UNIVERSIDAD DE VALLADOLID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA DE TECNOLOGÍAS ESPECÍFICAS DE TELECOMUNICACIÓN

Implementación de medidas de ciberseguridad en un vehículo conectado

Autor:

D. Álvar Alonso González

Tutor:

Dr. D. Juan Carlos Aguado Manzano

D. Adrián Rojo Becerril

VALLADOLID, JULIO 2022

TRABAJO FIN DE GRADO

TÍTULO:	**Implementación de medidas de ciberseguridad en un vehículo conectado**
AUTOR:	**D. Álvar Alonso González**
TUTOR:	**Dr. D. Juan Carlos Aguado Manzano** **D. Adrián Rojo Becerril**
DEPARTAMENTO:	**Teoría de la Señal y Comunicaciones e Ingeniería Telemática**

TRIBUNAL

PRESIDENTE:	**Dr. D. Ignacio de Miguel Jiménez**
VOCAL:	**Dr. D. Ramón José Durán Barroso**
SECRETARIO:	**Dr. D. Juan Carlos Aguado Manzano**
SUPLENTE:	**Dr. Dª Noemí Merayo Álvarez**
SUPLENTE:	**Dr. D. Patricia Fernández Reguero**

FECHA:
CALIFICACIÓN

RESUMEN

La evolución de las tecnologías de la información y la comunicación está facilitando el acceso a todo tipo de herramientas que mejoran la calidad de vida de la sociedad. Algunas de estas herramientas facilitan la movilidad de las personas de una manera sostenible y reduciendo la huella de contaminación. Durante el desarrollo de estas herramientas se debe planificar un plan de ciberseguridad que proteja la información que comparten los usuarios con las mismas. En este Trabajo de Fin de Grado se realiza un análisis en busca de fallas de seguridad de un sistema de *carsharing* que incorpora tecnologías de vehículo autónomo. Este sistema se comunica empleando el protocolo MQTT, por lo que el objetivo principal de este proyecto es proteger esta comunicación implementando un canal seguro a través del protocolo TLS (*Transport Layer Security*). Otro de los objetivos principales de este proyecto es el de fortificar el servidor que aloja el *software back-end* de este sistema de *carsharing* mediante el despliegue de un *firewall* que permita controlar el acceso a este componente.

ABSTRACT

The growth of the information and communication technologies is providing access to all kinds of tools that improves the life quality of society. Some of these tools provides sustainable mobility which reduces the pollution footprint. During the development of these tools, a cybersecurity plan must be organized in order to protect the information that users share by using them. This Final Degree Project analyses the security flaws in a carsharing system which includes autonomous vehicle technologies. This system uses the MQTT protocol for communications, so the main objective of this project is to protect these communications by implementing a secure channel through TLS (Transport Layer Security) protocol. Another main objective of this project is to reinforce the system which host the back-end software of this carsharing system by deploying a firewall to control access to this component.

Palabras clave:

Ciberseguridad, MQTT, TwizyLine, TLS, firewall, cifrado.

AGRADECIMIENTOS

Me gustaría agradecer, en primer lugar, a mi tutor Juan Carlos Aguado por ofrecerme la oportunidad de poder investigar sobre ciberseguridad en este proyecto y por apoyarme y estar dispuesto a escucharme durante la realización de este trabajo.

En segundo lugar, agradecer a mi cotutor Adrián Rojo por saber guiar este proyecto por el buen camino y por aportar nuevas ideas y soluciones durante todo este proceso.

En último lugar, agradecer a mi cotutor en la sombra, Adrián Mazaira, por el apoyo durante el entendimiento de este proyecto y por estar siempre dispuesto a ofrecer su mano.

Por último, gracias al equipo del LAB28, la compañía siempre es importante cuando se realiza esta clase de proyectos y más cuando existe buena vibra.

Índice de contenidos

Índice de Figuras

Índice de Tablas

1 INTRODUCCIÓN

1.1 Introducción y motivación

La principal motivación de este proyecto de fin de grado nace de mejorar el proyecto TwizyLine [1]. Este proyecto surge del concurso internacional llamado "Twizy Contest 2020", cuyo objetivo era desarrollar una solución de movilidad sostenible para los Juegos Olímpicos de París 2024, usando para ello el vehículo Renault Twizy [2]. TwizyLine es el proyecto que representó a España en este concurso, y se basa en una plataforma de *carsharing*, que "es un servicio de coche compartido que ofrece a sus usuarios una flota de vehículos durante cortos periodos de tiempo en donde varias personas usan el mismo vehículo reduciendo el número de vehículos necesarios para satisfacer una movilidad" [3]. Esta idea de *carsharing* emplea tecnologías de vehículo autónomo para aparcar los vehículos del servicio completamente solos. Además, el equipo de estudiantes que diseñaron el proyecto desarrolló también una aplicación móvil para el uso de esta idea, por lo que cualquier usuario que quisiese alquilar un vehículo, solo tendría que realizar un *login* clásico introduciendo su correo electrónico y contraseña, para después, buscar el aparcamiento TwizyLine más cercano, elegir un Twizy, y este se dirigiría de forma autónoma al lugar de salida para que el usuario pudiese coger los mandos del vehículo y dirigirse al lugar deseado [4].

Figura 1. Logotipo y símbolo del proyecto TwizyLine [1].

Gracias al continuo avance de las tecnologías, el proyecto de TwizyLine pudo llegar a ser desarrollado en un ámbito educativo como lo es el de una universidad. Este proyecto fue desarrollado con el fin de implementar una idea sostenible de *carsharing*.

Para realizar este proyecto se realizó: una arquitectura acorde, un diseño de la mensajería, un sistema *back-end*, un sistema *front-end*, modificaciones en la arquitectura eléctrica del vehículo (Ver Figura 2), etc. Pero no se tuvo en cuenta los peligros de seguridad informática a los que se podría enfrentar el sistema, como las posibles repercusiones de no poder hacer frente a un ciberataque.

Según la CSRC (Computer Security Resource Center), un ciberataque es "un ataque, a través del ciberespacio, cuyo objetivo es interrumpir, inutilizar, destruir o controlar maliciosamente un entorno o infraestructura informática; o destruir la integridad de los datos o robar información controlada" [5].

Los ciberataques, por tanto, son una amenaza que debe considerarse seriamente de cara al bien rendimiento de una empresa o negocio. Aunque las empresas que sufren ciberataques son reticentes a dar información sobre los mismos con la idea de no animar a más *hackers* a intentarlo, algunos casos se han hecho públicos, demostrando la extrema peligrosidad de los mismos a todos los *stakeholders* de una empresa, esto es, no solo accionistas de la empresa, sino trabajadores, clientes, etc.

Figura 2. Modificaciones y adiciones instaladas en los vehículos Twizy [1].

Un primer ejemplo lo encontramos en el caso del ataque con Expetr, también conocido como NotPeta, que fue un ciberataque que empezó a expandirse en 2016 y fue dirigido a ordenadores con el sistema operativo Windows. Este ciberataque se aprovechaba de un error de este sistema operativo y permitía hacerse con el control del ordenador después de que el usuario reiniciase el sistema. Solo se podía acceder de nuevo al sistema una vez se pagase una cifra de dinero por el rescate del sistema. Este ciberataque consiguió bloquear organizaciones localizadas por todo el mundo, como FedEx o DHL, pero principalmente fue un ataque dirigido a organizaciones ucranianas [6]. Se calcula que las empresas afectadas por este ataque llegaron a pagar un total de 10.00 millones de dólares a los responsables del ciberataque [7].

Un segundo ejemplo se puede encontrar cuando La Administración de Veteranos de EE.UU sufrió un importante ciberataque. En este caso el ciberataque consistió en el robo de los datos de más de 26 millones de veteranos, personal militar y familias que se encontraban sin cifrar dentro de la base de datos. El coste de este ataque se estima en casi 500 millones de dólares, a parte del escándalo público que generó la noticia [7].

De hecho los ataques de este tipo no se han reducido debido a un mejor control de la ciberseguridad, sino que se siguen produciendo incluso con mayor intensidad, tal y como se puede ver en la Figura 3 con el creciente coste que conlleva la pérdida de datos en los últimos 5 años.

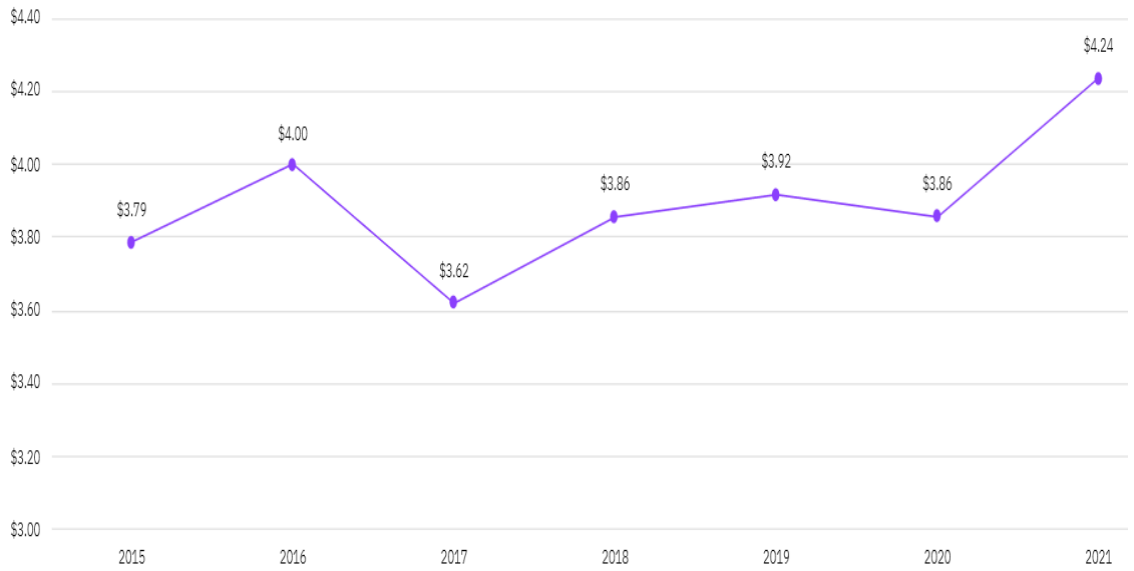

Figura 3. Coste medio en millones de dólares de la violación de datos o escape de datos [8].

Estos son solo dos ejemplos de lo que puede ocurrir si no se tienen en cuenta los peligros de no proteger debidamente un sistema informático. No solo es la cantidad de dinero que puede llegar a perder una organización, sino la pérdida de prestigio por el robo de datos personales de personas que confiaban en dar su información personal a estas organizaciones, junto con el posible daño que cada una de estas personas pueda sufrir por la filtración de sus datos. Es por esto por lo que es importante considerar la peligrosidad de los ciberataques cuando se está desarrollando una idea que hace uso de tecnologías informáticas.

1.2 Objetivos

El objetivo principal de este trabajo fin de grado es analizar la situación global de ciberseguridad del proyecto TwizyLine, y actuar en consecuencia con el fin de proteger cualquier tipo de información que pertenezca a este sistema. Por ello, se han fijado los dos siguientes objetivos principales:

- Análisis del estado del sistema de seguridad del proyecto TwizyLyne.
- Implementación de las medidas de seguridad necesarias con respecto al análisis anterior.

1.3 Pasos y métodos

Los pasos seguidos para llevar a cabo este proyecto fueron principalmente los siguientes:

1. Etapa de análisis y estudio: En un primer paso se realizó un estudio sobre las diferentes amenazas que podían existir en un sistema como el que representa TwizyLine. Esto no solo supone conocer cuáles son los niveles más básicos de protección que se recomiendan, sino también estudiar cuáles son los más adecuados al proyecto TwizyLine. Por otra parte, esto implicaba un estudio de los datos que manejaba la aplicación y el sistema para determinar el nivel de fortificación adecuado.
2. Etapa de definición de la estructura del proyecto: Después de la etapa de estudio, fue preciso determinar qué partes del proyecto TwizyLine eran críticas para ser

protegidas. El proyecto es lo suficientemente amplio como para no poder acometer de manera integral su seguridad en un único trabajo fin de grado como el que se presenta. Es por ello que en esta fase de determinaron las siguientes cuestiones clave:
    a. Elegir que partes del sistema eran críticas para ser aseguradas durante este Trabajo Fin de Grado.
    b. El análisis y estudio realizado en la fase anterior permitió ver que existían muchos niveles de protección (a partir de ahora fortificación) para los componentes elegidos. Por ellos, fue necesario definir el nivel de fortificación.
    c. Debido a la gran variedad de herramientas software que actualmente existen para lograr la fortificación de los sistemas informáticos, fue necesario seleccionar las que consideramos más adecuadas para nuestro sistema teniendo en cuenta el nivel de fortificación deseado y los límites materiales a los que nos enfrentamos en el desarrollo del proyecto.
3. Etapa de desarrollo del proyecto: Una vez elegido los sistemas, el nivel de fortificación y las herramientas a utilizar, durante esta fase se implementaron las medidas elegidas.
4. Etapa de integración y pruebas: Una vez implementadas las medidas para fortificar el sistema, se hicieron las pruebas pertinentes para validar que el sistema seguía funcionando correctamente y cumplía con las medidas de seguridad implementadas.

Debido a la complejidad del proyecto, se utilizó una metodología de seguimiento constante del trabajo. Se usaron herramientas que permitían tener un mayor control del tiempo empleado a cada tarea asignada, como el empleo del Diagrama de Gantt, el cual permite visualizar un cronograma del proyecto. Se realizaron reuniones con cierta frecuencia para controlar los avances realizados y los problemas encontrados, y así tomar las decisiones necesarias para redefinir el rumbo del proyecto.

1.4 Recursos disponibles

En cuanto al *hardware*, se ha utilizado dos Raspberry Pi 3 modelo B+ y un ordenador con un sistema operativo Windows como apoyo para la realización de la documentación y la búsqueda de información necesaria para realización del proyecto.

Por otro lado, en este proyecto se ha partido de la versión anterior del sistema de TwizyLine, que ha servido de base para el desarrollo del mismo. Dicha versión se encontraba disponible en el repositorio GitHub del Grupo de Comunicaciones Ópticas de la Universidad de Valladolid [9].

Todo el desarrollo del código necesario para fortificar el sistema ha sido llevado a cabo usando las herramientas de texto facilitadas por el sistema operativo Raspbian, como el editor “nano” de Unix y el IDE (*Integrated Development Environment*) de Python “Thonny”. Para la parte del desarrollo de la aplicación se ha usado el IDE de Android “Android Studio”.

Todo el progreso ha sido seguido usando el repositorio GitHub. Las pruebas de implementación han sido llevadas a cabo en dos Raspberry Pi 3 modelo B+.

1.5 Estructura de la memoria

Este documento explica la integración de medidas que permitan fortificar el sistema perteneciente al proyecto TwizyLine. Este trabajo está seccionado en seis capítulos, siendo el presente el primero de estos. La disposición del resto del trabajo es la siguiente:

- Capítulo 2: Estado del arte. En este capítulo se presenta como está desplegada la arquitectura del proyecto TwizyLine, conceptos básicos sobre cifrado, autenticación, protocolos de intercambio de claves, ciberdelincuencia y legislación y un estudio sobre la ciberseguridad aplicada a aplicaciones móviles, servidores y vehículos.
- Capítulo 3: Análisis de fallas de seguridad. En este capítulo se analiza las posibles deficiencias de seguridad que se pueden encontrar en dos de los tres componentes principales que forman el proyecto de TwizyLine en base a los recursos de información que estos manejan. Estos dos componentes son el servidor y la aplicación móvil. Después, en base a este análisis, se proponen una serie de objetivos para mitigar estas faltas de seguridad.
- Capítulo 4: Implementación. En este capítulo se describen las medidas que se van a implementar dentro del proyecto TwizyLine y como estas han sido implementadas. Este capítulo se divide en dos secciones, una por cada componente a fortificar.
- Capítulo 5: Resultados. En este capítulo se expone una prueba de los resultados logrados mediante el uso de la herramienta Wireshark [10] para husmear las comunicaciones entre los componentes fortificados.
- Capítulo 6: Conclusiones. En último lugar, se presenta una exploración del proyecto explicando las soluciones obtenidas, tanto positivas como negativas. En base a estas conclusiones, se exponen una serie de objetivos futuros para la futura continuación de este proyecto.

2 Estado del arte

En el apartado anterior se ha visto que el objetivo de este trabajo de fin de grado es analizar el proyecto TwizyLine en busca de fallas de seguridad y actuar en consecuencia. Una visión general de la arquitectura del proyecto puede ser observada en la Figura 4. Como se puede ver, el sistema está compuesto a nivel de comunicación por tres elementos distintos: por un lado el vehículo conectado, por otro los smartphones de los clientes y finalmente el servidor, que se comunicará con los otros dos extremos utilizando el protocolo MQTT. En la siguiente sección se profundiza más en esta estructura, lo que será necesario para determinar cuáles son los elementos de seguridad que es necesario estudiar a nivel de estado del arte. Posteriormente se presentarán los fallos más comunes que existen en cada uno de los componentes identificados y cuáles son las técnicas que se pueden utilizar para su fortificación.

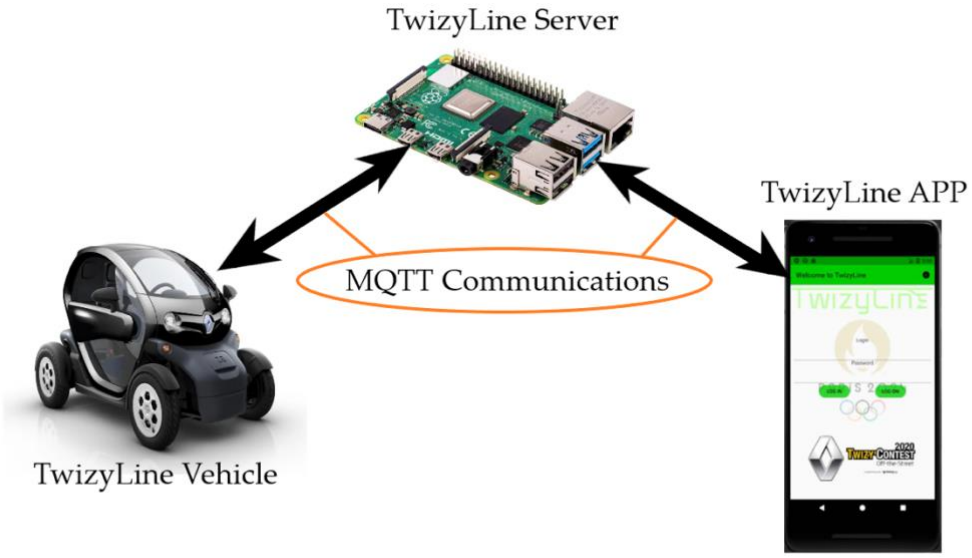

Figura 4. Visión general de la arquitectura del proyecto TwizyLine.

2.1 Proyecto TwizyLine

Tal y como se puede observar en la Figura 4, la arquitectura del proyecto TwizyLine está compuesta por tres componentes. Cada uno de estos componentes ofrece una serie de funcionalidades que hacen funcionar al sistema completo del proyecto. Para entender mejor el alcance de este proyecto es necesario exponer las principales funcionalidades de estos tres componentes:

- Vehículo: Sobre cada uno de los vehículos del proyecto está implementado un sistema acoplable que ofrece las capacidades necesarias para un aparcamiento del vehículo automatizado. Este sistema está formado, entre otros, por un módulo de comunicación y un módulo de control, los cuales están conectados entre sí por el mismo bus. El primero de estos módulos es el encargado de comunicarse con el servidor de TwizyLine. En esta comunicación se envía información del vehículo y las órdenes necesarias para el aparcamiento automatizado de este. Para esta comunicación se emplea

una tarjeta SIM que ofrece conexión 4G y la información es enviada empleando el protocolo MQTT, protocolo del que se hablará con más detalle en la próxima sección. El otro módulo es el encargado de supervisar todas las tareas relacionadas con el control del vehículo. Estos módulos están conectados entre sí y con el resto del vehículo a través del conocido y muy utilizado en la industria de la automoción bus CAN (*Controller Area Network*) [11]. Este segundo módulo también es el encargado de manejar los diferentes actuadores del vehículo para su posterior control automatizado. La Figura 5 muestra un resumen de las tareas realizadas por los módulos y su interconexión.

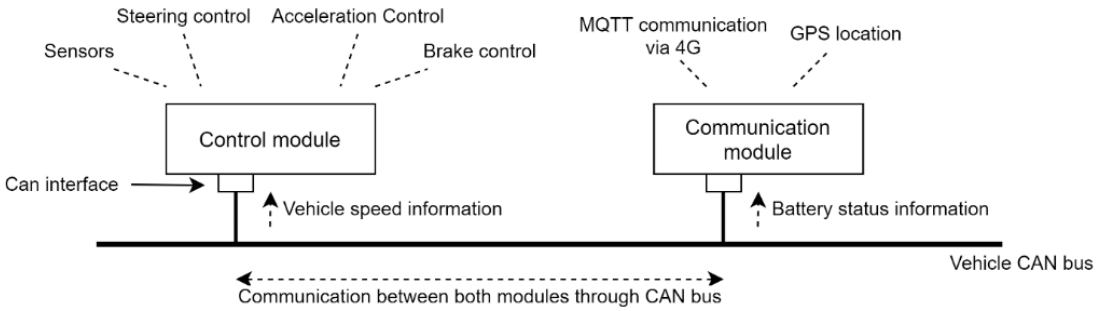

Figura 5. Tareas e interconexión de los módulos [12].

- Servidor: En el servidor (una Raspberry Pi 3 modelo B+) está implementado un *software back-end* sobre el sistema operativo Raspbian (Distribución de Linux). Este *software* permite controlar el sistema TwizyLine y es capaz de comunicarse con la aplicación móvil y con los vehículos a través Internet empleando para ello el protocolo MQTT. Para la implementación de este *back-end* se ha empleado: el gestor de bases de datos MariaDB [13] dónde es almacenada toda la información necesaria para que el sistema funcione adecuadamente, la librería de Python Paho [14] para implementar el protocolo MQTT y el lenguaje de programación Python para desarrollar el código encargado de controlar las comunicaciones entre la aplicación móvil y el servidor y las rutinas de llamadas a la base de datos MariaDB.
- Aplicación móvil: Esta aplicación móvil es una aplicación multiplataforma pero solo ha sido testada en sistemas operativos Android. Esta app proporciona a los usuarios el servicio de *carsharing*, dejando a los usuarios localizar el parking más próximo para que estos puedan adquirir un vehículo de este. Esta aplicación fue realizada empleando el IDE Android Studio [15], el *framework* Flutter [16] y el lenguaje de programación Dart [17]. Esta app se comunica con el servidor empleando el protocolo MQTT a través de la red inalámbrica del usuario. En esta aplicación móvil son guardadas las credenciales del usuario que permiten iniciar la aplicación de manera automática.

