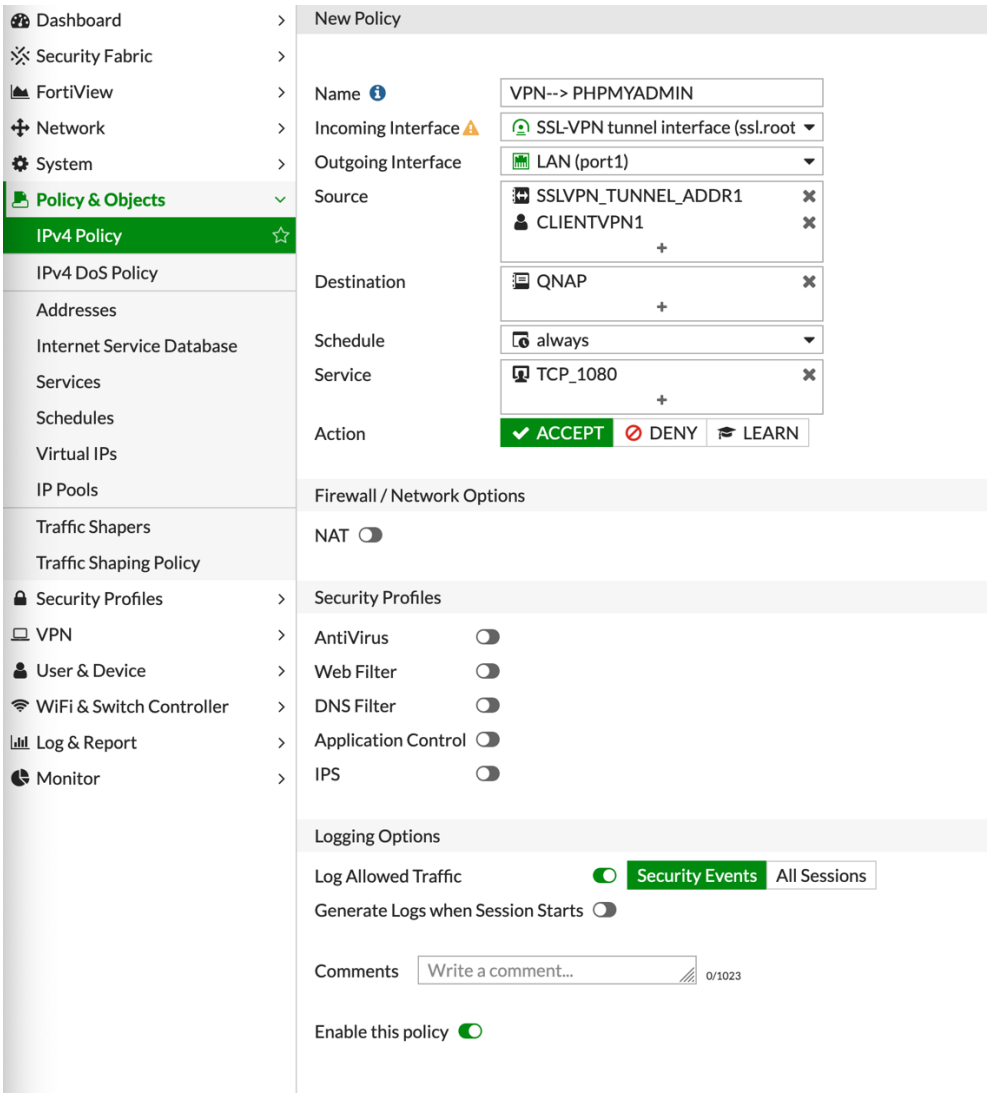
61. Configuración acceso VPN SSL



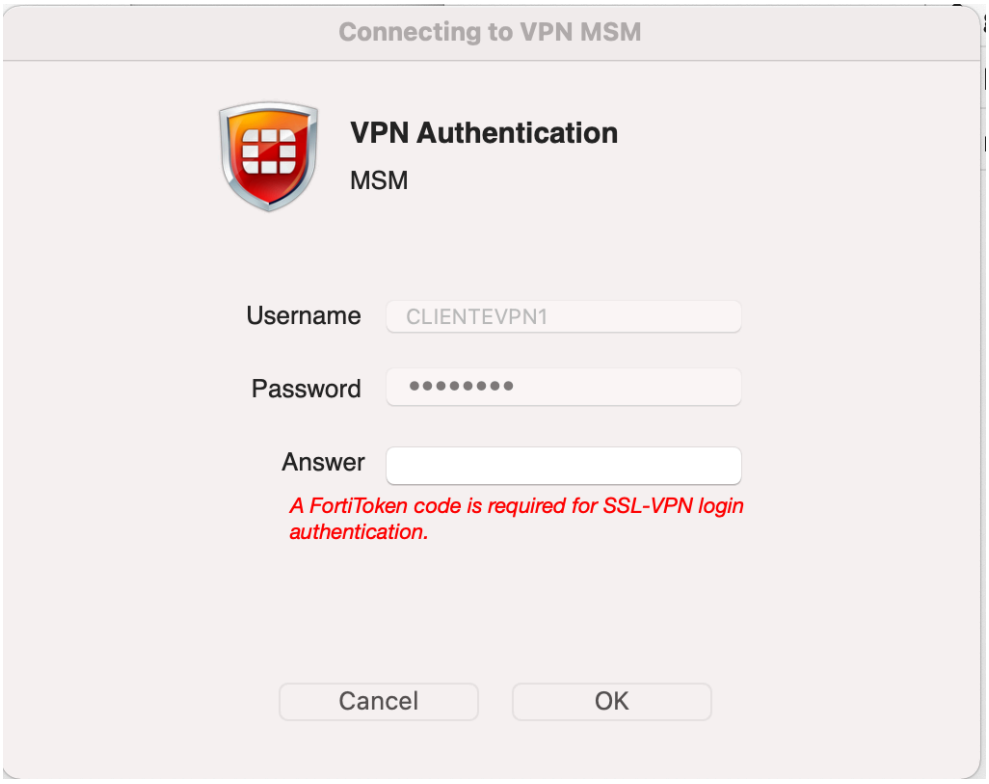