Aunque no se ha incluido como componente del sistema, es evidente que hay un cuarto actor del que no nos podemos olvidar: el usuario. El usuario será crítico en la cuestión de la seguridad, dado que, entre otras cuestiones, el sistema TwizyLine mantendrá una gran cantidad de información sensible del mismo.

En las secciones siguientes se expondrá el contexto de seguridad de los anteriores componentes, pero antes, debido a la importación central que tiene en el sistema,

será necesario introducir el protocolo de comunicación que se emplea en este sistema (MQTT). Posteriormente se abordará el concepto moderno de securización de las comunicaciones y de la autenticación de usuarios y componentes de la comunicación. Una vez terminada esta parte, se abordará la cuestión de los datos de los usuarios y cómo la ley exige recogerlos y asegurarlos, y finalmente se terminará con una revisión del estado del arte de la seguridad en los sistemas que componen la arquitectura de TwizyLine: los dispositivos móviles, los servidores y los vehículos.

2.2 Protocolo MQTT

Según se explica en la documentación que ofrece la organización de estandarización OASIS, "MQTT es un protocolo de transporte de mensajería de publicación/suscripción cliente-servidor. Es ligero, abierto, sencillo y está diseñado para ser fácil de implementar. Estas características lo hacen ideal para su uso en muchas situaciones, incluyendo entornos restringidos como la comunicación en contextos de Máquina a Máquina (M2M) e Internet de las Cosas (IoT), donde se requiere el desarrollo de un código que consuma los recursos mínimos y/o el ancho de banda de la red es muy importante" [18]. Esta definición nos explica las ventajas que nos ofrece este protocolo, que se resumen en que es un protocolo ligero, fácil de implementar y de desarrollar y que se ajusta a los dispositivos con recursos limitados como los dispositivos IoT. Para explicar el funcionamiento de este protocolo, se va a hacer uso de la documentación que ofrece 'La guía definitiva de MQTT para principiantes y expertos' escrito por el equipo de HiveMQ [19].

2.2.1 Arquitectura

La arquitectura en la que se basa MQTT ofrece una alternativa a la tradicional arquitectura de cliente-servidor. Esta alternativa, tal y como se puede ver en la Figura 6, es el modelo publicador-suscriptor, y está basada en publicaciones y suscripciones sobre tópicos. Este modelo desvincula al cliente que envía un mensaje, el publicador, del cliente o clientes que reciben los mensajes, el/los suscriptores. Esto hace que los publicadores y los suscriptores no sepan de la existencia del otro. La conexión entre un publicador y un suscriptor es gestionada por el tercer componente de la arquitectura, el bróker. Su trabajo es filtrar todos los mensajes entrantes y distribuirlos a los suscriptores.

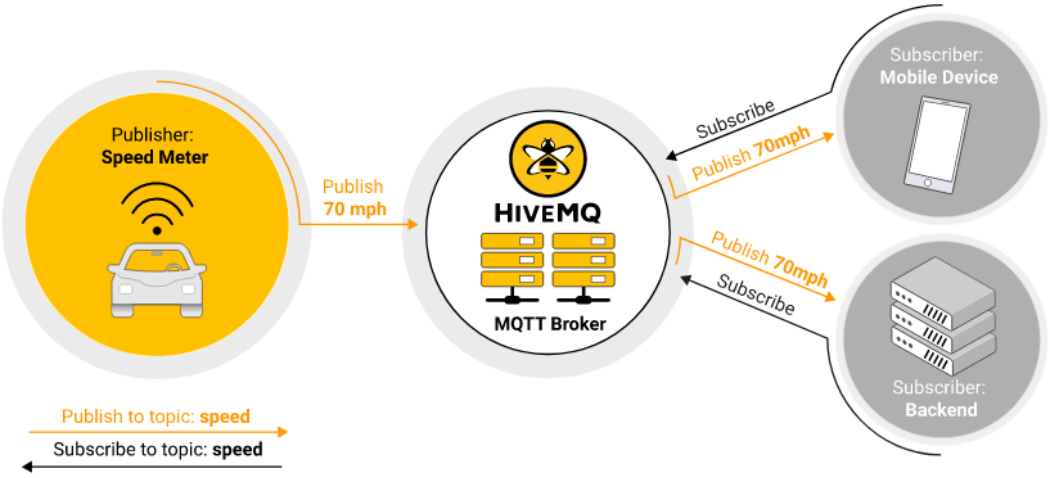

Figura 6. Arquitectura Publicador-Suscriptor MQTT [20].

En este tipo de arquitectura publicador-suscriptor, el cliente MQTT podrá ser tanto el publicador, como el suscriptor de esta arquitectura. Este cliente MQTT podría ser cualquier dispositivo que ejecuta una biblioteca MQTT y se conecta a un bróker MQTT a través de una red. Es decir, el cliente MQTT puede ser cualquier dispositivo que hable MQTT sobre una pila TCP/IP (*Transport Control Protocol/Internet Protocol*). Estas bibliotecas están disponibles para multitud de lenguajes como C, C++, Java, iOS, Android, Python, etc.

Por otro lado, el bróker es el responsable de recibir todos los mensajes de los publicadores y enviárselos a los suscriptores correspondientes. También será el encargado de la autenticación y autorización de los clientes, por lo que tendrá que saber a quién enviar qué tipo de mensaje.

Para poder enviar o recibir los mensajes, primero es necesario establecer una conexión entre el cliente MQTT y el bróker. Esta conexión será siempre iniciada por el cliente MQTT, por lo que, para iniciar una conexión, el cliente enviará un mensaje "CONNECT" al *bróker,* y si este mensaje está bien formado, este le responderá con un mensaje "CONNACK", que indicará que el intento de conexión fue exitoso y se dará por establecida la conexión. Esta conexión se mantendrá abierta hasta que el cliente envíe un comando de desconexión o la conexión se rompa.

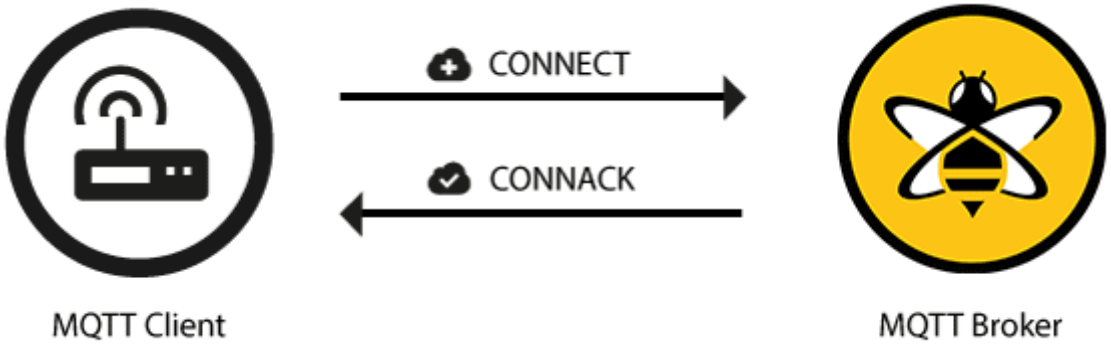

Figura 7. Establecimiento de una conexión MQTT entre un cliente y el bróker [21].

Un mensaje "CONNECT" bien formado, contendrá principalmente los siguientes campos:

- ClientID: Es único e identifica a cada cliente MQTT y su estado actual frente al bróker. Deberá de ir siempre relleno, a no ser que no se necesite que el bróker disponga del estado. En este caso, el siguiente campo debería de estar a *true*, o si no, el bróker rechazará la conexión.
- Clean Session: Este campo se usa para realizar una sesión persistente o no con el bróker. En una sesión persistente, el bróker almacenaría los mensajes perdidos por el cliente. Si no es persistente, el bróker eliminaría toda información de sesiones anteriores.
- Username/Password (Opcional): Permite autenticar a un usuario. Esta información se envía en texto plano, salvo que se use la implementación con TLS.
- KeepAlive: Es un intervalo de tiempo que define el mayor periodo de tiempo que el bróker y el cliente pueden soportar sin enviar un mensaje. Si no se

envían ningún mensaje, es responsabilidad del cliente enviar un paquete de tipo “PINGREQ” para mantener la conexión.

Por otro lado, un mensaje “CONNACK” enviará un código que informará al usuario si la conexión se ha establecido con éxito. En caso negativo, se le informará del motivo.

Una vez realizada la conexión, el cliente podrá publicar mensajes en un tópico o suscribirse a los mensajes que se publican en un tópico.

Para que un cliente pueda publicar en un tópico, este tiene que especificar:

- El tópico en el que publicar.
- La calidad de servicio (QoS) que desea, la cual determina qué tipo de garantía tiene un mensaje para llegar al destinatario, y va desde el 0 hasta el 2 (estos tipos se explicarán con más detalle en la sección siguiente).
- La dirección IP del bróker al que conectarse.
- El puerto por el que conectarse al bróker.
- El mensaje que desea publicar. Este mensaje puede ser una imagen, texto o cualquier dato binario.

Para que un cliente pueda suscribirse a un tópico, deberá de especificar los mismos puntos que los anteriores menos el último punto, ya que un suscriptor no envía mensajes, sino que los recibe.

2.2.2 Tópicos, filtrado de mensajes y QoS

La disociación que existe entre el publicador y el suscriptor permite que no necesiten conocerse entre ellos, por lo que no hay intercambio de direcciones IP y no necesitan funcionar al mismo tiempo ni interrumpir las operaciones de ambos componentes durante la publicación o la recepción de mensajes. Lo único que necesitan conocer los clientes es la dirección IP del dispositivo en el que se alojará el bróker, y un tópico en el que publicar o al que suscribirse para que pueda existir una comunicación entre el cliente y el bróker.

Este tópico consiste en una cadena de caracteres UTF-8, que es un formato de codificación de caracteres universal. Como se puede ver en la Figura 8, este tópico puede estar compuesto de uno o más niveles de tópicos. Estos niveles están separados por una barra (/).

Figura 8. Definición de un tópico con tres niveles.

Un cliente puede suscribirse a un tópico en particular, o a varios tópicos simultáneamente. Se podría suscribir a varios tópicos de un mismo nivel empleando el carácter '+', por lo que, si un cliente se quiere suscribir a los mensajes de todas las temperaturas de "micasa", se debería de suscribir de la manera "`micasa/+/temperatura`".

Si, por ejemplo, el cliente quisiera suscribirse a todos los tópicos que hay por debajo de un nivel, debería de emplear el carácter '#', por lo que, si un cliente quiere suscribirse a todos los tópicos de "micasa", se debería de suscribir de la manera "`micasa/#`".

Como se ha comentado, es el bróker el encargado de enviar a los suscriptores los mensajes en función del tópico o tópicos a los que se han suscrito. Por lo tanto, es el encargado de filtrar todos los mensajes que le llegan de cada publicador y enviárselo a su correspondiente suscriptor. Aparte de filtrar de la manera descrita hasta ahora, la cual filtra en función del tópico, existen otras dos maneras de filtrado:

1. Filtro basado en el contenido. El bróker filtra basándose en el tipo de contenido del mensaje, por lo que los clientes se suscriben a las peticiones de filtrado que les interesan. El inconveniente de este tipo de filtrado es que el contenido del mensaje debe conocerse de antemano y no puede cifrarse.
2. Filtrando por tipo/evento. Este filtrado resulta interesante cuando se utilizan lenguajes orientados a objetos, por lo que un cliente suscriptor podría suscribirse a todos los mensajes que contenga un tipo Excepción.

Para que este filtrado de mensajes funcione correctamente, es importante conocer de antemano la jerarquía de los tópicos. Ahora bien, esto no asegura que cuando un cliente publicador publica en un tópico, exista un cliente suscriptor que le esté escuchando en ese mismo momento. Por ello, si se desea, el bróker puede almacenar los mensajes para los clientes suscriptores que no están en línea.

Para poder controlar cómo se tratan los mensajes enviados, es necesario definir un nivel de QoS. Esta calidad de servicio se puede establecer en dos direcciones diferentes: la primera sería desde el cliente publicador hacia el bróker, y la segunda, sería desde el bróker al cliente suscriptor. Esto es porque cuando un cliente publica un mensaje, lo publica con un nivel de QoS, pero cuando un cliente se suscribe a un tópico, también se suscribe con un nivel de QoS. Estos niveles de calidad de servicio pueden ser diferentes. Los tres niveles de calidad de servicio que ofrece MQTT son los siguientes:

- Nivel 0 – Como máximo una vez: Este nivel de calidad garantiza una entrega *best-effort*. Este tipo de QoS indica que no garantiza la entrega del mensaje. El destinatario, ya bien sea el bróker o el cliente suscriptor, no acusa recibo del mensaje y el mensaje no es reenviado por el emisor.
- Nivel 1 – Al menos una vez: Este nivel garantiza que el mensaje se entregue al menos una vez a su correspondiente receptor. Esto es debido a que el emisor almacena el mensaje hasta que reciba un paquete del receptor que acusa recibo del mensaje. Si no recibe este paquete, este volverá a enviar el mensaje de nuevo hasta que recibe dicho paquete.

- Nivel 2 – Exactamente una vez: En este nivel, cada mensaje será recibido sólo una vez por sus destinatarios. Esto es posible gracias a los nuevos paquetes que surgen de solicitud/respuesta entre el emisor y el receptor. Tal y como se puede apreciar en la Figura 9, los nuevos paquetes que se envían son para confirmar que al emisor le ha llegado el acuse de recibo, anteriormente comentado, y el segundo paquete, confirma el anterior paquete.

Figura 9. Calidad de Servicio nivel 2 - Exactamente una vez.

2.2.3 Ventajas y desventajas

Las ventajas que ofrece MQTT son:

- Es muy sencillo de implementar, ya que no consume muchos recursos y basta con tener acceso a una de sus librerías, elegir un tópico al que publicar o suscribirse y empezar a funcionar.
- Escalabilidad: Las operaciones en el bróker se pueden paralelizar y los mensajes pueden ser procesados de una manera dirigida por eventos. En el caso de existir millones de conexiones, se pueden crear nodos bróker agrupados para distribuir la carga entre más servidores empleando equilibradores de carga.
- Ofrece una QoS ajustable a la fiabilidad de la red del usuario y a la lógica de su aplicación, ya que MQTT gestiona la retransmisión de mensajes y garantiza la entrega.
- Ofrece flexibilidad a la hora de crear soluciones debido a que la arquitectura publicador-suscriptor permite que muchos clientes produzcan y compartan información entre sí, los bróker pueden encadenarse y las colas de espera pueden hacer frente a las interrupciones de la conexión y proporcionar un búfer donde almacenar los mensajes.

Las desventajas que ofrece MQTT son:

- El establecimiento de conexión es muy costoso, por lo que, si solo se va a establecer la conexión para un solo mensaje, es muy ineficiente.
- Es un protocolo enfocado al desarrollo de proyectos IoT, por lo que si se quiere desarrollar un proyecto donde los requisitos de enrutamiento y direccionamiento son más complejos, se debería de buscar una mejor solución, como el uso del protocolo HTTP.

2.2.4 Aplicaciones del protocolo MQTT

Tras ver las ventajas y desventajas del protocolo MQTT, es importante conocer qué aplicaciones hacen uso de este protocolo y qué posibilidades tiene este protocolo para ofrecer seguridad a las comunicaciones. Esto ayudará a entender por qué se usó este protocolo como protocolo de comunicación para el proyecto TwizyLine.

MQTT es un protocolo de comunicación, y como tal, se puede emplear en el desarrollo de una aplicación como parte de la arquitectura de comunicaciones. En la sección anterior, se han comentado las ventajas que ofrece el uso del protocolo MQTT como protocolo para aplicaciones que no consumen muchos recursos y requieren de cierta flexibilidad. Estos requisitos son requisitos de la tecnología IoT.

La tecnología IoT tiene diversos usos, pero uno de los más habituales es su implantación en las llamadas "casas inteligentes", este tipo de edificios integran diferentes tecnologías para controlar los sistemas de seguridad, sistemas energéticos o comunicaciones para ofrecer eficiencia y confort a sus habitantes. De la mano del desarrollo de estas tecnologías, se desarrollan en paralelo aplicaciones móviles para que los habitantes de estas casas puedan controlar y visualizar los parámetros de estos sistemas [22]. Un protocolo que permite la interacción de dispositivos e instalaciones inteligentes en un hogar inteligente es el protocolo MQTT. Este protocolo forma parte de la tecnología que permite la interacción entre usuarios y dispositivos en el hogar inteligente [23].

Este ejemplo de casa inteligente se puede extrapolar al ejemplo de una *Smart* City o ciudad inteligente. Las aplicaciones de una ciudad inteligente, como el transporte, la sanidad, los edificios y los contadores inteligentes requieren el uso de protocolos e infraestructuras de telecomunicaciones estándar. IoT introduce la infraestructura necesaria y MQTT el protocolo de comunicación que permite la conexión entre elementos [24].

Pero la tecnología IoT no solo puede ser implementada en casas inteligentes, sino que se está buscando implementar esta tecnología en otros ámbitos. Por ejemplo, empresas petroleras y petroquímicas esperan utilizar esta tecnología para realizar la gestión remota de los equipos de las terminales de los campos petrolíferos para así optimizar la eficiencia y la seguridad, así como la escalabilidad del almacenamiento [25].

Tradicionalmente estas empresas empleaban redes inalámbricas o cableadas para transmitir información necesaria a los correspondientes centros de datos y bases de datos locales para su funcionamiento. Con la creciente demanda de consumo de datos en tiempo real por parte de las empresas, surgieron problemas como que el rendimiento esperado por los clientes no era el apropiado, los datos no estaban disponibles en tiempo real y el personal técnico de mantenimiento tenía una alta intensidad de trabajo y altos costes laborales.

Como el protocolo MQTT tiene las características de ligereza y alta fiabilidad QoS, se puede informar de un gran número de datos de equipos y sistemas en el área de producción al centro de datos de planta en tiempo real, lo que mejora en gran medida los problemas anteriormente comentados. Con la implementación de tecnología IoT

de la mano de MQTT se pueden reducir los costes generales de *hardware* y *software* en más del 50%, y los costes de personal al 70%.

Se ha visto que la tecnología IoT conecta dispositivos empleando el protocolo MQTT, para hacer más eficientes los procesos, proporcionar más comodidad y mejorar el trabajo y la vida personal. Sin embargo, se pueden llegar a exponer datos sensibles en las comunicaciones resultantes. Para proteger estos datos sensibles, MQTT ofrece las siguientes funcionalidades:

- Autenticación [26]: MQTT ofrece los campos de ClientID, usuario y contraseña en el mensaje "CONNECT" (visto en la sección Arquitectura) para poder autenticar al cliente cuando inicia su conexión con el bróker. Si esto no es suficiente, se puede añadir como método de autenticación un certificado del cliente X.509 (Estándar que define el formato de los certificados de infraestructura de clave pública [27]) El cliente presenta este certificado al bróker durante el *handshake* TLS (Más información sobre el protocolo TLS en la sección 2.3.2). Esto permite al bróker leer toda la información del certificado y utilizarla también para la autenticación.
- Autorización [28]: Para restringir que un cliente publique o se suscriba solo a temas autorizados, deben implementarse permisos en el lado del bróker. Estos permisos son configurables y una buena práctica es incluir el ID del cliente en el permiso. Estos permisos pueden ir en función del tema, operación permitida (publicar y/o suscribir) y el nivel de calidad de servicio permitido.
- Implementación de TLS [29]: MQTT se basa en el protocolo de transporte TCP. Por defecto, las conexiones TCP no utilizan una comunicación cifrada. Por ello, MQTT permite el uso de TLS en lugar de TCP simple. Para ello, se usará el puerto 8883. TLS proporciona un canal de comunicación seguro entre un cliente y un servidor.

2.3 ¿Qué es la seguridad?

Ahora que los conceptos básicos y necesarios para entender MQTT están expuestos, es necesario hablar sobre la seguridad y qué se entiende por seguridad dentro de la información y de la informática.

Para poder entender cómo de importante es la seguridad, hay que distinguir dos definiciones, seguridad de la información y seguridad informática:

La seguridad de la información es "aquel conjunto de medidas y técnicas empleadas para controlar y salvaguardar todos los datos que se manejan dentro de un sistema informático, y que aseguran que estos datos no salgan del sistema establecido" [30].

La seguridad informática es "una disciplina que se encarga de proteger la integridad y la privacidad de la información almacenada en un sistema informático" [31], que podría ser tanto un ordenador que aloje a un servidor, como un teléfono móvil.

El fin de este proyecto será el de aplicar ambas definiciones de seguridad al proyecto TwizyLine, ya que se quiere proteger los datos almacenados del usuario, que correspondería con la definición de seguridad informática, y los datos en tránsito

(datos que viajan a través de la red) del usuario, que corresponderían con la definición de seguridad de la información.

Para proteger cualquier tipo de datos, hay que tener en cuenta tres aspectos fundamentales: la confidencialidad, la disponibilidad y la integridad. Estos aspectos fundamentales son los objetivos de un Sistema de Gestión de la Seguridad de la Información (SGSI), cuyos requisitos están definidos en la Norma ISO 27001, que es un estándar para la seguridad de la información (*Information technology - Security techniques - Information security management systems - Requirements*) [32].

Confidencialidad**:** Este aspecto se asocia con la privacidad y el cifrado de la información, por lo que el objetivo es proteger la información del acceso no autorizado. En este sentido, los datos estarán disponibles sólo para los usuarios autorizados. Para poder llegar a ofrecer confidencialidad, la información será cifrada y sólo aquellos usuarios que hayan logrado pasar el control de acceso, mediante unas credenciales que les autentiquen, tendrán acceso a los datos descifrados [30]. Un buen ejemplo para usar como método de autenticación es emplear el sistema de triple factor (Ver Figura 10). Este esquema de autenticación consiste en dar algo que sabemos, como la contraseña, algo que poseemos, como un *token* recibido a través del teléfono móvil, y algo que somos, como la huella dactilar o nuestro propio rostro [33].

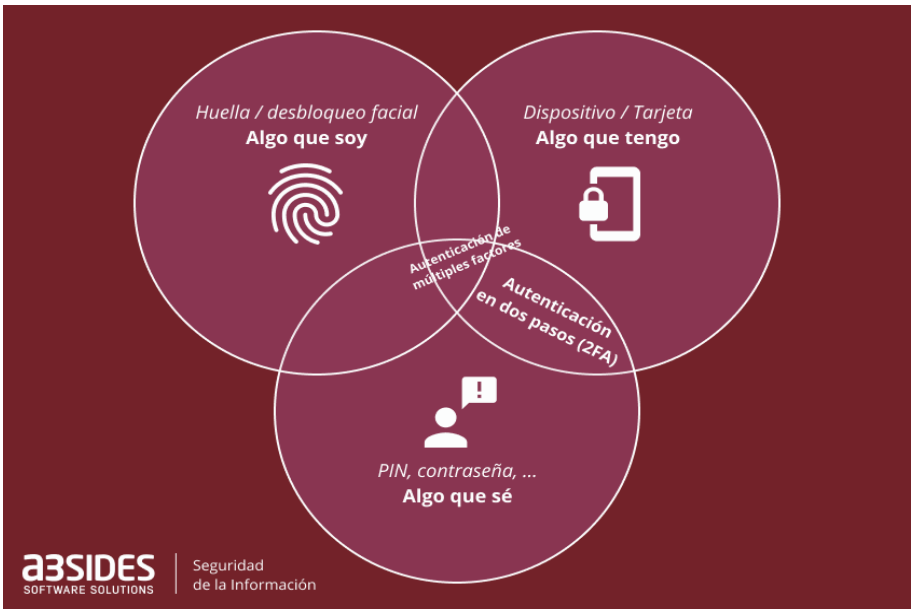

Figura 10. Autenticación en 3 pasos [34].

Disponibilidad**:** Una vez que el usuario ha pasado el control de acceso y ha sido identificado correctamente, este tiene que disponer de la información necesaria cuando la requiera. Esto hace que toda la infraestructura de soporte deba contar con mecanismo de protección y defensa ante amenazas y riesgos. Estas amenazas y riesgos no solo incluyen los posibles ciberataques, sino que pueden incluir problemas físicos como apagones o incendios. Para poder prevenir estas amenazas y cumplir con este pilar se deberán implementar medidas de soporte, tales como tener la información duplicada y protegida, y medidas que ofrezcan estabilidad en la red, como mantener los equipos constantemente actualizados y sin interrupciones [35].

Integridad**:** El usuario ya ha sido identificado en el sistema y acaba de acceder a su información. Ahora es necesario que esa información, sea realmente la información que ha solicitado el usuario. Si existe una alteración de esta información, significa que ha habido una pérdida de integridad. Este aspecto garantiza la veracidad de la información indicando que los datos no pueden alterarse sin autorización previa. Como medidas para proteger la integridad de los datos, es necesario certificar que el origen y destino de los datos coinciden, mediante el uso de protocolos de transporte seguros, y utilizar algoritmos MAC (*Message Authentication Code*), algoritmo que emplea una parte del mensaje original para autenticar el resto del mensaje, para identificar cambios en los mensajes transmitidos [36].

Figura 11. Los 3 pilares de la seguridad de la información: Confidencialidad, Disponibilidad e integridad [37].

Estos son los tres pilares que sustentan la seguridad de la información y de la informática. En estos tres aspectos se basarán las futuras decisiones para fortificar las comunicaciones MQTT entre vehículo-servidor-aplicación, el servidor donde se aloja el bróker MQTT y cualquier tipo de dato que se considere sensible.

2.3.1 Cifrado y autenticación

Para cumplir con los pilares de la confidencialidad y la integridad, será necesario aplicar herramientas de cifrado, de autenticación y protocolos de seguridad que velen por estos dos pilares. Esta sección se apoya en el libro 'Cryptography Engineering: Design Principles and Practical Applications' [38], donde se pueden encontrar más detalles sobre la cuestión.

Para hablar de cifrado, hay que hablar antes de qué es la criptografía. La criptografía es el arte y la ciencia del cifrado. Hoy en día es mucho más amplia y abarca la autenticación, las firmas digitales y muchas otras funciones de seguridad elementales. La criptografía estudiada en profundidad requiere un amplio conocimiento matemático, y cada algoritmo desarrollado se suele basar en complejas operaciones y propiedades matemáticas. Sin embargo, en esta sección solo se pretende dar conceptos generales de la criptografía aplicada, no se hará un estudio profundo de la misma y de las distintas técnicas existentes.

El cifrado es el objetivo original de la criptografía. Para explicar el cifrado, partiremos del escenario mostrado en la Figura 12, en la que Alice quiere enviar el mensaje "m" a Bob y Eve se encuentra espiando esta comunicación.

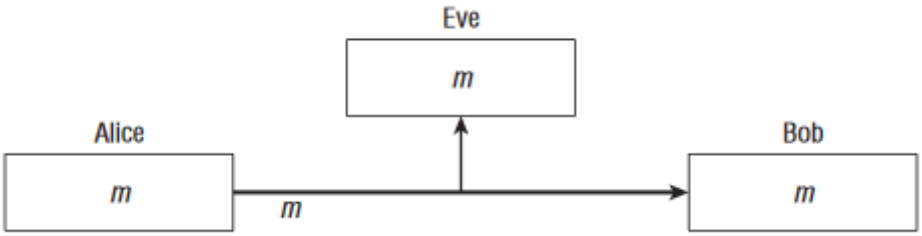

Figura 12. Escenario de comunicación entre Alice y Bob donde Eve se encuentra espiando la comunicación [38].

Para evitar que Eve entienda la conversación que mantienen Alice y Bob, estos deberán encriptar los mensajes que se encuentran en texto plano (así se conocen a los mensajes que se envían sin cifrar) y convertirlos en texto cifrado con una clave para que Eve no logre conocer el contenido de los mensajes. La clave para el cifrado y el descifrado es la misma, y es compartida por ambos extremos de la red. Una buena función de encriptación será aquella que hace imposible encontrar el texto plano a partir del texto cifrado sin conocer la clave. Como Bob conoce la clave con la que Alice ha encriptado el texto plano, este podrá descifrar el texto cifrado y conocer el mensaje enviado por Alice. A este método de encriptación donde se usa la misma clave para cifrar y descifrar se llama cifrado de clave simétrica o cifrado de clave secreta.

Para utilizar la encriptación, Alice y Bob deben compartir la clave que van a usar para cifrar sus mensajes. La pregunta es, ¿cómo comparten esta clave sin que Eve pueda capturar esta clave mientras está espiando? La respuesta a esto es usar el método de clave pública o clave asimétrica, donde la clave que se usa para cifrar es distinta a la que se usa para descifrar. Este método consiste en que Bob genera una clave pública para compartir con Alice y otra clave que se guarda para él mismo. Alice lo que hace es encriptar sus mensajes con la clave pública de Bob. Cuando le llega este mensaje cifrado a Bob, Bob usa la otra clave que había generado para descifrar el mensaje. Lo mismo ocurriría si Bob quiere enviar mensajes cifrados a Alice.

El método de encriptación de clave asimétrica parece más seguro que el método de clave simétrica. Esto es porque no compartes la clave con la que descifras el mensaje. El problema detrás del método de clave asimétrica es que depende en gran medida de la matemática, ya que un requisito obvio es que sea imposible calcular la clave que se usa para descifrar a partir de la clave que se usa para cifrar. Esta premisa hace que usar este método para todo sea demasiado costoso desde el punto de vista computacional. Por ello, en los sistemas reales se usa una mezcla de algoritmos de clave pública y clave privada. Los algoritmos de clave pública se utilizan para establecer una clave secreta. Esta clave se usará para cifrar y descifrar los mensajes intercambiados. Esto combina la flexibilidad de la criptografía de clave pública con la eficacia de la criptografía de clave simétrica.

No se entrará en más detalle con los diferentes algoritmos que existen para generar tanto la clave pública como la clave privada, pero es necesario exponer cómo funcionan estos métodos para entender cómo funcionan los protocolos que se implementan en las comunicaciones seguras.

El método de cifrado expuesto hasta el momento asegura la confidencialidad de la conversación. Sin embargo, no es capaz de asegurar la autenticidad de los extremos, por lo que es posible que un tercer actor se haga pasar por uno de los extremos, lo cual también comporta riesgos evidentes.

No hay que confundir los conceptos de encriptar y autenticar. Encriptar un mensaje no impide la manipulación de su contenido, y autenticar un mensaje no mantiene el mensaje en secreto. Uno de los errores clásicos en criptografía es pensar que encriptar un mensaje también impide que Eve pueda manipular este mensaje y esto no es así.

Por lo tanto, ahora que ya están expuestos los métodos que se usan para generar las claves con las que cifrar los mensajes, se va a explicar cómo Bob se puede asegurar de que el mensaje recibido es de Alice, y no un mensaje que ha enviado Eve en nombre de Alice. Este ejemplo puede verse en la Figura 13.

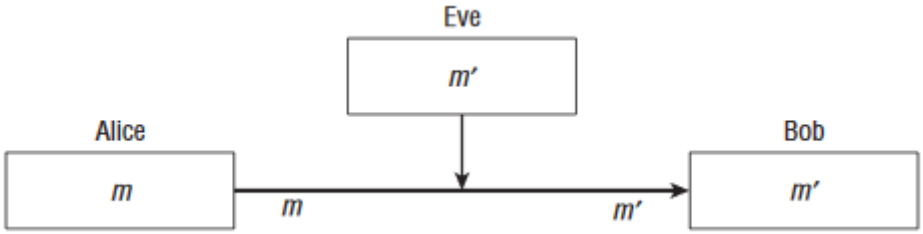

Figura 13. Eve intercepta y altera un mensaje que envía Alice y este se lo envía a Bob haciéndose pasar por Alice [38].

Para que Bob no dude del origen del mensaje, Alice emplea un código de autenticación de mensaje (MAC) que acompaña al mensaje original. Para generar la **MAC** se emplea una clave secreta (simétrica) conocida por Alice y Bob. Para calcular esta clave compartida, se emplea el método descrito anteriormente, dónde se usa el método de clave pública para generar la clave secreta. Aplicando la clave simétrica al mensaje original es como se genera esta MAC. Cuando el mensaje llegue a Bob, este comprobará que la MAC es correcta, y si no lo es, descartará el mensaje.

Se ha visto que la criptografía de clave pública simplifica la gestión de claves, pero Alice sigue teniendo que recibir la clave pública de Bob y autenticar que ha sido Bob quien se la ha mandado. La solución a esto es usar una **PKI** (*Public Key Infraestructure*), el cual es un sistema que permite reconocer qué clave pública pertenece a quién. El funcionamiento de una PKI es el siguiente: Existe una Autoridad Certificadora (CA) la cual tiene una clave pública conocida por todo el mundo (también por Bob) y todo el mundo conoce a la CA (es necesario que se confíe en la CA). Entonces Alice, que ha generado su clave pública, le dice a la CA: “Soy Alice y esta es mi clave pública”. La CA verifica la identidad de Alice y firma una declaración digital del estilo “Esta clave pública pertenece a Alice”. Esta declaración firmada se conoce como **certificado**. Entonces, ahora Alice puede enviar su clave pública y su certificado a Bob. Como la clave pública del CA es conocida por todos, Bob puede verificar la firma del certificado y confiar que la clave pública pertenece a Alice. Lo mismo pasaría al revés, Bob enviaría su clave pública y certificado a Alice.

Emplear una PKI para autenticar la clave pública con la que poder generar una clave secreta con la que encriptar y descifrar, y añadiendo una MAC al mensaje, permite

que Alice pueda enviar un mensaje con la total seguridad de que es prácticamente imposible que ese mensaje pueda ser entendido por terceras personas. Además, Bob puede tener la certeza de que ese mensaje recibido es de Alice.

2.3.2 Protocolo de intercambio de claves

Ahora que se tiene claro como poder enviar información de una forma segura y con autenticación, es necesario exponer aquellos protocolos que implementan las anteriores medidas. Estos protocolos se conocen como protocolos de intercambio de claves autenticadas. Este apartado es una descripción simplificada de todo el proceso a realizar, no pretende hacer una revisión exhaustiva, sino describir los elementos necesarios que se deben conocer para su posterior implementación. Una revisión más profunda de esta cuestión se puede encontrar por ejemplo en 'Protocols for Authentication and Key Establishment' [39], fuente que se ha utilizado para completar esta sección.

Entre los diferentes protocolos de intercambio de claves destacan:

- TLS: Conocido anteriormente como SSL (*Secure Sockets Layer*). Está desarrollado y mantenido por el grupo de trabajo TLS del IETF (*Internet Engineering Task Force*) y funciona sobre la pila de protocolos TCP/IP.
- SSH (*Secure SHell*): También funciona sobre TCP/IP y se utiliza principalmente para asegurar los inicios de sesión de líneas de comandos remotas, aunque también puede emplearse para la transferencia de archivos, así como para implementar una red privada virtual rudimentaria.
- IPsec (*Internet Protocol Security*): Este protocola en la capa IP y se aplica automáticamente a todos los datos de las aplicaciones que están por encima de ella. IPsec puede ejecutarse en diferentes arquitecturas como *host-to-host*, *host-to-gateway* y *gateway-to-gateway*.

En esta sección se estudiará más en profundidad el protocolo TLS debido a su uso generalizado y sus complejidades de diseño, dado que combina directamente el intercambio de claves y la encriptación autenticada para establecer un canal seguro. Además, es el único protocolo que soporta MQTT para implementar un canal seguro.

El protocolo TLS consta de varios subprotocolos: el protocolo *handshake*, el protocolo de capa de registro y el protocolo de alerta. A efectos criptográficos, los más importantes son los dos primeros, ya que el protocolo de alerta se utiliza para notificar a los pares sobre los errores o para cerrar la conexión.

2.3.2.1 Protocolo *Handshake*

Para establecer una conexión TLS, el cliente y el servidor deben proceder con el protocolo *handshake*. Durante este procedimiento, el cliente y el servidor se ponen de acuerdo en un conjunto de parámetros criptográficos, llamado *ciphersuite*, que incluye el algoritmo de intercambio de clave, algoritmos de cifrado, MAC y otros parámetros, intercambian credenciales de autenticación, establecen un secreto compartido, realizan una autenticación explícita y obtienen claves para el cifrado y la autenticación de mensajes. En el caso de que el cliente y el servidor han establecido previamente una sesión, pueden realizar un *handshake* abreviado para reanudar la sesión.

Los pasos a seguir se pueden ver en la Figura 14, y son los siguientes:

1. El cliente inicia la conexión enviando un mensaje *ClientHello*. El objetivo principal de este mensaje es transmitir las preferencias de los parámetros del cliente. En este mensaje se incluye una lista de algoritmos de cifrado y de intercambio de clave y compresión, en orden de preferencia, soportados por el cliente. Además, se especifica la versión de TLS más alta soportada y el ID de sesión y un *nonce*, que es una cadena de bytes aleatorios. Este *nonce* se usará para generar las claves de sesión.
2. El servidor responde al cliente con varios mensajes:
    a. *ServerHello*: Indica los parámetros elegidos para el cifrado y compresión y envía el *nonce* suyo y el del cliente.
    b. *Certificate*: Este mensaje es opcional, ya que, dependiendo del algoritmo de intercambio de clave escogido, es necesario o no el envío del certificado. Este mensaje contiene una cadena de certificados. Esta cadena contiene el certificado del servidor y el certificado de la CA.
    c. *ServerKeyExchange*: Este mensaje se envía solo si el cifrado escogido requiere que el servidor envíe una clave pública efímera.
    d. *CertificateRequest*: Este mensaje es opcional, ya que, dependiendo del algoritmo de intercambio de clave escogido, el servidor requerirá que el cliente se autentique con un certificado o no.
3. El cliente verifica el certificado del servidor y responde con los siguientes mensajes:
    a. *Certificate*: Igual que en el *ServerHello*.
    b. *ClientKeyExchange*: Este mensaje siempre es enviado por el cliente para establecer un secreto *premaster*. Este secreto se empleará para generar un secreto *master* compartido. La estructura del mensaje depende del algortimo de intercambio de clave escogido.
    c. *CertificateVerify*: Este mensaje se envía si el cliente ha enviado un certificado. El mensaje se calcula como la firma de todos los mensajes de enlace que ha enviado y recibido el cliente hasta este mensaje, pero sin incluirlo.
    d. En este punto, con los *nonce* del cliente y el servidor y el secreto *premaster* se obtiene el secreto *master* del que se obtendrán las claves de sesión: dos claves para el cifrado (cliente-servidor y servidor-cliente) y dos claves para la autenticación del mensaje (cliente-servidor y servidor-cliente).
4. El cliente envía un mensaje *ChangeCipherSpec*, que indica que todos los mensajes posteriores se enviarán cifrados utilizando el protocolo de capa de registro (Se verá más adelante) y usando las claves de sesión compartidas. Por último, el cliente envía (cifrado por la capa de registro) un mensaje *Finished*, que contiene un valor de confirmación de la clave.
5. El servidor obtiene las claves de sesión de la misma manera que el cliente. Después, verifica la autenticación del cliente, si la hay, y luego verifica el mensaje *Finished* que ha recibido del cliente.
6. El servidor envía un mensaje *ChangeCipherSpec*, que todos los mensajes posteriores se enviarán cifrados utilizando el protocolo de capa de registro.

7. Por último, el servidor envía (cifrado por la capa de registro) su propio mensaje *Finished* para la confirmación de la clave. A partir de este momento se puede comenzar a intercambiar información del nivel de aplicación, ya que esta irá ya cifrada con los parámetros negociados anteriormente.

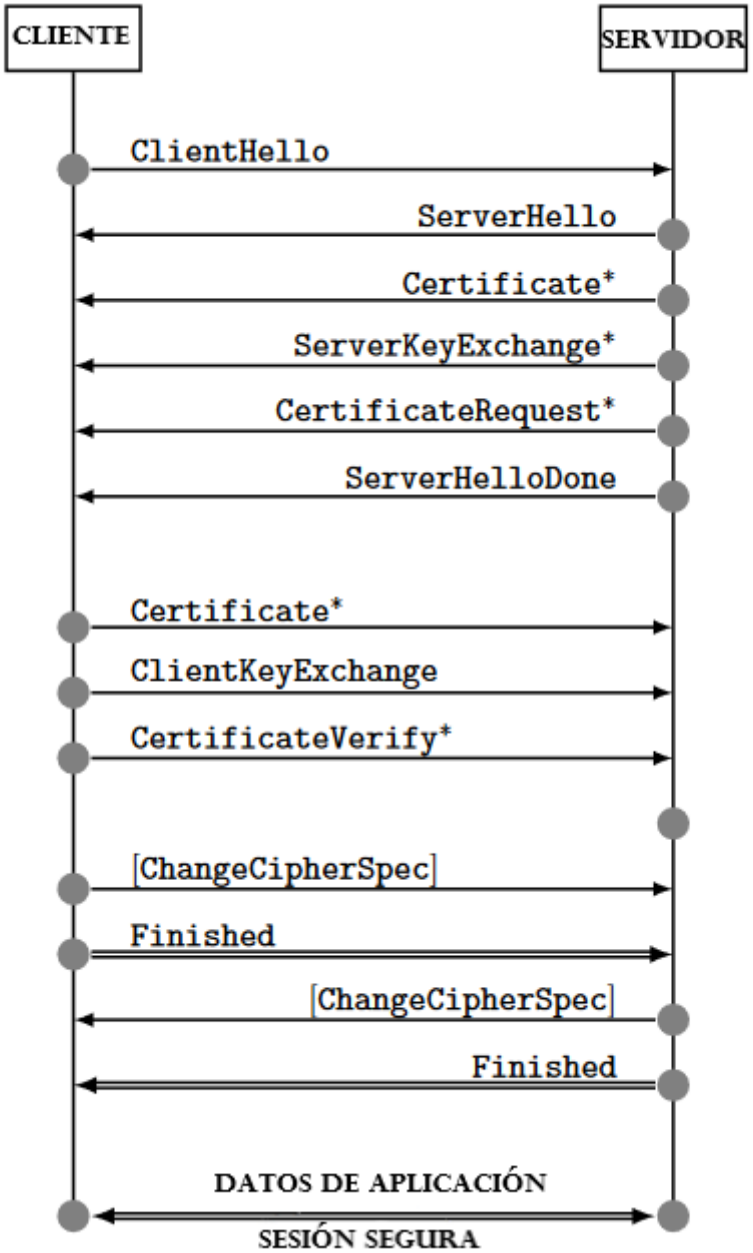

*Figura 14. Fases del protocolo Handshake [39]. * estos mensajes pueden no estar presentes en todos los cifrados. Las flechas singulares indican flujos de texto plano y los dobles flujos de texto cifrado.*

Estos son los pasos que se siguen durante el protocolo *handshake* por lo que a partir de este punto, los de la aplicación pueden ser enviados usando el protocolo de capa de registro, el cual se encarga principalmente de cifrar los datos de aplicación.

2.3.2.2 Protocolo de capa de registro

La funcionalidad del protocolo de la capa de registro proporciona la entrega de todos los mensajes de aplicación cifrados y/o comprimidos por los algoritmos negociados en el protocolo *handshake*, incluidos los mensajes del protocolo de enlace y los datos de la aplicación,

La carga útil de un mensaje enviado por la capa de registro contiene los siguientes campos:

- Un tipo de contenido que especifica qué tipo de datos contiene la carga útil; puede tratarse de un mensaje del protocolo *handshake*, un mensaje *ChangeCipherSpec*, un mensaje de alerta o datos de aplicación.
- La versión de TLS empleada.
- La longitud de la carga útil. Esta debe de ser múltiplo de la longitud del bloque del cifrado escogido. Si es necesario, los paquetes son fragmentados.
- Los datos de la carga útil, que pueden ser datos del protocolo *handshake*, un mensaje *ChangeCipherSpec*, un mensaje de alerta o datos de aplicación.

Una vez añadido estos campos a la carga útil, esta se comprime utilizando el algoritmo de compresión que se haya negociado durante el protocolo *handshake*. Y finalmente, la carga útil comprimida se encripta y se autentifica en función del algoritmo de cifrado utilizado. En la Figura 15 se puede ver una ilustración de los pasos que se realizan cuando llegan datos de aplicación para ser transmitidos.