62. Configuración de usuario VPN con doble factor



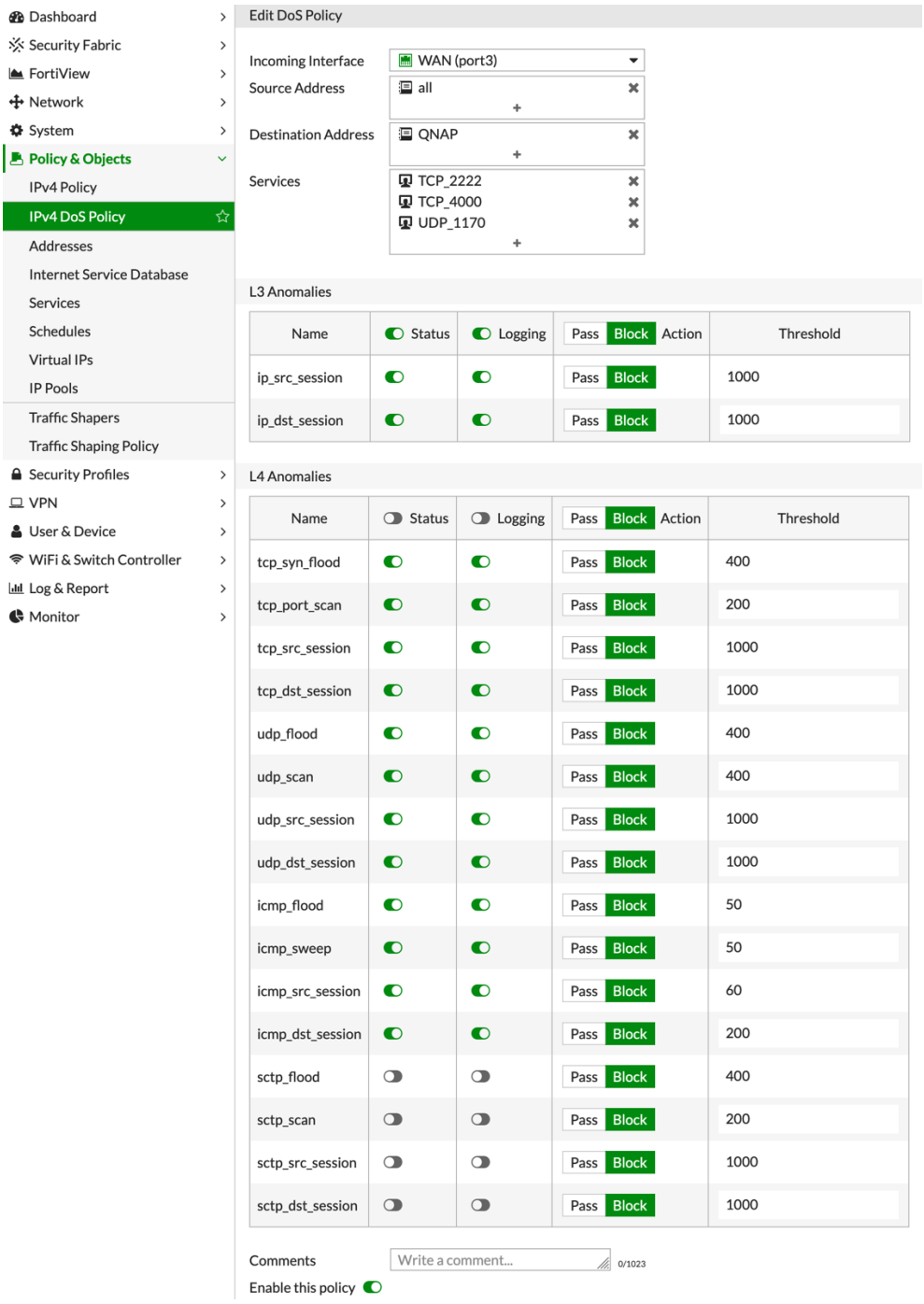
63. Política de seguridad VPN → PHPMYADMIN