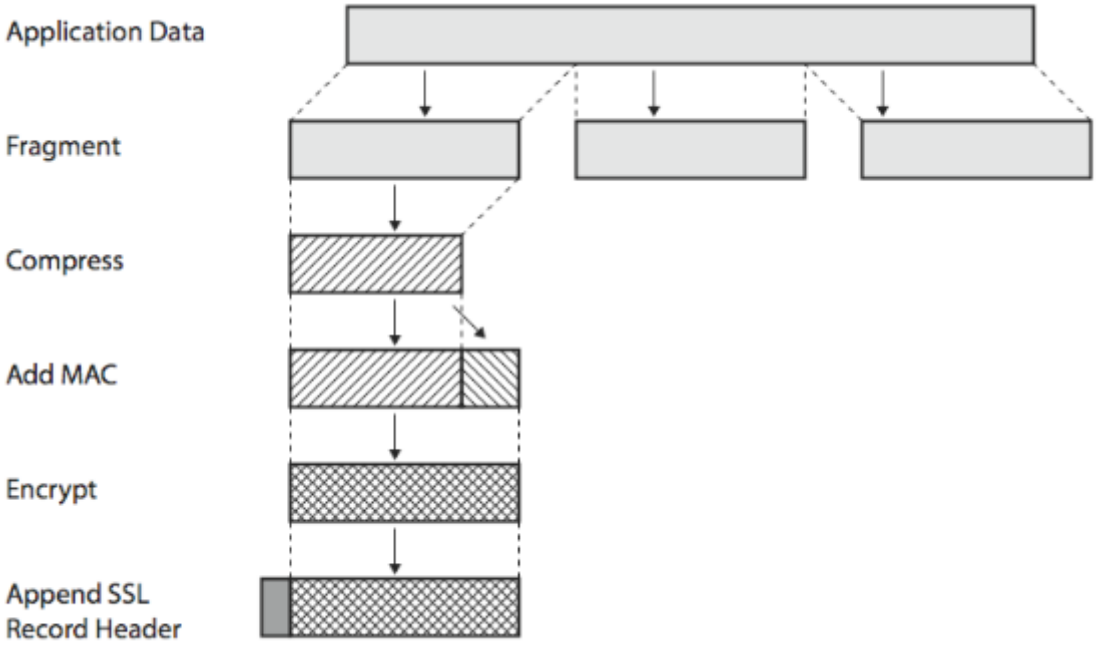

Figura 15. Operación del protocolo de la capa de registro [40].

2.3.3 Los ciberdelincuentes y los ciberataques

Los ciberdelincuentes son personas que realizan actividades delictivas en la red contra personas o sistemas informáticos [41]. Son una de las mayores amenazas para todas las empresas e internautas del mundo. Tal es la amenaza que en 2021 la ciberdelincuencia costó en torno a los 6 billones de dólares anuales, haciendo que el coste medio de la ciberdelincuencia para una organización sea de 13 millones de dólares [42]. Estos costes están aumentando en la medida de que los ciberataques también están evolucionando desde el punto de vista de sus objetivos, esto es, no solo utilizan técnicas más potentes, sino que buscan objetivos que les sean más rentables económicamente.

Por ejemplo, desde que empezó la pandemia, el marco laboral ha adoptado un modelo híbrido en el que conviven el trabajo presencial con el teletrabajo. Esta transformación digital ha hecho que los ciberdelincuentes también se hayan adaptado a este cambio con el fin de poder explotar las nuevas vulnerabilidades y oportunidades que ofrece [43].

En 2021, las organizaciones pertenecientes a EMEA (Europa, Oriente Medio y África), a EEUU y a APAC (Asia-Pacífico) experimentaron un incremento en ciberataques desde el principio del año hasta la primera mitad del 2021 del 36%, 17% y 13% , respectivamente. Entre las nuevas amenazas que sufrieron las diferentes organizaciones en la primera mitad de 2021 destacan [43]:

- Triple extorsión por *Ransomware*: Un *Ransomware* es un *malware* cuya finalidad es secuestrar un dispositivo de tal manera que, si el propietario de este dispositivo no paga el rescate, no podrá acceder al dispositivo [41]. La triple extorsión consiste en: primero los ciberdelincuentes roban datos de la organización y los cifran para que sean inaccesibles; segundo, amenazan con hacer esta información púbica si la empresa no paga el rescate; y tercero, envían peticiones de rescate de datos a los propios clientes amenazándoles con hacer pública la información de cada uno de ellos. El pago medio de los rescates por parte de las organizaciones ha ido creciendo hasta los trecientos treinta mil dólares.
- Ataques a la cadena de suministro digital: Una de las normas para estar protegido de los ciberataques es mantener el software actualizado. Es por esta clase de actualizaciones por donde pueden entrar a atacar los ciberdelincuentes. Lo que hacen los ciberdelincuentes es atacar a las empresas que ofrecen este suministro de actualizaciones para poder atacar a través de estos a empresas de mayor interés. Esto hizo que empresas que actualizaban su software mediante una herramienta de suministro digital, acabaran suministrando información sensible al atacante. Esto ha hecho que proveedores de este servicio, aunque sean de pequeña escala, se conviertan en objetivos lucrativos para los ciberdelincuentes. Y no es porque tengan información valiosa, sino porque pueden ser una puerta de entrada a empresas de mayor escala.
- Ataques vía teléfono móvil: Debido a las circunstancias del teletrabajo, los dispositivos móviles han cobrado protagonismo a medida que estos se están empezando a usar en mayor medida como parte del conjunto de herramientas profesionales. Como resultado, una mayor cantidad de información profesional se almacena o es accesible desde los teléfonos móviles, y por tanto, un vector de ataque típico es atraer a la víctima para que descargue una aplicación, aparentemente no fraudulenta, en la que el *malware* que permite acceder a todos los recursos del móvil esté incrustado en esta aplicación.

Los tipos de malware más empleados para realizar un ciberataque en la primera mitad de 2021 fueron los siguientes (Ver Figura 16):

- *Botnet:* Conjunto de ordenadores controlados remotamente por un atacante que pueden ser utilizados para realizar actividades maliciosas como envío de *spam*, ataque de denegación de servicio, etc. [41].
- *Banking*: Se trata de un *malware* que se instala en el dispositivo del usuario e intenta obtener acceso a la información confidencial que se almacena o procesa a través de los sistemas bancarios en línea [44].
- *Cryptominers*: Este tipo de *malware* emplea los recursos del dispositivo para "minar" criptomonedas. Estas criptomonedas son tipos de dinero digital creadas como alternativa al dinero tradicional [45]. Se conoce al acto de "minar" como el acto de utilizar los recursos informáticos para procesar transacciones y obtener recompensas en forma de criptomonedas [46].

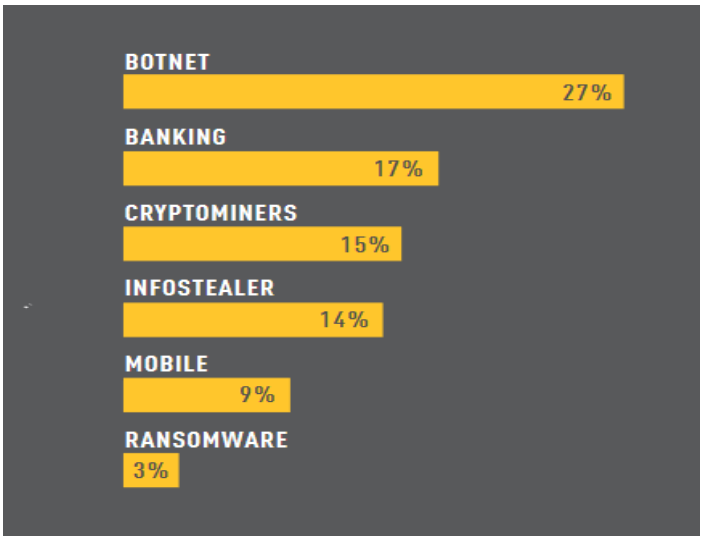

Figura 16. Porcentaje de redes corporativas atacadas por cada tipo de malware .

En la Figura 17 se puede observar el nivel de amenaza a ciberataques por países, el cual está basado en la probabilidad de que una máquina en un país determinado sea atacada por un malware.

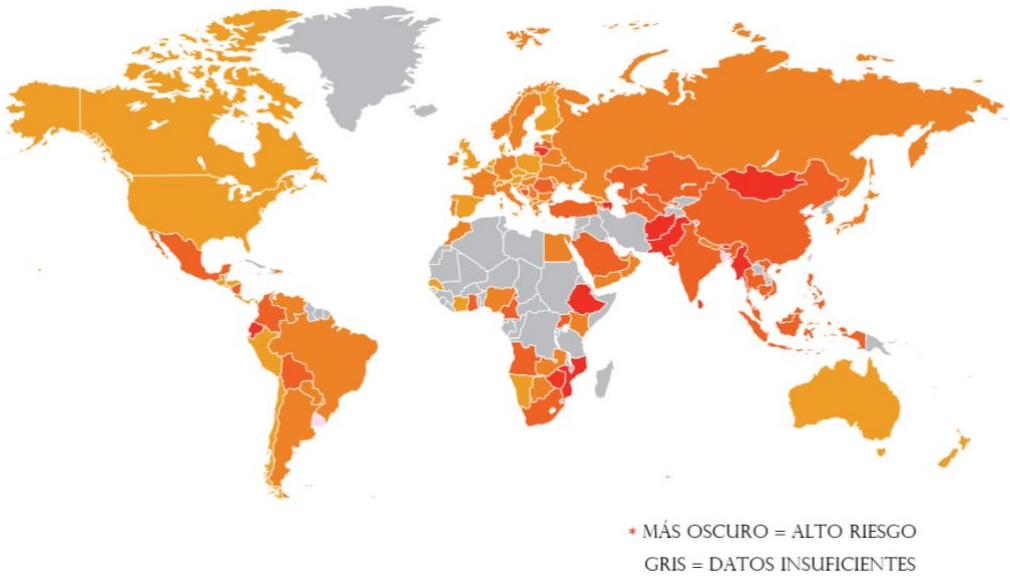

Figura 17. Mapa del índice de amenaza global [43].

Los cambios continuos del *malware* y de los ciberataques, el trabajo híbrido y los procesos empresariales digitales en la nube, han introducido nuevos riesgos para las empresas. Los consejos de administración de las empresas están empezando a incorporar expertos en ciberseguridad específicamente para examinar las cuestiones de seguridad y riesgo. Así podrán combatir las amenazas anteriormente comentadas. La empresa de auditoría Gartner recomienda entender las siguientes tendencias en ciberseguridad para capacitar a las empresas y para que sepan abordar los nuevos riesgos [47]:

- Ampliación de la superficie de ataque: El trabajo híbrido, el incremento del uso de la nube pública, las cadenas de suministro altamente conectadas y el uso de sistemas ciberfísicos (sistemas que integran la detección, la computación, el control y la conexión en red en objetos físicos e infraestructuras [48]) han expuesto nuevas superficies de ataque para los ciberdelincuentes, dejando a las organizaciones más vulnerables. Entender estas nuevas superficies permitirá a las empresas adaptarse a los futuros ataques.
- Defensa de los sistemas de identidad: El uso indebido de credenciales es ahora un método que los atacantes utilizan para acceder a los sistemas. Se recomienda el uso de sistemas de detección y respuesta a las amenazas de identidad para defender los sistemas de identidad.
- Riesgo en la cadena de suministro digital: Como se ha comentado anteriormente, se pueden sufrir ataques a través de la cadena de suministro digital, por lo que los líderes de seguridad y gestión de riesgos de las organizaciones deben presionar a los proveedores digitales para que demuestren las mejores prácticas de seguridad.
- Malla de ciberseguridad: En los últimos dos años, a causa de la pandemia, se ha acelerado la tendencia en la que los activos digitales se encuentran cada vez más fuera de la infraestructura empresarial tradicional. La malla de seguridad es un enfoque conceptual moderno de la arquitectura de seguridad que permite a la empresa desplegar e integrar la seguridad en los activos, ya estén en las instalaciones, en los centros de datos o en la nube. Esto podría reducir el impacto financiero de los incidentes de seguridad individuales en una media del 90% para 2024.
- Más allá de la conciencia: El error humano sigue estando presente en la mayoría de las violaciones de datos, lo que demuestra que los enfoques tradicionales de la formación en materia de seguridad son ineficaces. Las organizaciones progresistas están dejando atrás las anticuadas campañas de concienciación basadas en el cumplimiento y están invirtiendo en programas holísticos de cambio de comportamiento y cultura diseñados para provocar formas de trabajo más seguras.

Expuestos estos conceptos, los expertos en ciberseguridad de las diferentes organizaciones tienen que: repensar la pila tecnológica de seguridad para hacer frente a las nuevas amenazas, llevar la toma de decisiones de ciberseguridad a las unidades de negocio para mejorar su postura de seguridad y evolucionar y replantear la práctica de la seguridad para gestionar mejor los riesgos.

2.3.4 Leyes y reglamentos

Habiendo expuesto los principales conceptos de ciberseguridad, los diferentes protocolos de comunicación segura y los riesgos que pueden correr las organizaciones frente a la ciberdelincuencia, se va a exponer el marco legal sobre la protección de los datos de los usuarios, dentro del marco normativo de la Unión Europea.

Para que el proyecto TwizyLine funcione, se emplean datos de los usuarios, como el nombre y apellido de estos, por lo que es necesario conocer los derechos que tienen estos usuarios y entender el marco legal de estos datos.

Este marco tiene su pilar en el Reglamento General de Protección de Datos (RGPD) [49]. Este reglamento "establece las normas relativas a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos". Este reglamento no excluye el derecho de los Estados miembros de la Unión Europea a adaptar este reglamento y especificar sus normas, siempre y cuando se garantice que el tratamiento de los datos personales debe ser igual en todos los Estados miembros. En España, la adaptación de este reglamento es Ley Orgánica de Protección de Datos y Garantía de Derechos Digitales (LOPDGDD) [50].

La RGPD establece que todo individuo tiene derecho a la protección de los datos de carácter personal. Este derecho no es un derecho absoluto, sino que tiene que guardar relación con su función y mantener el equilibrio con otros derechos fundamentales. El tratamiento de estos datos está concebido para servir a la sociedad.

La creación de este marco legal viene motivada por: la evolución de la tecnología en la vida social, la globalización, el aumento del intercambio de los datos personales y la creación de un nuevo marco legal más sólido. Todo ello con el fin de fortalecer la seguridad práctica y jurídica para las personas físicas, los operadores económicos y las autoridades. Los principios que regentan este reglamento son los siguientes:

- Principio de licitud, transparencia y lealtad: Se hará un tratamiento lícito, leal y transparente de los datos personales por parte del interesado.
- Principio de finalidad: Todos los datos recogidos por parte del interesado serán tratados con uno o varios fines determinados, explícitos y legítimos y nunca tratados posteriormente con otro fin diferente.
- Principio de minimización de los datos: Se recogerán solo los datos precisos y necesarios según los fines especificados.
- Principio de exactitud: Esto implica que se debe garantizar la integridad y la confidencialidad de los datos recogidos por parte de los responsables.
- Principio de limitación del plazo de conservación: Los datos deben conservarse durante no más tiempo del necesario para los fines del tratamiento. Una vez finalizado este fin, deberán eliminarse estos datos recogidos.
- Principio de seguridad: Garantiza la confidencialidad, la disponibilidad y la integridad de los datos tratados.
- Principio de responsabilidad activa: Obliga a mantener la atención debida por parte de los responsables para proteger y garantizar los derechos y libertades

de los usuarios cuyos datos han sido tratados, así como demostrar que el tratamiento de estos datos se ajusta a la LOPDGDD y al RGPD.

Los principios de la protección de datos deben aplicarse a toda la información relativa a una persona física identificada o identificable. Estos datos personales incluyen:

- Datos personales relativos a la salud. Estos datos pueden ser tratados sin el consentimiento del interesado cuando se tratan de categorías especiales de datos personales y solo si es por razones de interés público en el ámbito de la salud pública. Se deberá de garantizar que estos datos no acaben en manos de terceros.
- Datos relativos a menores de edad. Estos datos merecen una protección específica, por lo que el consentimiento del titular de la patria potestad no debe ser necesario en caso de servicio preventivo ofrecido directamente a los menores de edad.
- Datos que revelen el origen étnico o racial. Se deberá proteger los datos que por su naturaleza sean sensibles con los derechos y libertades fundamentales del individuo.
- Datos personales. Este reglamento se aplicará a todos los datos personales ya bien sean con fines de investigación científica, histórica y/o estadística.

Una vez que se ha recogido los principios del reglamento y a qué tipo de datos aplica, este reglamento define dos acciones que recaen sobre los tratadores de los datos:

- Los tratadores de datos deberán proceder con la pseudonimización de todos los datos recogidos. Este procedimiento consiste en reemplazar los campos de información personal dentro de un registro de datos por uno o más pseudónimos. Este procedimiento deberá apoyarse de mecanismo de cifrado con clave secreta, cifrado determinista, función con clave almacenada, descomposición en *tokens*, etc.
- Los tratadores de datos deberán de informar al usuario del tratamiento de sus datos. Como se ha comentado, este tratamiento deberá ser lícito y leal. Se deberá informar al usuario de los datos que se están recogiendo, así como la medida en que dichos datos son o serán tratados. El consentimiento por parte del usuario debe darse mediante un acto afirmativo que refleje voluntad propia. El usuario dispondrá de mecanismos para acceder, rectificar y suprimir sus datos personales a fin de mantener el derecho al olvido.

Como se ha comentado, la adaptación de este reglamento en España está recogida por la LOPDGDD. La autoridad de control, independiente de las Administraciones Públicas, encargada de velar por el cumplimiento de esta adaptación, así como la protección de datos, es la Agencia Española de Protección de Datos (AEPD). Mediante esta agencia, el ciudadano puede interponer una reclamación sobre derechos a la entidad responsable, ofreciendo modelos para facilitar el ejercicio de los derechos y libertades de las personas físicas. También ofrece ayuda específica a menores y a responsables del tratamiento de los datos [51].

2.4 La seguridad en aplicaciones móviles

Se entiende como seguridad de las aplicaciones a las medidas de seguridad tomadas a nivel de aplicación para proteger al dispositivo móvil del robo o secuestro de los datos del usuario guardados dentro de la propia aplicación [52]. Estas medidas se deberán de tomar, en primera instancia, durante el desarrollo de la aplicación para después mantener esta seguridad a lo largo de la vida útil de la misma.

Estas medidas de seguridad no suelen ser habituales por parte de los desarrolladores. Esto es debido a que estos están más obcecados en otros aspectos como la experiencia de usuario, la usabilidad y la interfaz gráfica, ya que son los aspectos que resultan más atractivos para el usuario. Además, el desarrollo de estas medidas puede retrasar el trabajo de estos desarrolladores, ya que estas medidas tienen que ser compatibles en multitud de diferentes dispositivos y diferentes sistemas operativos móviles [53].

Pero pese a estos problemas, el número de empresas que apuestan por la seguridad de su aplicación es mayor cada año. Esto se ve reflejado en la Figura 18, donde se puede observar que el número de ataques a usuario de móviles ha ido disminuyendo a lo largo de los últimos 3 años. También se puede ver en la Figura 19, que el número de paquetes de instalación maliciosos ha ido disminuyendo. Esto nos puede indicar que las aplicaciones móviles cada vez son más seguras.

Aunque estas estadísticas parezcan positivas, no significa que no se tenga que seguir invirtiendo en seguridad. La empresa consultora Gartner, prevé que para este año "al menos el 50% de los ataques contra aplicaciones móviles podrían haberse prevenido gracias a la protección integrada dentro de las mismas" [55].

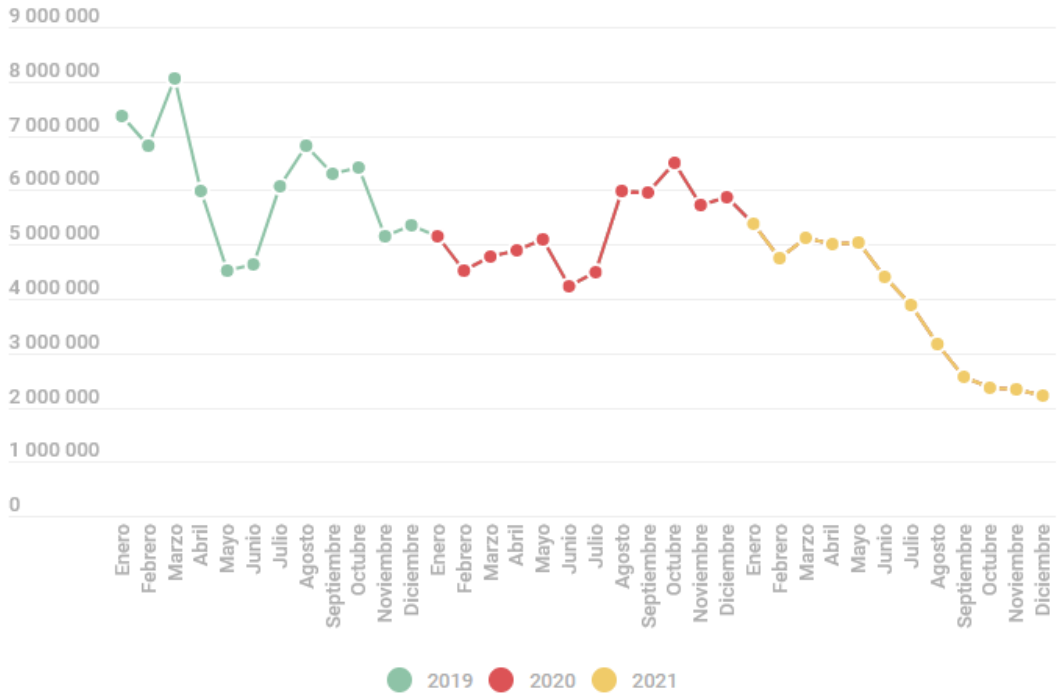

Figura 18. Número de ataques a usuarios móviles por mes y año [54].

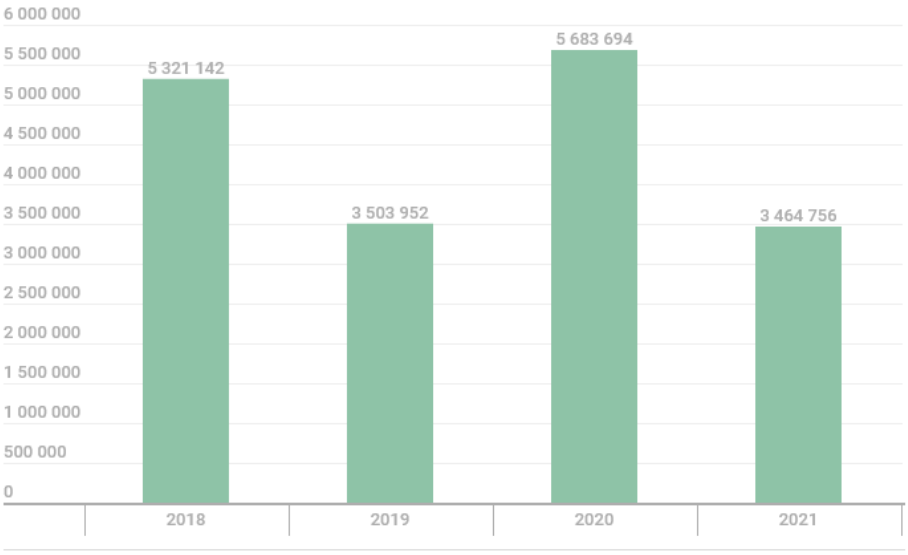

Figura 19. Número de paquetes de instalación maliciosos [54].

Una de las medidas que puede tomar un desarrollador a la hora de crear una aplicación es pasar esta aplicación por un proceso de *hacking* ético. Según el INCIBE (Instituto Nacional de Ciberseguridad), un *hacker* es aquella "persona con grandes conocimientos en el manejo de las tecnologías de la información que investiga un sistema informático para reportar fallos de seguridad y desarrollar técnicas que previenen accesos no autorizados" [41], por lo que someterse a este proceso de *hacking* ético sería dejar la aplicación en manos de un *hacker* para que este pueda identificar y reparar las posibles vulnerabilidades de la aplicación [56].

Es lógico pensar que un *hacker* puede solucionar los problemas del desarrollador, pero hay medidas de seguridad básicas que puede tomar el desarrollador durante el camino. Un ejemplo es la posibilidad de implementar el estándar de seguridad de aplicaciones móviles OWASP (*Open Web Application Security Project*). OWASP es una organización sin ánimo de lucro que trabaja para mejorar la seguridad del *software* y cuyo objetivo es desarrollar comunidades a través de eventos, ofrecer publicaciones y recursos educativos y apoyar la construcción de proyectos de impacto [57]. Entre los requerimientos de seguridad que ofrece este estándar destacan tocar los siguientes puntos [53]:

- Almacenamiento de datos y privacidad: No almacenar datos sensibles en el terminal del usuario, esto incluye *logs* del sistema y copias de seguridad, y si fuese necesario, encriptar estos datos con una clave derivada del *hardware* de almacenamiento seguro. Estos datos tampoco deben exponerse en la interfaz del usuario de una manera legible.
- Uso de claves criptográficas: Es necesario usar claves criptográficas para proteger los datos sensibles del usuario, pero estas claves no deben de ser reutilizadas y tienen que ser generadas con un generador de números aleatorios lo suficientemente robusto.
- Autenticación: Una aplicación segura tiene que contar con un mecanismo de autenticación, como la implementación de mecanismo de triple factor de autenticación, comentado en la Figura 10.

- Calidad del código: La app debe capturar y gestionar debidamente las posibles excepciones y debe estar firmada y provista con un certificado válido. Para poder valorar la calidad del código es recomendable el uso de herramientas que realicen un análisis estático y dinámico de la aplicación. El primer análisis lo que hace es examinar la app sin ejecutarla y realiza una aproximación de los posibles flujos de ejecución. El segundo permite analizar el comportamiento durante una ejecución real de la app. Ambos análisis tienen como fin buscar fuentes (código que tiene acceso a datos personales) y sumideros (código que puede filtrar datos personales fuera del dominio de la app) [58].
- Ingeniería inversa: Se deben implementar mecanismos que eviten la ingeniería inversa. Para ello, se pueden hacer uso de herramientas como el firmado de la aplicación utilizando claves públicas y privadas, la ofuscación del código, que hace ilegible el código, y el mecanismo "Anti-Clone", que permite evitar que la aplicación pueda ser clonada y ser distribuida con *malware* (tipo de *software* que tiene como objetivo dañar un sistema de información [41]).

Se ha comentado lo importante que es la seguridad a la hora de desarrollar una aplicación, pero una vez que esta aplicación se ha lanzado, cierta responsabilidad la tiene el usuario que hace uso de esta aplicación. Entre los consejos que puede seguir un usuario para sentirse más seguro y evitar que sus datos puedan quedar expuestos, se destacan los siguientes [55]:

- Evitar instalar aplicaciones que no sean consideradas seguras y que no estén certificadas por la tienda. Una herramienta que se puede usar con Android es "Google Play Protect", la cual escanea las aplicaciones en busca de amenazas.
- Usar los mecanismos de autenticación ofrecidos por la aplicación, ya que esto protege de que cualquier otro usuario suplante tus credenciales.
- No conceder todos los permisos que te exija la aplicación, solo conceder los que consideres necesarios. No es lógico conceder el permiso de tus contactos y ubicación a una aplicación que hace que se encienda el *flash* del dispositivo. Si una app pide más permisos de los que el usuario cree necesarios, es porque seguramente se trate de una aplicación maliciosa.
- Mantener actualizadas las aplicaciones, ya que esto permite que si ha habido algún fallo de seguridad y los desarrolladores lo conocen, cubrirán ese fallo con una actualización.
- Usar una aplicación de seguridad, como un antivirus, ya que esta aplicación puede cubrir tareas como la de proporcionar privacidad para las aplicaciones, protección antirrobo, modos VPN, limpieza y bloqueo de archivos, etc [59].

2.4.1 La seguridad en otras aplicaciones móviles

La seguridad en las aplicaciones móviles es una cuestión compleja como hemos visto. En nuestro caso además resulta especialmente complicada porque utilizamos un protocolo orientado a comunicación máquina a máquina, MQTT, para dar servicio a clientes reales. Aunque es cierto que el protocolo MQTT se puede securizar, parece

interesante repasar qué medidas de seguridad se toman en aplicaciones de móviles más habituales, refiriéndonos en este caso a las aplicaciones de redes sociales [60].

En la Figura 20 se puede observar las 10 aplicaciones más descargadas desde las tiendas de aplicaciones que ofrecen los sistemas operativos de los diferentes dispositivos móviles. Entre ellas, se encuentran las aplicaciones de "WhatsApp Messenger", "Telegram" y "Facebook".

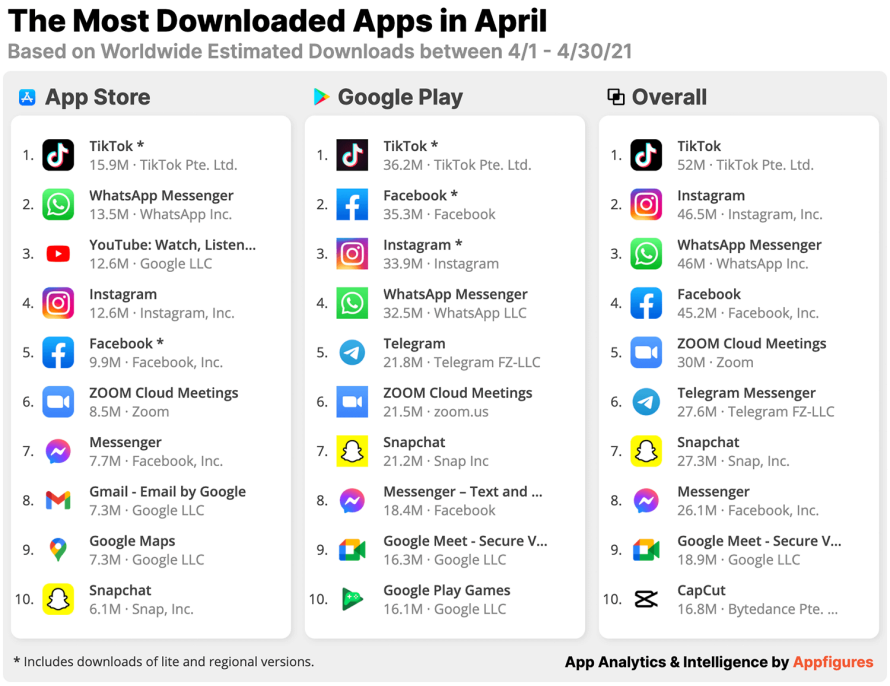

Figura 20. Aplicaciones más descargadas en Abril de 2021 [61].

Estas aplicaciones no son solo las más descargadas, sino que como se puede ver en Figura 21, la mayoría de los usuarios que disponen de un *smartphone* hacen uso de ellas, siendo el número total de usuarios con *smartphones* en 2021 de 3.800 millones de personas [62].

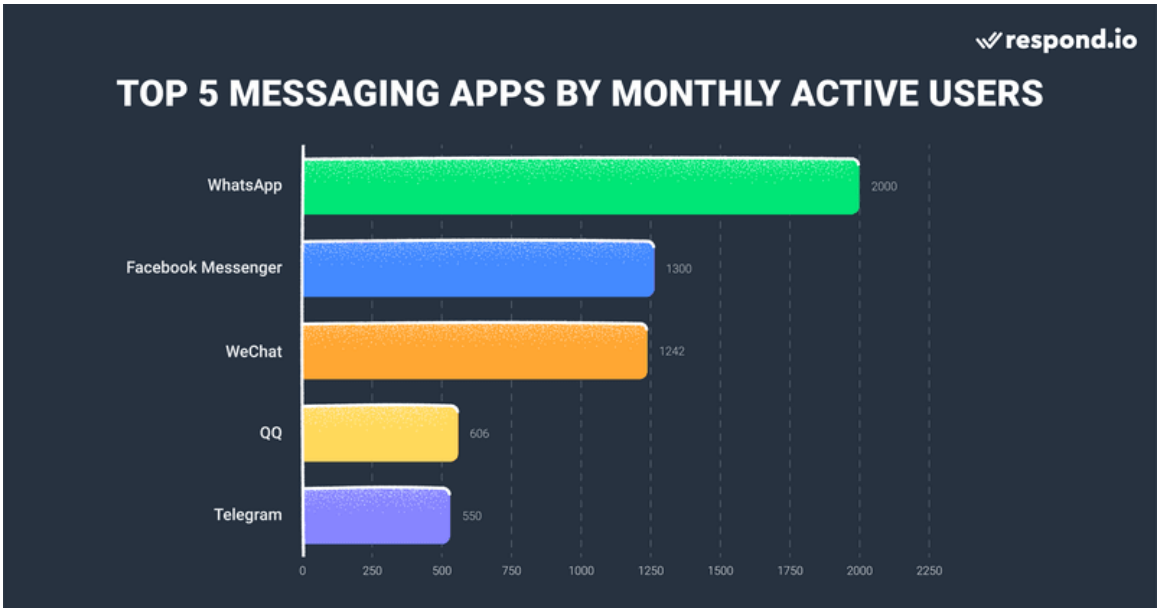

Figura 21. Las 5 aplicaciones de mensajería más populares en función del número de usuarios activos mensuales [63].

Se han elegido por lo tanto Whatsapp, Facebook y Telegram para hacer un repaso de las principales medidas de seguridad y privacidad que ofrecen a los usuarios:

WhatsApp**:** Por defecto, cualquier usuario de WhatsApp puede ver la foto de perfil, estado y última hora de conexión de la persona que se acaba de instalar esta aplicación. Además, es necesario introducir un número de teléfono personal para poder acceder a este servicio, en vez de escoger un alias y una contraseña. También ofrece cifrado extremo a extremo (Ver Figura 22), que "asegura que solo el emisor y el receptor puedan leer lo que es enviado; y que nadie más, ni siquiera WhatsApp, lo pueda hacer. Esto es porque los mensajes están protegidos con un clave, y solo el emisor y el receptor tienen esta clave que permite leer los mensajes. Para mayor protección, cada mensaje que se envía tiene su propia clave " [64]. También ofrece otras funcionalidades para la autenticación, como la verificación en dos pasos, que exige una segunda credencial para autenticarse, y la verificación biométrica, que permite usar tu huella dactilar o rostro para acceder a la aplicación. En cuanto a la privacidad, tiene el defecto de hacer un *backup* (Copia de seguridad) del historial de las conversaciones en la nube, lo que hace que exista un problema de privacidad si esta información no estuviese cifrada [65].

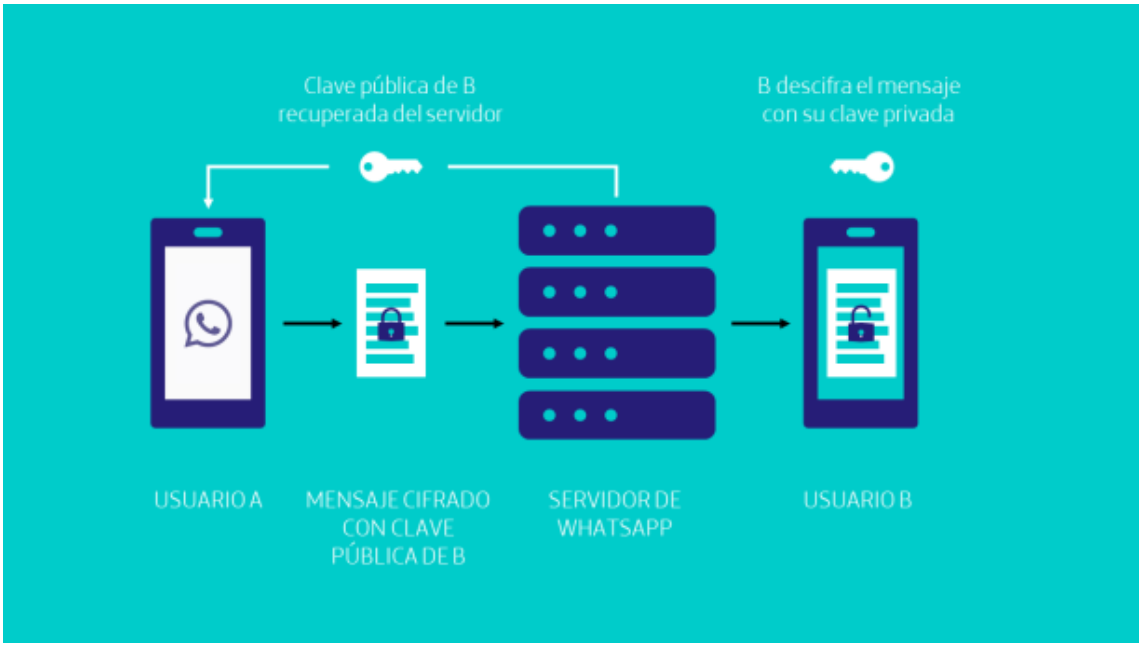

Figura 22. Explicación simplificada de cómo funciona el cifrado extremo a extremo de WhatsApp [66].

Facebook Messenger**:** Facebook pertenece a Meta Platforms, al igual que WhatsApp, por lo que muchas medidas de seguridad tomadas son idénticas a las de WhatsApp. Facebook Messenger también ofrece cifrado extremo a extremo, aunque esta opción no está incorporada por defecto en todos los chats, sino que ofrece la funcionalidad de "conversaciones secretas". Esta funcionalidad tiene que ser activada de forma explícita. Esto es un problema, ya que la aplicación da a entender que los mensajes que no considera el usuario como secretos corren el riesgo de ser accedidos por terceros [67]. También ofrece la funcionalidad de "modo de desaparición", donde los mensajes enviados desaparecen automáticamente cuando se cierra la ventana del chat [68]. También te avisará si una persona realiza una captura de pantalla de una conversación secreta, aunque no hará nada más al respecto [69]. Respecto a la

autenticación, puedes identificarte con una cuenta de Facebook o dando tu número de teléfono.

Telegram**:** Si bien es la aplicación con menor número de usuarios, ha sido la aplicación de mensajería más segura desde los inicios. Para demostrar que tenían confianza absoluta en el nivel de seguridad de su aplicación, la empresa organizó concursos en los que se retaba a los usuarios a intentar descifrar su cifrado con premios de hasta 300.000$ [70]. Respecto al cifrado, Telegram utiliza el protocolo MTProto 2.0 (*Mobile Transport Protocol*). Este protocolo, que se encarga de cifrar los mensajes entre el cliente y el servidor, es un protocolo propietario de Telegram. Esto hace que, aunque Telegram esté basada en la nube y los mensajes e información de los usuarios se sincronicen constantemente con los servidores de Telegram, esta información siempre estará cifrada. Además, Telegram ofrece un cifrado extremo a extremo para aquellos chats seleccionados por el usuario. Estos chats son los llamados "chats secretos", por lo que ocurre el mismo problema que con Facebook Messenger, ya que esta opción no está implementada por defecto [71]. Respecto a la privacidad que ofrece a la hora de almacenar los datos personales del usuario, se sabe que los centros de datos donde se almacena la información del usuario son gestionados por terceros, pero se alquila un espacio a Telegram, por lo que los datos personales no se comparten con los centros de datos, sólo se almacenan utilizando los servidores. Según las políticas de privacidad de la plataforma, todos los datos se almacenan fuertemente encriptados para que los ingenieros locales de Telegram o los intrusos físicos no puedan acceder a ellos [70].

Se ha visto que las medidas comunes en este tipo de aplicaciones es aplicar el cifrado extremo a extremo, en la que solo el emisor y el receptor pueden llegar a conocer el contenido de los mensajes (mismo caso que empleando TLS) e implementar varias funcionalidades para autenticar al usuario, como la autenticación en 2 pasos.

2.5 La seguridad en los servidores

La palabra servidor tiene dos definiciones en el campo de las computadoras dependiendo del punto de vista: la primera definición tiene que ver con la computadora que pone sus recursos a disposición a través de la red, desde un punto de vista físico o de *hardware*, y la segunda se refiere al programa o conjunto de programas que funcionan dentro de una computadora, desde el punto de vista del *software* [72]. Nos interesan ambas definiciones, ya que, para lograr el fin de este proyecto, se necesita proteger el contenido del servidor físico y luego desarrollar un programa servidor, también llamado *back-end*, que controle la comunicación entre los diferentes terminales.

Cuando se despliega un servidor y se ponen sus servicios a disposición, se sigue una arquitectura basada en cliente-servidor (Ver Figura 23). El cliente es quien solicita un recurso al servidor, y es este quién envía su respuesta con el recurso solicitado. En el protocolo MQTT, los clientes serían los distintos publicadores y/o suscriptores, y el servidor sería el bróker.

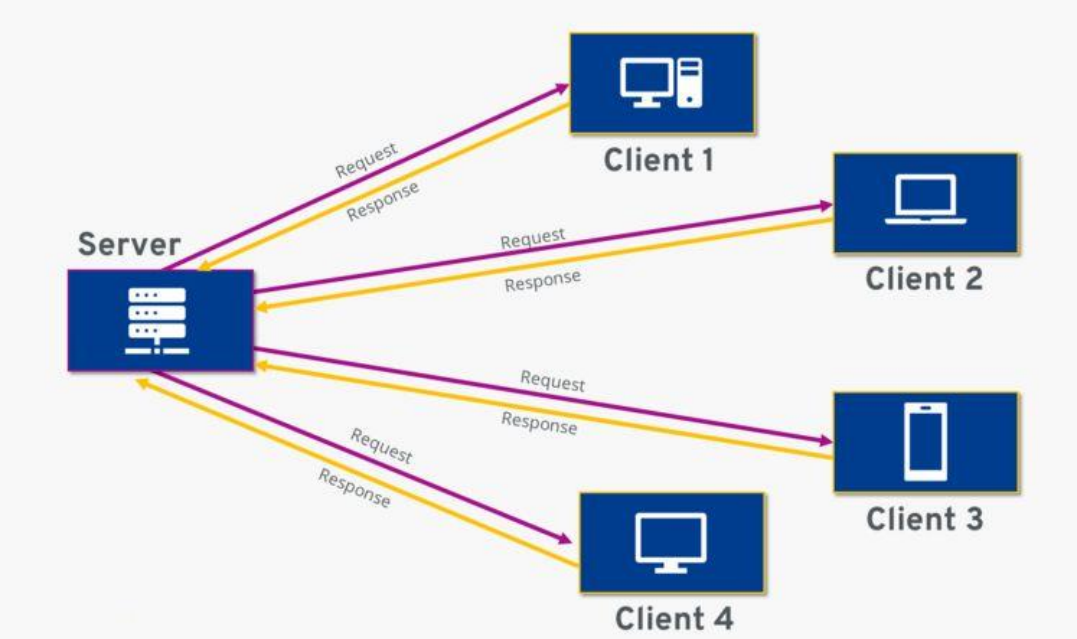

Figura 23. Arquitectura cliente-servidor [72].

Para que un servidor pueda funcionar apropiadamente, este tiene que estar configurado para escuchar las solicitudes de los diferentes clientes que quieran acceder al servicio. Esto hace que los servidores se conviertan en un objetivo para ataques, ya que guardan datos e información confidencial. En este sentido, para cumplir los tres pilares de la seguridad informática, se tienen que tomar varias medidas para proteger a un servidor.

Las medidas de seguridad dependerán del uso que se le da al servidor. Para ello habrá que caracterizar al servidor de entre los diferentes tipos de servidores que puede haber. De entro los más empleados destacan:

Un servidor web es un *software* que se utiliza para servir archivos a sitios web en Internet y es responsable de garantizar que la comunicación entre el servidor y el cliente sea segura y sin fallos. Este software hace de enlace entre el servidor físico y el terminal del usuario de modo que, cuando el cliente realice una petición al servidor web, este coge los archivos del servidor físico y se los envía al usuario. Este tipo de servidor puede ofrecer el servicio a varios usuarios simultáneamente. Entre los más comunes se encuentran Nginx, Apache y Cloudfare Server [73].

Un servidor de base de datos permite la organización de la información mediante el uso de tablas índices y registros. Este tipo de servidor permite a los usuarios realizar las siguientes operaciones: especificar la estructura, el tipo y las restricciones de los datos; y permitir la inserción, actualización, eliminación y consulta de los datos. Es importante distinguir entre la base de datos y el DBMS (Sistema de Gestión de Bases de Datos), ya que la primera es el almacén donde se guardan los datos y la segunda es la herramienta necesaria para acceder y manipular ese almacén de información [74]. En un servidor de base de datos, habrá implementado un DBMS que hará uso de una base de datos. Entre los gestores de bases de datos comerciales más usadas se encuentran Oracle, IBM DB2 y Microsoft SQL [75].