64. Inicio de sesión VPN con OTP en NGFW



65. Política de DoS



66. Opciones de registro

Anexo D: Currículum Vitae

Fecha	Junio 2018 a la actualidad
Empresa	**Ingeniería de Sistemas para la Defensa de España (ISDEFE).**
Cargo ocupado	Ingeniero Sénior de Seguridad
Principales actividades y responsabilidades	En proyecto para la Agencia Estatal de Seguridad Aérea (AESA): Responsable de los proyectos de seguridad en las comunicaciones y sistemas de la información de la seguridad aérea, que comprenden desde la planificación, control y seguimiento de los proyectos según metodología PMP, pasando por la implementación de las arquitecturas y su posterior puesta en producción y el mantenimiento del ciclo de vida de la seguridad. Gestión de la seguridad en redes cableadas basadas seguridad perimetral, control de acceso a las redes (NAC) y sistemas de acceso remoto VPN. Auditoria y análisis de seguridad de las redes cableadas, inalámbricas y de acceso remoto VPN. Diseño e implantación de plataforma SIEM para la centralización de eventos de seguridad. Responsable del proyecto de migración de balanceadores Web e implementación de seguridad WAF en Sede Electrónica. Asesoramiento a la jefatura del área en la implementación de las políticas de seguridad basadas en el ENS, soporte en preparación de pliegos de prescripciones técnicas (PPT), elaboración de informes ejecutivos del área de seguridad y aplicación de la normativa legal en área de seguridad. Responsable de la gestión documental del área de seguridad. Realización de documentación de proyecto y técnica. Asistencia al resto de áreas TIC de la Agencia en la implementación de medidas de seguridad en los entornos de gestión y producción. Gestión de equipo de trabajo del área de seguridad y comunicaciones.
Fecha	Julio 2014 a junio 2018
Empresa	**ATOS IT SPAIN (antes BULL ESPAÑA)**
Cargo ocupado	Ingeniero Sénior de Telecomunicaciones y Seguridad
Principales actividades y responsabilidades	Responsable de la realización de proyectos de Telecomunicaciones, que comprenden desde el diseño, la planificación y la implementación de las arquitecturas de redes Wireless, entornos de seguridad perimetral y control de acceso (NAC). Ingeniería de proyecto de IoT, LoraWAN, Sigfox. Despliegue y migración de Balanceadores Web en entornos de alta seguridad. Diseño de infraestructuras de autentificación basado en PKI. Auditoria y análisis de seguridad de redes Wireless. Estudios y planificación Wireless. Preventa y propuestas a clientes. Análisis y resolución de problemas en las redes 802.11.
Fecha	Diciembre 2012 a Julio 2014
Empresa	**Grupo NCL**
Cargo ocupado	Ingeniero Sénior de Telecomunicaciones
Principales actividades y responsabilidades	Responsable de la realización de proyectos de Telecomunicaciones, que comprenden desde el diseño, la planificación y la implementación de las arquitecturas de redes, seguridad perimetral, servicios de VoIP, redes Wifi y WIMAX y entornos de monitorización. Realización análisis de cobertura de redes Wifi y WIMAX. Realización de labores de preventa en apoyo al departamento de ventas, así como la realización de las ofertas técnicas de los proyectos.
Fecha	Diciembre 2007 a mayo 2012
Empresa	**Hewlett Packard CDS**
Cargo ocupado	Ingeniero de Telecomunicaciones
Principales actividades y responsabilidades	En proyecto de la DGT, las principales tareas realizadas son: Diseño, instalación, gestión e implementación de la arquitectura de de redes LAN y MAN, despliegue de ToIP y entorno de gestión de redes. Responsable de la implantación de las políticas de seguridad NAC. Despliegue de la solución de virtualización de servidores. Implantación de balanceadores de trafico e implantación de la solución tecnológica VSS de Cisco. Implantación y configuración equipamiento DWDM. Gestión de dispositivos de seguridad perimetral y de clasificación de trafico. Análisis y resolución de incendias de nivel 3 en el entorno LAN, ToIP y seguridad perimetral. Gestión de personal de nivel 1 y 2. Gestión de los proyectos del área de Redes y ToIP.