Los servidores de correo son el tipo de servidor más antiguo (sus inicios datan de 1960) y el más usado en el mundo. Las funciones que ofrecen este tipo de servidor es la del procesado, almacenamiento, envío, recepción reenvío y recepción de mensajes. Estos servidores se componen de un servidor SMTP (*Simple Mail Transfer Protocol*), que se encarga de enviar los correos a su destino; un servidor POP (*Post Office Protocol*) o IMAP (*Internet Message Access Protocol*), el cual se encarga de acceder a estos correos enviados; y un cliente de correo, que es el *software* que permite leer y enviar los correos [76].

Un servidor DNS (*Domain Name System*) permite la traducción de un nombre legible por una persona, como por ejemplo *www.google.es*, en una dirección IP y viceversa. Estas direcciones IP son las que leen las computadoras para conectarse entre sí. Cuando se introduce en el navegador el nombre *www.google.es*, el navegador hace una petición a un servidor DNS para traducir este nombre y obtener la dirección IP a la que se desea acceder. Lo usuarios usan los nombres legibles porque son más fáciles de recordar y porque las direcciones IP suele variar con frecuencia mientras que el nombre se mantiene estable en el tiempo [77].

Los servidores de seguridad disponen de un software especializado para detener intrusiones maliciosas. Normalmente están equipados con software antivirus, *antispyware*, *antiadware*, además de contar con cortafuegos de diferentes niveles o capas del modelo OSI (*Open Systems Interconnection*) con la finalidad de evitar ataques. La colocación física de estos servidores suele estar antes de que cualquier conexión entre al servidor funcional, como un servidor web, para así proteger a este de intrusiones u otro tipo de ataques [78].

Por último, se denominará como *cluster* de servidores a la agrupación de varios servidores dedicados a diferentes fines, como, por ejemplo, tener un servidor de cada tipo anteriormente nombrado para un objetivo compartido, como puede ser dar servicio a una misma empresa. La ventaja que tienen esta clase de sistemas es que son más fiables y permiten escalabilidad [79].

Entre las medidas que se pueden tomar para mantener los tres pilares de la seguridad informática están la de crear una política de contraseñas efectiva, donde cada usuario sólo tenga acceso a aquello que le compete; el despliegue de un cortafuegos (*firewall*) que solo permita conectarse directamente a las máquinas imprescindibles y que deje los puertos abiertos necesarios para el funcionamiento normal del *back-end*; deshabilitar todo tipo de servicios prescindibles como Telnet, FTP, HTTP, etc; habilitar conexiones SSH seguras mediante el uso de llaves criptográficas para que el administrador pueda conectarse al servidor de forma segura; implementar ambientes aislados de ejecución donde cada tipo de servidor se ejecute en una computadora diferente o más actualmente en máquinas virtuales o contenedores distintos; por último, realizar auditoría de archivos e implementar sistemas de detección de intrusos que permita llevar un control de los ficheros del sistema.

A lo largo de este trabajo, se dará por supuesto que el acceso físico al servidor es limitado o incluso inaccesible, ya que los clientes del servicio acceden de forma telemática al servicio y no de forma física, por lo que se supondrá que la protección física del servidor es impenetrable.

Para implementar este tipo de servidores, se pueden usar diferentes tipos de dispositivos. En el caso del proyecto TwizyLine, se decidió usar una Raspberry Pi 3 modelo B+. En ella, se instalaron las dependencias de Python para el desarrollo del *software* que gestiona el *back-end*, y el Gestor de Bases de Datos MariaDB para la implementación de la base de datos SQL [74]. En el siguiente apartado se explicarán las ventajas y desventajas del uso de un dispositivo como este para actuar como servidor.

2.6 Vehículos

La seguridad en los vehículos es de las características más importantes que un posible comprador busca en su futuro automóvil [81]. Pero esta seguridad que se busca no se refiere a la seguridad que tiene el vehículo ante ciberataques, sino que tienen que ver con elementos como el cinturón de seguridad, el airbag, ayudas a la conducción, etc. Estos sistemas de seguridad ayudan a que el vehículo tenga una mayor eficacia y estabilidad cuando está en marcha, y a reducir los daños que se pueden producir sobre los pasajeros cuando el accidente es inevitable.

Los sistemas de seguridad (Ver Figura 24) se clasifican según su función: seguridad activa y pasiva. La primera hace referencia a la seguridad que previene de los accidentes, como los sistemas de frenado, control eléctrico de estabilidad, frenada autónoma, control de tracción, etc; y la segunda hace referencia a aquella seguridad que minimiza los daños a los ocupantes una vez producido el accidente, como el cinturón de seguridad, airbags, estructura de deformación programa, etc [82].

Figura 24. Sistemas de seguridad en un vehículo [82].

Todos los sistemas anteriormente comentados están controlados por dispositivos electrónicos. Estos dispositivos son las ECUs (*Electronic Control Unit*). Dichos dispositivos incorporan *software* para poder ofrecer las funcionalidades anteriormente mencionadas. Es fácil creer que estos sistemas lleven poco tiempo en los vehículos, pero el primer coche de producción que incorporó *software* integrado se fabricó en el año 1977. Este vehículo tenía una ECU que gestionaba la sincronización electrónica de la chispa que prendía el combustible con el motor.

Actualmente, los vehículos dependen de hasta 100 ECUs que controlan y supervisan desde la cadena cinemática hasta los sistemas de seguridad anteriormente mencionados [83].

Como los vehículos que se fabrican hoy en día están cada vez más conectados por el aumento del número de ECUs, parece lógico pensar que la seguridad de un vehículo no solo se queda en los conceptos de seguridad pasiva y seguridad activa, si no que el concepto de ciberseguridad tiene que añadirse a la definición de seguridad de un vehículo.

Las ECUs conviven en el vehículo empleando diferentes tipos de protocolos y de redes que se conectan entre sí, las más importantes actualmente CAN (Controller Area Network) y Ethernet. CAN es un estándar de comunicación bus para vehículos y está diseñado para permitir que diferentes dispositivos se comuniquen entre sí sin necesidad de un ordenador central. Esto funciona debido a que los mensajes enviados por cualquier dispositivo son recibidos por el resto de los dispositivos que están conectados a la red [84]. Por las características de este protocolo, si un ciberdelincuente puede acceder a la red de ECUs de un vehículo, este podrá enviar órdenes al vehículo desde una ubicación remota para, entre otras cosas, robar datos privados y corporativos, rastrear vehículos individuales o flotas enteras y secuestrar funciones no críticas y críticas para la seguridad [85]. En general los fabricantes han confiado en algún elemento firewall a la entrada de la red en el caso de que existiera entrada OBD, o bien en que solo ellos tenían las bases de datos que permitían decodificar el significado del contenido de los datos. Esta visión es claramente ineficaz en las circunstancias actuales. Para poder proceder con este tipo de ataques, primero deberá de existir una puerta de entrada. Prácticamente todos los vehículos fabricados ofrecen conectividad integrada, anclada o por medio de un teléfono inteligente. Ya en 2016, los vehículos representaron un tercio de todos los nuevos dispositivos celulares. Esta conectividad crea una superficie de ataque con la que se puede acceder a la red CAN del vehículo [85].

Sin embargo, la propia evolución de las necesidades del vehículo ha venido a ayudar a superar este problema, dado que la mayor necesidad de recursos a obligado a introducir como red central en los últimos vehículos a las redes Ethernet, en las que es fácil implementar medidas de seguridad como las que se han mencionado en secciones anteriores.

2.6.1 Ciberataques a vehículos

Para lograr entender de mejor manera la ciberseguridad que debe haber dentro de los sistemas electrónicos de un vehículo, se va a abordar el tema de la ciberseguridad desde el enfoque de que el vehículo es víctima de un ciberataque. Este enfoque es necesario para entender las puertas de entrada que tiene un vehículo a los ciberataques. Para poder ofrecer este enfoque, se va a apoyar esta sección explicando el proyecto OMSP (*Open Mobility Security Project*) [86].

OMSP es un proyecto abierto dedicado a estandarizar un marco de controles técnicos para evaluar la seguridad a nivel de hacking en todo tipo de vehículos. Este proyecto nos puede ayudar a definir las posibles fallas de seguridad que puede tener un vehículo.

Figura 25. OMSP logo [86].

Este proyecto tiene como objetivo estandarizar los procesos de auditoría técnica de ciberseguridad de un vehículo de una forma repetible, evidenciable y orientada al cumplimiento normativo y regulador. Esta estandarización que proponen ofrece una relación entre los posibles activos que se pueden encontrar dentro de un vehículo y las posibles amenazas que les afectan. Con estas posibles amenazas, se podrían obtener los diferentes métodos de ataque. En el caso de que sea posible llevar a cabo estos ataques, se podrían obtener las medidas necesarias para mitigarlos. En las siguientes líneas se exponen las posibles amenazas de seguridad que podrían existir en función de los diferentes activos:

A través de comportamientos inesperados, como conectar un teclado a la conexión USB del vehículo, es posible generar un problema en el vehículo o incluso instalar *software* malicioso a través de esta conexión. También puede haber fallas en los *kits* de expansión, cuya misión es añadir funcionalidades extras al vehículo de fábrica. Estos *kits* están conectados a las ECUs del vehículo, por lo que si se consigue suplantar este kit (ataque *hijacking*), se podría manipular el comportamiento del vehículo.

Como los vehículos usan tecnologías de conectividad celulares, como 4G o 5G, esta puede ser otra puerta de entrada. Estas tecnologías transmiten por difusión, por lo que, si la información que intercambia el vehículo con la estación base no está cifrada, se podría capturar esa información para que un atacante configure y establezca un punto de acceso falso. También se podría husmear el tráfico en aquellos vehículos que ofrezcan conexión vía WiFi a sus usuarios, para después manipular o denegar este servicio.

Los fabricantes de vehículos a veces ofrecen una aplicación que permite a sus clientes interactuar con sus vehículos, ofreciendo funcionalidad como abrir y cerrar el coche, encender las luces, gestionar la alarma, etc. Estas posibilidades de control remoto implican una mayor responsabilidad por parte del fabricante a la hora de validar las peticiones de estas aplicaciones, ya que estas peticiones podrían ser manipuladas para hacerse con el control del coche.

Otra posible puerta de entrada sería el GPS, ya que son tratadas como puertas de entrada de confianza, por lo que un atacante puede aprovechar esa confianza para manipular el posicionamiento y la navegación.

A través de los navegadores web incorporados por el fabricante pueden surgir fallas de seguridad. Estos navegadores deberían de estar limitados para no acceder a los recursos locales ni para visitar las IP locales del vehículo, ya que cierta parte de la arquitectura de un vehículo está construida sobre *Ethernet*. Otros problemas podrían surgir mediante la instalación de *plugins* no autorizados dentro del navegador.

Todo lo comentado anteriormente, mostrado de una forma resumida por la OSMP en la Figura 26, ha de tenerse en cuenta por parte de los fabricantes de automóviles y por aquellos usuarios que deciden añadir *kits* de expansión a sus vehículos, para así proteger la seguridad de sus usuarios y evitar posibles accidentes que puedan acabar con sus vidas.

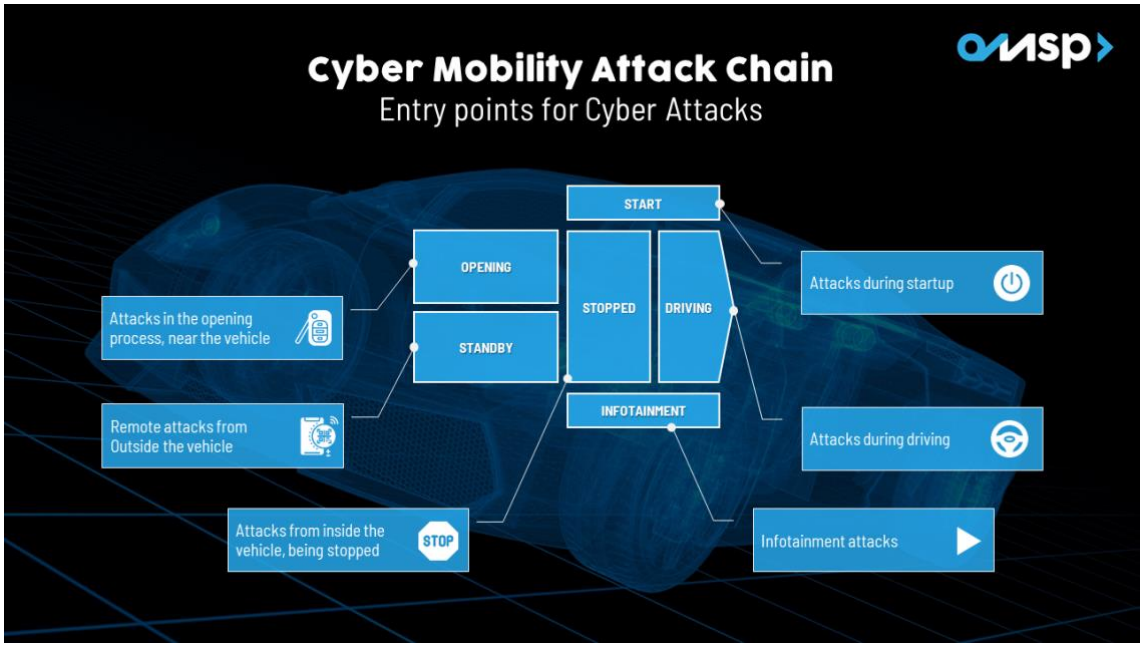

Figura 26. Posibles puntos de entrada de un vehículo a un ciberataque [86].

3 Análisis de fallas de seguridad

En esta sección se va a describir el análisis realizado a la actual arquitectura del proyecto TwizyLine (ver Figura 4). La base de documentación sobre este análisis se deposita en los proyectos de fin de carrera del equipo del TwizyLine [87] [12] [74]. Otra de las bases de documentación han sido los recursos de información publicados por la OWASP [57], cómo "*Top 10 Mobile Risks*" [88], o el "*OWASP Top Ten*" [89].

Este análisis se ha realizado de una forma general, basándose en los recursos que puede almacenar cada componente. El resultado de este análisis ofrece los riesgos de seguridad, causas de estos riesgos, tipos de ataque que se pueden recibir por culpa de estos riesgos y prevención ante estos riesgos. El fin de este análisis es el de entender los problemas de seguridad que pueden existir en esta arquitectura.

3.1 Componentes susceptibles de análisis

A la vista del sistema ideado para TwizyLine, el análisis OWASP debería ser realizado sobre tres componentes básicos: el vehículo, la aplicación móvil y el servidor.

Este análisis está realizado sin tener en cuenta el marco legal de la RPGD. Como se ha visto en el estado del arte hay multitud de dimensiones que afectan a la seguridad, y este es un punto que se debe tener en cuenta, ya que esta normativa es muy clara en muchos aspectos, como que ningún usuario pueda acceder a otros datos que no sean los suyos o la de informar al usuario que tipo tratamiento van a tener sus datos y con qué fin. Este análisis no se ha realizado por límite de tiempo y queda previsto para líneas futuras

Comenzando por el vehículo, hubiera sido necesario realizar un análisis de las posibilidades que tiene un *hacker* de manipular físicamente dicho vehículo. En este caso los resultados serían bastante evidentes, dado que el modelo Twizy no está concebido como un vehículo que sea necesario proteger frente a ningún tipo de ataque. De por sí incluye una entrada OBD de la cual se puede leer en plano toda la información que circula en el único bus de comunicación CAN del vehículo, y también es posible escribir. Además, las nuevas ECUs se unieron a este bus directamente. En este sentido, asegurar esta parte del sistema puede ser un gran reto. Sin embargo, en este Trabajo Fin de Grado se ha decidido obviar este análisis por varias razones:

1. En primer lugar, porque en un servicio real se utilizarían probablemente vehículos más modernos, que de por sí implementarían algún tipo de defensa frente a intrusos que el Twizy por antigüedad no lo incorpora.
2. En segundo lugar, se puede suponer que de alguna forma es posible evitar el acceso físico al vehículo, de tal manera que solo quede una vía de entrada al mismo: la ECU de comunicación, que convierte al Twizy en un vehículo conectado.
3. En tercer lugar, asumiendo el punto anterior, solo sería necesario fortificar la ECU de comunicación mediante: a) autenticación frente al servidor b) confidencialidad de las comunicaciones. Este tipo de medidas serían las mismas que se tienen que implementar en el servidor, debido a que esta ECU tiene instalada una distribución de Linux (igual que el servidor), y por lo tanto

analizando este es posible determinar las medidas que se deben incorporar en esta ECU.

Siguiendo con el servidor, el análisis realizado a este componente se ha llevado a cabo teniendo en cuenta que el acceso físico a este es limitado o prácticamente imposible. Esto es debido a que los clientes acceden al servicio acceden de forma telemática a través de la aplicación móvil, y no de forma física, por lo que se supondrá que la protección física del servidor es hermética.

Esto nos deja con el análisis del servidor y de la aplicación móvil. Después se realiza un análisis por separado, ya que hay riesgos encontrados en un componente que no se pueden encontrar en el otro componente. Además, se ha realizado un análisis estático de la aplicación móvil, por lo que se comentarán los resultados obtenidos.

3.2 Análisis de móvil y servidor

Para poder entender este análisis realizado, se van a explicar que tipos de recursos son manejados por la aplicación móvil y por el servidor:

- Como se puede ver en el proyecto de fin de carrera de A. Mazaira [87], los únicos recursos que son almacenados dentro del terminal móvil son las credenciales del usuario. Esto es debido a que el resto de la información es almacenada en el servidor. Cuando el usuario rellena el formulario de inscripción de la aplicación, los datos rellenados son enviados y almacenados en el servidor. Las credenciales consisten en un nombre de usuario y una contraseña. Las credenciales se guardarán en el terminal una vez el usuario haya iniciado sesión en la aplicación por primera vez. A partir de este momento, cada vez que el usuario inicie la aplicación, esta iniciará sesión automáticamente con las últimas credenciales guardadas en el terminal. El fin de esta funcionalidad es mejorar la experiencia del usuario.
- Como se puede ver en el proyecto de fin de carrera de S. Pilar [74], en el servidor se ha implementado una base de datos en la que se almacena la información necesaria para el funcionamiento del proyecto global de TwizyLine. Este funcionamiento implica que la aplicación reciba la información necesaria una vez el usuario ha sido autenticado por el servidor. También implica la directiva de órdenes que reciben los vehículos de servicio para que sean aparcados de manera autónoma. Los recursos almacenados dentro del servidor son los almacenados dentro de la base de datos.
- Además, ambos componentes manejan datos en tránsito. Estos datos son todo intercambio de información que ocurre entre la aplicación móvil del servidor. Esta información puede recoger datos de sesión, localización del usuario, credenciales, etc.

En la siguiente tabla se puede observar el análisis conjunto realizado de la aplicación móvil y al servidor:

Tabla 1. Análisis de los riesgos, causas, tipos de ataque y prevención de los recursos manejados por la aplicación móvil y el servidor.

RIESGOS	CAUSAS	TIPOS DE ATAQUE	PREVENCIÓN
Comunicación insegura. Los datos intercambiados pueden ser cambiados sin que este cambio sea detectable.	La principal causa es el uso de protocolos de comunicación no seguros. En el caso de que sí que se esté empleando este tipo de protocolo, como TLS, no configurar y validar correctamente una conexión TLS mediante la comprobación de certificados y uso de cifrados débiles llevan a una comunicación insegura. El uso de un protocolo de transporte seguro no significa que el componente lo haya implementado correctamente.	Los ataques dirigidos son los más fáciles de realizar, ya que lo difícil es encontrar esta comunicación insegura entre todo el tráfico de una red. Entre los posibles ataques destaca el ataque *man-in-the-middle*. En este tipo de ataques, una tercera persona se coloca en medio de dos partes que se comunican entre sí y modificar los mensajes que se intercambian sin que ninguna de las dos partes lo sepa [90].	Asumir siempre que el canal de transporte es un medio inseguro y por lo tanto hay que usar un protocolo de comunicación seguro. Configurar correctamente este protocolo con: uso de algoritmos de cifrado fuerte y estándar, uso de certificados firmados por un CA de confianza, establecer una conexión segura solo después de verificar la identidad del servidor o del cliente, no enviar datos sensibles a través de otros canales. Si es posible, emplear una capa de cifrado añadida antes de enviar los datos por el canal seguro.
Autenticación insegura. Un usuario no es quién dice ser que es.	El componente emplea políticas de contraseñas débiles, usa un sistema de reconocimiento de huellas dactilares como método de autenticación, almacena cualquier contraseña localmente o es capaz de ejecutar de forma anónima una solicitud de servicio API *backend* sin proporcionar un *token* de acceso, es decir, sin que el servidor verifique la identidad asociada a la petición.	Falsear el proceso de autenticación. Esto permite que el atacante ejecute de forma anónima funcionalidades que afectan a los usuarios. En el caso de una contraseña débil, el atacante podrá deducir la tabla de *hash* con la que la contraseña del usuario es almacenada.	Asegurarse de que todas las solicitudes de autenticación se hagan en el servidor, y así asegurarse que los datos de la aplicación solo estén disponibles después de una autenticación exitosa. Para ello, se tendrán que reforzar los controles de autorización y autenticación del servidor. No almacenar la contraseña del usuario en el dispositivo y emplear una política de contraseñas robusta.

Autorización insegura. La autorización es el acto de comprobar que el usuario identificado tiene los permisos necesarios para realizar la petición solicitada.	Si existe un problema de autenticación insegura, también hay riesgo de que ocurra un problema de autorización insegura. Como causas de autorización insegura: No realizar comprobaciones de autorización en funcionalidades ocultas; transmitir los permisos del usuario al servidor como parte de una solicitud; permitir ejecutar una funcionalidad de privilegio alto usando un usuario de menor privilegio.	Los ataques pueden ser realizados mediante el uso de herramientas automatizadas. Una vez el atacante sepa cuál es la vulnerabilidad del proceso de autorización, este se conecta al componente cómo un usuario legítimo e intenta escalar privilegios. Esta clase de ataques se realizan a través de *malware* instalado en el componente.	Reforzar los controles de autorización y autenticación en el lado del servidor. Verificar los roles y permisos del usuario empleando únicamente la información contenida en el sistema *back-end*. Este sistema deberá verificar de forma independiente que cualquier identificador entrante asociado a una solicitud que venga junto con la identificación coincida y pertenezca a la identidad entrante.
Baja calidad del código. Problemas con el propio lenguaje de programación.	Causado por malas prácticas a la hora de programar que pueden dar resultado a sumideros y desbordamientos de búfer.	Ejecutar código de forma remota mediante el uso de *malware*. Esto puede provocar robo de información y degradación del rendimiento de la aplicación.	Realización de análisis estático y dinámico del flujo de datos de la aplicación. Estos análisis permiten facilitar la búsqueda de desbordamiento de búfer y figas de memoria.
Manipulación del código.	Todo código móvil es vulnerable a la manipulación del código debido a que este código se ejecuta en un entorno que no está bajo el control del desarrollador. En el caso del servidor, el software se ejecuta en un entorno seguro, donde el usuario no tiene acceso directo a este, pero al emplear un gestor de bases de datos, se podría	En el caso de la aplicación: el atacante podrá realizar cambios en el paquete de la aplicación y en los recursos que están dentro del paquete de la aplicación. También podrá redirigir o reemplazar las APIs del sistema para ejecutar código malicioso. También podrá modificar los archivos binarios para crear una nueva	En el caso de la aplicación móvil, esta tiene que ser capaz de detectar en tiempo de ejecución la manipulación de código y reaccionar adecuadamente en consecuencia y en tiempo de ejecución. Una aplicación que ha sido modificada se ejecutará en un entorno rooteado (entorno donde se tienen permisos de superusuario, es decir, control total del sistema operativo), por lo

	llegar a inyectar datos en el código a través de los datos que se le piden introducir al usuario en la aplicación.	aplicación y subirla a una tienda de terceros y hacerla pasar como la aplicación original. En el caso del servidor: el atacante podría realizar un ataque de inyección SQL. Este ataque consiste en la inserción de una consulta SQL a través de los datos de entrada del cliente a la aplicación. Un ataque exitoso puede leer datos de la base de datos, ejecutar operaciones sobre la base de datos e incluso emitir comando al sistema operativo.	que será necesario tratar de detectar estos entornos y reaccionar en consecuencia. En el caso del servidor, el *software* debe de usar sentencias preparadas mediante el uso de consultas parametrizadas, validar los datos de entrada procedentes de la aplicación cuando sea necesario realizar una consulta SQL y tratar de no suministrar como dato de entrada la suministrada por el usuario.

En el caso de la aplicación móvil se ha encontrado un riesgo adicional:

Tabla 2. Riesgo, causa, tipo de ataque y prevención exclusiva de la aplicación móvil.

RIESGOS	CAUSAS	TIPOS DE ATAQUE	PREVENCIÓN
Uso inapropiado de la plataforma. Esto incluye cualquier uso indebido de una característica que forma parte del sistema operativo móvil, como el uso indebido de permisos o de controles de seguridad que forman parte del móvil.	Violación de directrices publicadas por la plataforma y de prácticas comunes. También puede ser causado por un uso incorrecto de las llamadas a una API o por fallos en los modelos de permisos de la plataforma.	Cualquier llamada a una API expuesta puede ser fruto de un vector de ataque. Por ejemplo, un uso indebido es almacenar las credenciales en el terminal sin usar una API de la plataforma que ofrezca un almacenamiento seguro, como *Keychain* para plataformas iOS.	Seguir las directrices publicadas por cada plataforma y hacer un buen uso de las prácticas comunes.

En el caso del servidor se han encontrado los siguientes riesgos específicos para él:

Tabla 3. Riesgo, causa, tipo de ataque y prevención exclusiva del servidor.

RIESGOS	CAUSAS	TIPOS DE ATAQUE	PREVENCIÓN
Uso no autorizado de las comunicaciones del servidor, baja disponibilidad del servicio o fallo sistémico del servicio.	El servidor está excesivamente abierto.	Denegación de servicio Ataque de fuerza bruta.	Configurar servidor para dejar solo abiertos los puertos estrictamente necesarios. Instalar *software* específico para mitigar este tipo de ataque como por ejemplo *firewalls*.
Una persona no autorizada puede acceder de forma remota con permisos que no corresponden	No utilizar canal seguro de comunicación para tareas de administración, como el empleo de herramientas que permitan la conexión con el servidor pero que dicha información es enviada en claro.	Hijacking. Secuestro del servidor por parte del atacante.	Establecimiento de un canal seguro.
En caso de caídas del sistema, se pueden llegar a tardar días o semanas en recuperar el sistema a su estado funcional.	No disponer de un plan de recuperación frente a catástrofes.	Los tipos de evento que pueden hacer fallar el sistema pueden ser: violación de seguridad, *malware*, caídas del servidor o cortes de electricidad.	Establecer un plan recuperación frente a riesgos, como la realización de copias de seguridad del sistema periódicas o la creación de un programa que permita instalar las dependencias necesarias para recuperar el sistema. Back-up
El servidor puede ser infectado con cualquier tipo de virus o *malware.*	No disponer de herramientas que protejan al servidor frente a virus o malware*,* tener desactualizado las bases de datos que controlan los tipos de *malware* o virus o no realizar un análisis del sistema en busca de estos problemas con la suficiente frecuencia.	Un usuario ajeno podría infectar el servidor con cualquier virus o *malware*, lo que puede derivar en que el atacante se haga con el control del sistema.	La instalación de un antivirus y otro tipo de dependencias que permitan realizar un análisis de los ficheros del sistema en busca de virus y *malware* y que dispongan de una base de datos actualizada.

Como se ha comentado, también se ha realizado un análisis estático de la aplicación. Como se comentó en la sección 2.4, este tipo de análisis toma como entrada el código fuente de la aplicación móvil y lo examina sin ejecutarlo para realizar un análisis de los posibles flujos de ejecución en busca de fuentes (código que lee información) y sumideros (canales que pueden filtrar datos personales fuera de la aplicación) [58].

Este análisis ha sido realizado con una herramienta externa llamada MobSF [91]. Esta herramienta es un *framework* automatizado que permite realizar análisis de *malware* y evaluación de seguridad de aplicaciones móviles (Android, iOS y Windows) y es capaz de realizar análisis estáticos y dinámicos. Además, cuando esta analiza el código, en los resultados que ofrece, indica una fuente de información (enlace a OWASP) al problema encontrado en el código.

El resultado de este análisis ha encontrado los siguientes problemas en el código de la aplicación:

- La aplicación registra información sensible. Los archivos de registro (*logs*) pueden almacenarse localmente cuando la aplicación está desconectada y enviarse al punto final una vez que la aplicación está conectada. Sin embargo, el registro de datos sensibles puede exponer los datos a atacantes o aplicaciones maliciosas, y también podría violar la confidencialidad del usuario.
- La aplicación copia información sensible al portapapeles. El portapapeles es accesible en todo el sistema y, por tanto, es compartido por todas las aplicaciones instaladas en el terminal. Este uso compartido puede ser aprovechado por aplicaciones maliciosas para obtener datos sensibles almacenados en el portapapeles.

Salvo estos problemas encontrados, el resultado del análisis fue exitoso, obteniendo un resultado de la seguridad de la aplicación de 100 sobre 100 y una puntuación media de CVSS (*Common Vulnerability Scoring System*) de 7’5 sobre 10. Esta última puntuación es un estándar que evalúa la seriedad de las vulnerabilidades de seguridad de los sistemas informáticos.

3.3 Necesidades de protección según el análisis de seguridad

En base al análisis anterior realizado, se marcan las siguientes necesidades para proteger los recursos manejados por el sistema:

- En base al primer riesgo común encontrado (comunicación insegura): Lo importante en una comunicación segura es que la información enviada no pueda ser reemplazada por otra información sin que la aplicación móvil o el servidor se den cuenta de ello. Para conseguir este fin, se necesita implementar un protocolo de intercambio de claves estándar para la comunicación entre la aplicación móvil y el servidor, como podría ser TLS, para crear un canal seguro por el que poder comunicarse. A este canal seguro se le puede añadir una capa de cifrado en paralelo. Esto implica cifrar los datos antes de ser transmitidos por este canal implementado.

- En base al segundo riesgo común encontrado (Autenticación insegura): Implementar una autenticación segura implica implementar medidas de multi-factor y/o una política de contraseñas robusta en la aplicación móvil, por lo que en el formulario de creación de una nueva cuenta que ofrece la aplicación, esta debe obligar al usuario a introducir una contraseña robusta y a establecer un segundo paso de autenticación. Actualmente, la aplicación guarda las credenciales sin cifrar en el terminal del usuario, por lo que se debe cifrar estas credenciales para evitar problemas de autenticación. En la medida de lo posible, se deberán realizar los controles de autenticación en el servidor.
- En base al tercer riesgo común encontrado (Autorización insegura): Se deben establecer medidas para realizar una autenticación segura, ya que este es el principal problema de la autorización insegura. En el caso de este sistema existen dos tipos de usuario, administrador o usuario. Que solo exista un tipo de administrador hace que este disponga de todos los permisos para modificar el sistema y acceder a información confidencial. Por ello, se debe fragmentar estos permisos de administrador y dividirlo en diferentes administradores que tengan diferentes permisos para hacer diferentes cambios sobre el sistema.
- En base al cuarto riesgo común encontrado (Baja calidad del código): Se debe realizar un análisis más exhaustivo del código mediante el uso de herramientas de análisis que permitan obtener resultados sobre la calidad del código. En el móvil se ha realizado un análisis estático, pero falta la realización de un análisis dinámico de esta. En el caso del código que implementa el *back-end* no se ha realizado ningún análisis, por lo que se debe de seguir la misma dinámica de análisis que con la aplicación móvil.
- En base al último riesgo común (Manipulación del código): Del lado de la aplicación, se deben implementar medidas para comprobar si la aplicación ha sido *rooteada*. Como los test se han hecho en un sistema operativo Android, hay distintas comprobaciones que se deben de realizar para esta comprobación: Comprobar si existen certificados OTA, comprobar si hay ficheros binarios SU, intentar el comando SU directamente (si este devuelve 0, la aplicación está rooteada) y comprobar si el fichero 'build.prop' contiene la línea 'ro.build.tags=test-keys' (indica que es una ROM no oficial). Del lado del servidor, se deberá comprobar en el *software* encargado de hacer las consultas que: No realiza consultas directamente con la información proporcionada por el usuario, valida los datos de entrada de usuario y usa sentencias preparadas con consultas parametrizadas.
- En base al riesgo adicional encontrado en la aplicación móvil (Uso inapropiado de la plataforma): Se debe revisar el código en busca de malas prácticas del código como uso de funcionalidades necesarias, uso innecesario de permisos, procesamiento de datos sensibles, uso de código no nativo. Se debe analizar la aplicación con un análisis dinámico empleando técnicas de generación de eventos que emulen la interacción del usuario con la aplicación en busca de fallas de seguridad.

4 Implementación de la fortificación

En este capítulo se va a describir la implementación realizada sobre la nueva arquitectura. Esta implementación toma como referencia el análisis realizado en el capítulo anterior y está dividida por cada componente en el que se ha trabajado: el servidor y la aplicación móvil. En este capítulo no se hablará de la implementación necesaria en el vehículo debido a que la implementación de medidas de seguridad en este componente queda fuera de este Trabajo de Fin de Grado. No obstante, sí que se ha podido observar que el análisis al servidor puede ser válido para tomar las medidas de seguridad a implementar en el módulo de comunicaciones del vehículo. Por ello, todas las medidas implementadas en el servidor podrían ser implementadas en el módulo de comunicaciones del vehículo.

El capítulo se ordena de la siguiente manera: Se explica las diferentes implementaciones llevadas a cabo para poder proteger los recursos manejados por cada componente y cómo se han llevado a cabo estas implementaciones. Alguna de ellas no interactúa directamente con los recursos, pero son necesarias para el proyecto.

4.1 Fortificación del servidor

El servidor es la mayor fuente de problemas de seguridad que tiene la arquitectura. Esto es debido a que es la encargada de validar cualquier tipo de autenticación y autorización realizada por cualquier tipo de cliente entrante. Tiene que proteger la información enviada tanto a la aplicación móvil como al vehículo, ya que esta información es crítica y sensible.

La conexión entre el servidor y el vehículo permite enviar información del estado del vehículo, como el estado de la batería, la localización, mensajes de error, etc. Esta información viaja desde el vehículo hacia el servidor, pero en dirección opuesta el servidor envía órdenes al vehículo, tales como: ordenar al vehículo que avance o se pare, ordenar una ruta a seguir o hacer que el vehículo cambie de modo manual a modo autónomo. Más información sobre los mensajes intercambiados entre el servidor y en el vehículo pueden encontrarse en el proyecto de fin de grado de I. Royuela [12].

La comunicación entre el servidor y la aplicación móvil permite el envío de los recursos almacenados en la base de datos, tales como el número de vehículos disponibles o los aparcamientos más cercanos a la localización del usuario. Pero también permite autenticar al usuario que está intentando iniciar sesión en la aplicación móvil. Más información sobre los mensajes intercambiados entre la aplicación móvil y el servidor en el proyecto de fin de grado de A. Mazaira [87].

4.1.1 Decisiones tomadas

En base a las comunicaciones resultantes entre el servidor y la aplicación móvil, y en base al análisis realizado en el capítulo anterior, se han tomado las siguientes decisiones:

- Implementar un *firewall*, o cortafuegos, que proteja al servidor de cualquier conexión no necesaria. El objetivo de un cortafuegos es proteger al servidor

de ataques realizados desde Internet. Este limita el tráfico entrante y/o saliente del servidor.

- Implementar una entrada segura al servidor mediante la configuración del protocolo SSH. Mediante este protocolo, el administrador del servidor podrá acceder al servidor y controlar y modificar el sistema de *back-end* de una forma segura.
- Desarrollo de un *script* de seguridad que permita reinstalar todas las dependencias ya instaladas junto a sus correspondientes ficheros de configuración.
- Implementar el protocolo TLS en las comunicaciones MQTT que existen entre el servidor y la aplicación móvil. Para ello, primero se modificará el *software back-end*, para después modificar el *software* de la aplicación móvil.
- Instalación de dependencias necesarias para monitorizar los registros del sistema en busca de actividad maliciosa y de vulnerabilidades, y para escanear *rootkits* (malware que permite el acceso no autorizado a un equipo ajeno [92]), *exploits* (fragmentos de código que permiten aprovechar una vulnerabilidad del sistema [93]) o puertas traseras (*malware* que permite el acceso a usuarios remotos ajenos [94]) del sistema.

4.1.2 Firewall

Un cortafuegos, o *firewall*, es un sistema de seguridad que filtra el flujo de tráfico entrante y saliente de una red en base a unas reglas. Como se puede ver en la Figura 27, el cortafuegos suele estar colocado en los límites de una red e Internet, ya que permite un mejor control del tráfico. En el caso particular de este proyecto, el cortafuegos se ubicará dentro del servidor, como un *software*, pero se podría dedicar un único ordenador a esta causa, colocándolo entre el servidor del proyecto TwizyLine e Internet.

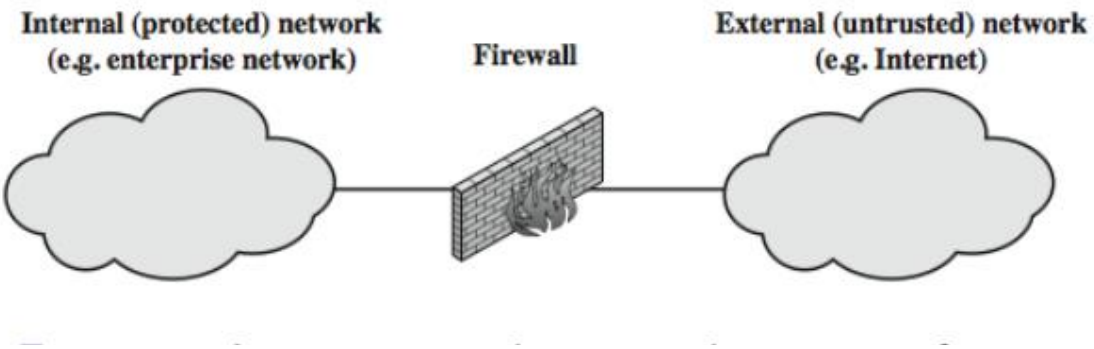

Figura 27. Arquitectura de una red con cortafuegos [95].

Como se ha comentado, en la arquitectura actual una Raspberry hace de servidor. Sobre esta Raspberry corre el sistema operativo Raspbian, que tal y como se comentó anteriormente, está basado en Debian, una distribución del sistema operativo GNU/Linux. Linux proporciona una herramienta llamada “iptables”. Esta herramienta ofrece un cortafuegos a nivel de red, la cual permite filtrar los paquetes de tráfico entrante y salientes mediante el uso de tablas. En estas tablas se formalizan las reglas de filtrado. Los conceptos principales para entender cómo funciona son [40]:

- Tabla: Es un contenedor de cadenas asociado a un determinado tipo de protocolo, como ipv4, ipv6, etc.
- Cadena: Es un contenedor de reglas asociado a una tabla. Dependiendo de en qué lugar del flujo esté el paquete, se aplicará una cadena u otra. El flujo de los paquetes dentro de un sistema Linux puede verse en la Figura 28. Cada cadena es responsable de una tarea [95]:
    - *Prerouting*: Esta cadena decide lo que ocurre con un paquete antes de que llegue a la interfaz de red.
    - *Input*: Cadena usada para manejar los paquetes entrantes.
    - *Forward*: Cadena responsable del reenvío de paquetes.
    - *Output*: Cadena que puede limitar los paquetes salientes después de que hayan sido creados o procesados.
    - *Postrouting*: Cadena encargada de decidir qué ocurre con el paquete antes de que abandone la tarjeta de red y se haya tomado la decisión de enrutamiento.
- Reglas: Contienen acciones que se realizan sobre los paquetes en función de sus características, como la dirección IP de origen o destino, el puerto de origen o de destino, etc.

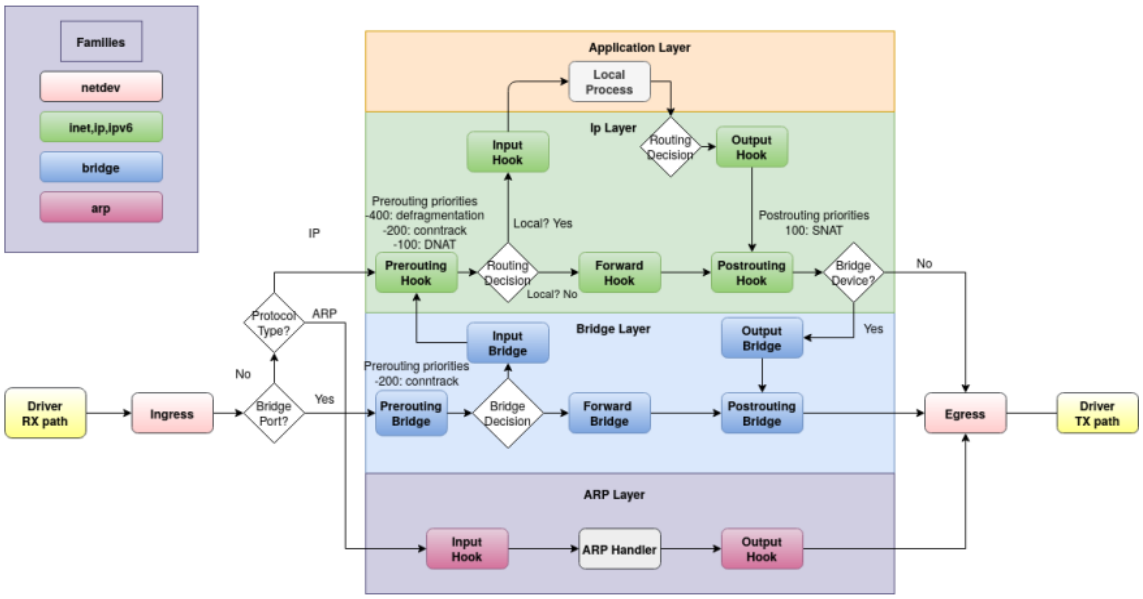

Figura 28. Flujo de los paquetes en el software de red de GNU/Linux [95].

Para añadir nuevas reglas en el cortafuegos mediante el uso de iptables, hay que usar la línea de comandos llamando al comando iptables y especificando la regla que queremos especificar y sobre qué cadena va a actuar esa regla. En el caso particular de este proyecto, al ser el cortafuegos de un servidor, la mayoría de las conexiones siempre van a empezar desde Internet hacia este, por lo que lo primero que se haría es establecer una política de descarte de paquetes de entrada por defecto y después establecer una regla que acepte como paquetes de entrada aquellos paquetes de conexiones establecidas o relacionadas, para que el servidor pueda comunicarse hacia fuera. Para introducir estas dos reglas, se deberían de ejecutar los siguientes comandos:

```
iptables -P INPUT DROP
```

```
iptables -A INPUT -m conntrack -ctsate ESTABLISHED, RELATED, -j
ACCEPT
```

Como se ha visto, la sintaxis para trabajar con iptables es algo compleja de entender, por lo que para este proyecto se va a usar una herramienta que facilite el trabajo de implementación del cortafuegos. Esta herramienta se llama UFW (Uncomplicated Firewall) [96]. UFW es una aplicación que facilita la introducción de reglas en iptables, es decir, es una interfaz para iptables. Esta herramienta permite realizar de una forma sencilla la configuración del cortafuegos, pero sin reducir la seguridad implementada.

La sintaxis para ejecutar un comando con UFW es la siguiente:

```
ufw action flow port|service
```

, donde:

- *action*: indica si acepta, deniega, rechaza o limita el paquete mediante el uso de *allow*, *deny*, *reject* o *limit*, respectivamente.
- *flow*: Aquí se especifica el tipo de cadena, pero solo acepta entrante o saliente (*incoming* o *outcoming*). Para afectar a otro tipo de cadenas, hay que especificarlo dentro del fichero de configuración de UFW, *before.rule*s.
- *port|service*: Puerto o servicio (tcp o 53, http o 80, etc) al que aplicar la acción y cadena.

Como se ha comentado antes, una de las primeras reglas a introducir en el cortafuegos es establecer una política por defecto de descarte para todos aquellos paquetes de entrada, pero permitiendo entrar a aquellos paquetes que tengan que ver con conexiones establecidas por el servidor. Para introducir esta regla se introducirá el siguiente comando:

```
ufw default deny incoming
```

Las únicas conexiones que se quieren permitir hacia el servidor son las provenientes del protocolo MQTT y el protocolo SSH. El primer protocolo es empleado por la aplicación móvil, y la segundo, por el administrador para conectarse de forma remota al servidor. Por ello, el único tráfico de entrada que se va a permitir es el tráfico proveniente de los puertos 8883 y 22445, que son los puertos que usan el protocolo MQTT sobre TLS, y el protocolo SSH (se explicará más adelante el porqué de estos puertos). Además, para la fase de pruebas, se ha habilitado el puerto 1883, que es el puerto empleado por defecto por MQTT.

Para aplicar esta configuración expuesta se introducirán los siguientes comandos:

```
ufw allow in 22445
```

```
ufw allow in 8883
```

```
ufw allow in 1883
```

El resultado de esta configuración es la siguiente:

Figura 29. Estado del firewall implementado con la herramienta ufw.

Además de esta configuración, se ha configurado también el fichero de configuración 'before.rules' de UFW. Este fichero permite añadir al *firewall* reglas más específicas que afectan al *loopback*, al *ping* y al DHCP (Protocolo de configuración dinámica de host) de las que se pueden añadir por línea de comandos. En este caso, este fichero será ejecutado por UFW antes de las normas introducidas por terminal. Existen un fichero con el mismo objetivo, pero que se ejecuta después de lanzar las reglas introducidas por línea de comandos.

El fichero 'before.rules' se ha modificado para eliminar cualquier paquete ICMP (*Internet Control Message Protocol*) entrante. Estos paquetes son recibidos cuando se ejecuta el comando *ping* hacia nuestro servidor, por lo que, denegando estos paquetes, se podrán evitar ataques de denegación de servicio. Esta modificación de la configuración puede verse en la Figura 30.

```
# ok icmp codes for INPUT
-A ufw-before-input -p icmp --icmp-type destination-unreachable -j DROP
-A ufw-before-input -p icmp --icmp-type time-exceeded -j DROP
-A ufw-before-input -p icmp --icmp-type parameter-problem -j DROP
-A ufw-before-input -p icmp --icmp-type echo-request -j DROP
```

Figura 30. Contendio del fichero before.rules

También resulta interesante habilitar el registro de datos para comprobar qué conexiones se han realizado con el servidor y verificar si las reglas funcionan correctamente. El contenido de este registro de datos se guarda en el fichero *ufw.log*. Para habilitar el registro de datos se introducirá el siguiente comando:

```
ufw logging on
```

Como se ha comentado anteriormente, UFW es una interfaz de iptables, por lo que cuando se introducen todas las reglas anteriores y se lanza UFW, las tablas de iptables son modificadas por UFW. En el Anexo I se puede encontrar como quedaría configurado iptables con la configuración realizada.

4.1.3 SSH

SSH es un protocolo de comunicación que permite una comunicación cifrada entre dos ordenadores. Mediante esta comunicación, un usuario puede acceder y controlar el ordenador al que se está accediendo de una manera segura. SSH usa el protocolo de transporte TCP y usa un protocolo *handshake* parecido al de TLS para negociar los parámetros de la comunicación. Durante esta negociación, las dos partes acuerdan un algoritmo de cifrado simétrico y generan la clave de cifrado a usar. Aparte de usar un algoritmo de encriptación estándar, este protocolo también emplea un algoritmo de *hash* estándar [97].

El puerto que usa por defecto SSH es el puerto 22, pero este puerto ha sido modificado a un puerto muchísimo más alto por motivos de seguridad. Este puerto es pertenece a *los well-known ports* y puede ser una entrada de acceso a personas no autorizadas o una puerta de entrada por la que recibir ataques de fuerza bruta. Como se ha visto en el apartado anterior, se ha dejado abierto el puerto 22445 para las comunicaciones SSH. El beneficio que tiene reconfigurar el puerto a uno con un número tan alto es que la mayoría de escaneos de puertos se realizan contra los *well-known ports.* Estos puertos van del 0 al 1024. Escanear todos los puertos de un servidor es algo que requiere de mucho tiempo, y normalmente, este tipo de escaneo es algo que se quiere realizar sin llamar la atención. Si un escaneo dura demasiado o no se realiza en una red local, puede ser detectado por el *firewall* o por un pico en el tráfico de la red, por lo que los *hackers* para asegurarse la eficacia de los escaneos evitan que perduren mucho en el tiempo. Se puede consultar un escaneo de los puertos del servidor en el Anexo II.

SSH dispone de dos ficheros de configuración. Uno es para configurar las conexiones SSH cliente, es decir, si el servidor inicia la comunicación SSH. El segundo es para configurar las conexiones SSH servidoras, es decir, en el caso de que un usuario remoto inicie la comunicación SSH hacia el servidor. Este segundo fichero de configuración, llamado *sshd_config*, es el que hay que modificar, ya que solo es necesario configurar SSH para que un usuario remoto se conecte al servidor.

Las decisiones tomadas para mejorar la seguridad de esta herramienta en base a la utilidad que se le quiere sacar son las siguientes:

- Cambiar el puerto de acceso a uno mucho más grande (22445).
- Bloquear que el usuario *root* pueda iniciar una sesión SSH de manera remota. Esto obliga al atacante a tener que acertar el usuario que tiene permisos para iniciar una sesión SSH e impide que el atacante logre la contraseña del usuario *root* mediante un ataque de fuerza bruta al servidor SSH.
- Limitar el tiempo que tiene el usuario remoto para introducir la contraseña del usuario con el que desea iniciar sesión (30 segundos); limitar el número de intentos para introducir la contraseña (3 intentos); limitar el número de sesiones abiertas permitidas por usuario remoto (3 sesiones). Todas estas medidas evitan que cualquier atacante pueda probar de manera ilimitada pares de usuario y contraseña para adivinar unas credenciales que permitan abrir una sesión SSH con el servidor.
- Permitir que solo un usuario con permisos limitados y que cuente con una contraseña robusta pueda iniciar una sesión SSH con el servidor (usuario 'remoto').

Tras tomar estas medidas, el fichero de configuración de SSH debería de verse de la siguiente manera:

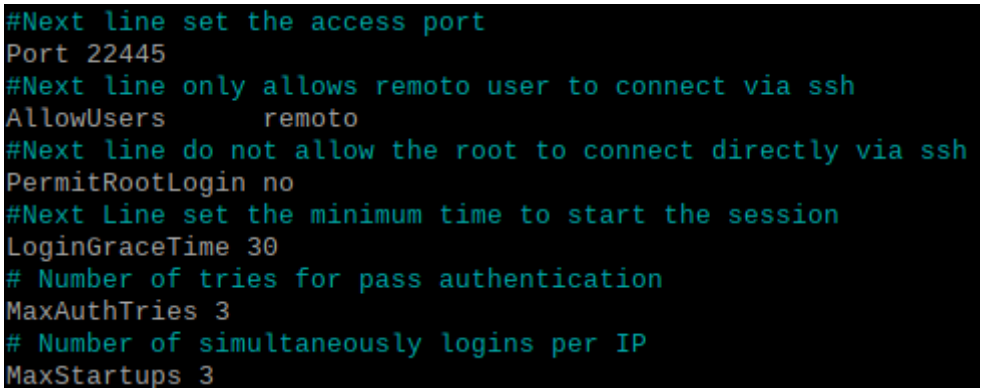

Figura 31.Contenido del fichero sshd_config después de su configuración.

Para aplicar los cambios establecidos en el fichero de configuración se tendrá que reiniciar el servicio SSH con el siguiente comando:

```
systemctl restart ssh
```

Ahora que está completada la configuración, si el administrador de la red quiere acceder de forma remota al servidor, tendrá que iniciar la sesión con el usuario remoto y especificando el puerto 22445. El comando que emplearía el administrador sería el siguiente:

```
ssh remoto@<dirIpServidor> -p 22445
```

, donde <dirIpServidor> es la dirección IP del servidor.

Se ha establecido una contraseña considerada segura, aunque no se ha establecido ningún mecanismo que fuerce al usuario a introducir una nueva contraseña fuerte en el caso de que cambie la anterior. Esta es una política que debe ser implementada en revisiones futuras de la seguridad del servidor.

4.1.4 Back Up Script

Se conoce como fichero de *back up* a aquel fichero que cuando se ejecuta, guarda una copia de seguridad del sistema o una copia del sistema de directorios especificado. Este tipo de fichero puede desarrollarse usando *Shell script*. Un *Shell script* es un fichero que contiene una secuencia comandos que entiende un sistema operativo basado en Linux [98]. Pero un fichero de *back up* no solo puede llegar a guardar una copia de seguridad, si no que puede recuperar el estado que tenía el sistema en un momento determinado, no solo a nivel de ficheros, si no a nivel de dependencias instaladas que hacen que el sistema funcione correctamente.

En este caso particular, se ha desarrollado un *script* que permita instalar todas las dependencias y ficheros de configuración necesarios para el funcionamiento de este proyecto. Este fichero permite la migración del sistema a otro dispositivo que emplee un sistema operativo basado en una distribución de Linux, con el fin de disminuir el RTO (*Recovery Time Objective*). Realizar esta clase de programas permite disminuir el tiempo que tarda un sistema en volver a funcionar después de un evento inesperado como el ataque de un *malware*, una violación o pérdida de datos, una caída del servidor o un corte de electricidad. La realización de esta clase de programas entra dentro de la elaboración de un plan de recuperación de desastres [99].

Este *script* realiza lo siguiente:

- Actualiza la lista de paquetes y actualiza los aquellos paquetes instalados que puedan ser actualizados.
- Instala las dependencias de la herramienta 'mosquitto'. Esta herramienta permite instalar el *broker* MQTT en el sistema. Además, se crea un fichero de configuración (default.conf) que es leído por el *broker*. Este fichero indica al *broker* que no permita conexiones anónimas y que escuche conexiones en el puerto 1883. Además, se crean las credenciales del usuario por defecto y se cifra la contraseña de este usuario, almacenando esta información en el fichero 'passwd'. Además, se indica al *broker* la localización de este fichero para que este solo acepte conexiones de este usuario por defecto.
- Instala el cliente de mosquitto. Esta herramienta permite lanzar ordenes como cliente publicador y como cliente suscriptor. No es necesaria para el funcionamiento del servidor, pero sí para la realización de pruebas.
- Se crea el usuario 'remoto' con una contraseña robusta. El fin de este usuario es permitir una puerta de entrada segura para el administrador a través del canal seguro SSH.
- Se crea y se modifica el fichero 'default.conf'. Este fichero es llamado por el fichero de configuración del servidor SSH 'sshd_config. En este fichero creado, se implementan las medidas de seguridad mencionadas en la anterior sección, como el puerto a usar (22445).
- Se instala la aplicación 'fail2ban', cuyo funcionamiento se explicará más adelante. Se realizan cambios en su fichero de configuración 'defaults-debian.conf' para definir el tiempo de prohibición y el número máximo de intentos (Información más detallada en la sección 4.1.6).
- Se instala la herramienta 'rkhunter' y se lanza la orden correspondiente para actualizar su base de datos (Información más detallada en la sección 4.1.6).
- Se instala la aplicación 'debsecan' (Información más detallada en la sección 4.1.6).
- Se instala la herramienta 'UFW' y se lanzan las órdenes que deniegan el tráfico entrante pero aceptan el tráfico proveniente de los protocolos MQTT, MQTT con TLS y SSH.

Todos los anteriores dependencias y ficheros de configuración comentados son instalados y modificados, respectivamente, de manera automática tras lanzar el fichero de *backup* con permisos de superusuario. Esto permite disminuir en gran medida el RTO, ya que con una línea en el terminal del servidor se puede recuperar una gran medida de las funcionalidades del sistema.

Este fichero está almacenado en el almacén de repositorios GitHub perteneciente al proyecto y es un fichero cambiante. Esto quiere decir que cada cambio que se haga en la configuración del servidor se deberá ver reflejado en este *script*. También es aconsejable crear diferentes *scripts* con diferentes configuraciones para poder instalar un modo degradado del sistema u otras configuraciones necesarias.

El contenido del fichero puede verse en el Anexo III.

4.1.5 MQTT over TLS

Para fortificar las comunicaciones realizadas con el protocolo MQTT, hay que implementar el protocolo TLS. MQTT ofrece esta opción, por lo que lo primero que habrá que hacer es cambiar la configuración del bróker MQTT, ya que cualquier mensaje publicado por un cliente, tiene que pasar por el bróker.

En el proyecto de TwizyLine, el despliegue del bróker y cliente MQTT se realiza mediante el uso de herramientas que ofrece la fundación Eclipse [100]. Para más información sobre las herramientas empleadas para desplegar el bróker, consultar el proyecto de fin de grado de S. Pilar [74].

Para poder desplegar MQTT sobre TLS se necesitan los siguientes componentes:

- Un certificado CA y firmarlo con una clave creada para la CA (ca.crt).
- Un certificado para el bróker firmado por el certificado de la CA (broker.crt).
- Una clave para el bróker con el que se firma la petición de certificado de este (broker.key).
- Un certificado para el cliente firmado por el certificado de la CA (client.crt).
- Una clave para el cliente con el que se firma la petición de certificado de este (client.key).

Para poder generar los certificados, las peticiones de certificado y las claves, se ha hecho uso de la herramienta de línea de comandos ofrecida por Linux *openssl* [101]. Esta herramienta implementa los protocolos TLS y los estándares de criptografía relacionados con este. Para la creación de las claves y de los certificados se han seguido los siguientes pasos:

1. Lo primero es crear nuestro propio certificado de CA (ca.crt) y su clave (ca.key). Esto se hará, primero, creando una clave de una longitud de 2048 bits con un algoritmo de clave pública y con encriptación, para después, con esta clave, firmar el certificado de nuestra CA. El certificado es creado con una duración de un año. lo Normal es que los certificados se renueven entre 1 a 5 años. Para ello, se introducen los siguientes comandos:

```
openssl genrsa –des3 –out ca.key 2048
```

```
openssl req –new –x509 –days 365 –key ca.key –out ca.crt
```

Cuando el segundo comando sea lanzado, se deberá de rellenar la información solicitada de la siguiente manera (Es importante que cada vez que openssl nos solicite rellenar un formulario para un certificado, la información de los campos sea lo más distinta posible):

```
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:Valladolid
Locality Name (eg, city) []:Valladolid
Organization Name (eg, company) [Internet Widgits Pty Ltd]:CAmaster
Organizational Unit Name (eg, section) []:TEST
Common Name (e.g. server FQDN or YOUR name) []:CA
Email Address []:piCA@uva.es
```

2. Lo siguiente será crear una clave secreta sin encriptación, de la misma manera que en el paso 1, para que sea usada por el servidor:

```
openssl genrsa –out server.key 2048
```

3. Después, se generará la solicitud de firma para enviársela a nuestra CA:

```
openssl req –new –out server.csr –key server.key
```

Se nos pedirá rellenar una un formulario como en el paso 1. Es importante que el campo '*Common Name*' tenga el valor de la dirección IP del servidor (en este caso es 10.0.103.46) donde está alojado el *broker* MQTT:

```
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:Burgos
Locality Name (eg, city) []:Aranda
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Server-cert
Organizational Unit Name (eg, section) []:test
Common Name (e.g. server FQDN or YOUR name) []:10.0.103.46
Email Address []:pi@uva.es

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

4. Esta solicitud será enviada a la CA para que la firme y genere un certificado de una duración de 1 año para el servidor de la siguiente manera:

```
openssl x509 –req –in server.csr –CA ca.crt –Cakey –CAcreateserial
–out server.crt –days 365
```

5. Se dará permisos al servicio mosquitto para que pueda acceder a estos ficheros de la manera:

```
chown mosquitto:root server.key
```

6. Con esta configuración tendríamos los siguientes ficheros, donde los ficheros necesarios para configurar el *broker* son 'ca.crt', 'server.crt' y 'server.key':

```
-rw-r--r-- 1 root      root 1415 Jul 21 12:06 ca.crt
-rw------- 1 root      root 1743 Jul 21 12:05 ca.key
-rw-r--r-- 1 root      root   41 Jul 21 12:29 ca.srl
-rw-r--r-- 1 root      root 1294 Jul 21 12:29 server.crt
-rw-r--r-- 1 root      root 1045 Jul 21 12:17 server.csr
-rw------- 1 mosquitto root 1679 Jul 21 12:15 server.key
```

Con estos ficheros, ya se puede configurar el bróker. Para ello, se tiene que modificar el fichero de configuración de *mosquitto* (mosquitto.conf) para que emplee nuestro propio certificado CA, el certificado y clave del bróker, la versión de TLS especificada (Se ha decidido usar esta versión debido a que es la que se aconsejaba en el manual de mosquitto), el puerto 8883, que es el puerto que emplea MQTT sobre TLS y la opción '*require_certificate*' a *true* para que no acepte conexiones de clientes que no presenten un certificado válido. El fichero de configuración resultante se puede ver en la Figura 32.

```
# Port to use for the default listener.
listener 8883

# Certificates
cafile ca.crt
keyfile server.key
certfile server.crt
tls_version tlsv1.2

# This means that the client that want to connect to the broker must provide a valid certificate
require_certificate true
```

Figura 32. Contenido del fichero de configuración del bróker mosquitto.conf

Ahora que está configurado el *broker*, hay que generar la clave secreta del cliente y el certificado del cliente. Para ello, se seguirá con el siguiente procedimiento:

1. Lo primero es generar una clave secreta para el cliente con encriptación, de la misma manera que para el servidor:

```
openssl genrsa –des3 –out client.key 2048
```

2. Después se generará la solicitud de firma para enviársela a nuestra CA, de la misma manera que con el servidor. Es estrictamente necesario que los certificados del cliente estén firmados por la misma CA que se especifica en el fichero de configuración del *broker*:

```
openssl req –out client.csr –key client.key -new
```

   En este apartado, no es necesario hacer hincapié en el campo '*Common name*', aunque se ha rellenado el formulario de solicitud de la siguiente manera:

```
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:Teruel
Locality Name (eg, city) []:Teruel
Organization Name (eg, company) [Internet Widgits Pty Ltd]:GCO
Organizational Unit Name (eg, section) []:TFG
Common Name (e.g. server FQDN or YOUR name) []:elCliente
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

3. Se enviará la solicitud de firma creada a nuestra CA para generar el certificado del cliente, de la misma manera que con el servidor:

```
openssl x509 –req –in client.csr –CA ca.crt –Cakey ca.key –
CAcreateserial –out client.crt –days 365
```

4. Los ficheros resultantes serán los siguientes, aunque solo se necesitan el fichero 'client.key' y 'client.crt':

```
-rw-r--r-- 1 root      root 1245 Jul 21 12:46 client.crt
-rw-r--r-- 1 root      root  993 Jul 21 12:42 client.csr
-rw------- 1 root      root 1743 Jul 21 12:40 client.key
```

Con la configuración realizada en el bróker y teniendo la clave y certificado del cliente, para poder suscribirse al tópico "prueba" empleando el cliente que ofrece mosquitto sería necesario usar el siguiente comando:

```
mosquitto_sub –t “prueba” --cafile ca.crt --key client.key --cert client.crt
```

Como el sistema *back-end* está desarrollado en el lenguaje Python, hay que modificar los ficheros encargados de gestionar el sistema de *back-end* para que MQTT implemente TLS empleando los ficheros anteriormente descritos. Para ello, cuando el cliente quiere establecer conexión con el bróker, tiene que indicar qué certificados, clave y versión va a usar el cliente. Esta configuración consiste en especificar la ruta a los ficheros ca.crt, client.key y client.crt, la versión de TLS a emplear y la contraseña que descifra la clave del cliente. Un ejemplo de esta configuración dentro de un fichero con Python se puede ver en la Figura 33.

```
server.tls_set(ca_certs=CA_CERT_PATH, certfile=CLIENT_CERT_PATH,
                       keyfile=CLIENT_KEY_PATH, cert_reqs=ssl.CERT_REQUIRED,
                       tls_version=ssl.PROTOCOL_TLSv1_2, ciphers=None,
                       keyfile_password=CLIENT_PASS)
    server.connect(host=BROKER_ENDPOINT,port=8883)
```

Figura 33. Ejemplo del código empleado para fortificar las comunicaciones MQTT.

4.1.5.1 Gestor de secretos

El código encargado del sistema del *back-end* está encargado de establecer las conexiones con el bróker MQTT y con la base de datos. Para establecer estas conexiones, aparte del uso de certificados y claves necesarias para implementar TLS, es necesario identificarse con un usuario y una contraseña frente al bróker. Estas credenciales están especificadas en claro en el código, por lo que se trata de una práctica poco segura. A este tipo de práctica se le conoce como ‘*Hardcoded Passwords*’.

Este tipo de prácticas puede utilizarse en dispositivos, aplicaciones y sistemas, lo que ayuda a simplificar la configuración a escala, pero al mismo tiempo supone un riesgo considerable de seguridad. Para reducir la exposición de estas contraseñas es recomendable introducir una solución de gestión de contraseñas de aplicación de terceros que obligue solicitar el uso de la contraseña desde una caja fuerte de contraseñas centralizadas. Esto quiere decir que desde el programa se solicite acceso a la caja fuerte de contraseñas para poder acceder a la contraseña que iría en claro dentro del código del programa [102].

Para este caso, se ha decidido usar la librería de Python ‘keyring’ como gestor de secretos. Esta librería emplea el servicio de llaveros del sistema de Python y permite guardar cualquier contenido de tipo texto de manera protegida. Este servicio requiere de introducir una contraseña antes de ejecutar la línea de código que llama al servicio.

La forma de guardar esta información es mediante el uso de dos palabras clave que identifiquen la información a guardar. A continuación, un ejemplo de las líneas de código Python necesarias para almacenar y recoger la contraseña “1234”:

```
import keyring

keyring.set_password(“usuario”, “samuel”, “1234”)

keyring.get_password(“usuario”, “samuel")
```

4.1.6 Otras dependencias instaladas

A parte de implementar un cortafuegos, se ha decidido instalar otro tipo de dependencias que permitan monitorizar el estado del sistema. Estas dependencias consisten en un detector de *rootkits*, *exploits* y puertas traseras, una aplicación que previene de ataques de fuerza bruta y ataques de denegación de servicio, y una aplicación que evalúa la seguridad del sistema y muestra las vulnerabilidades más comunes. Estas dependencias son:

- Fail2ban: Esta aplicación escanea los archivos de registro y prohíbe el acceso a aquellas IPs que muestren signos maliciosos, como demasiados fallos al intentar introducir la contraseña. Esta aplicación permite frenar ataques de denegación de servicio y fuerza bruta [103]. En este caso particular, ha sido configurado para que el número máximo de intentos para introducir correctamente la contraseña sea 3 y que el tiempo de baneo, una vez se supere el número de intentos sea de 30 minutos.
- rkhunter: Es una herramienta que detecta la presencia de *rootkits*, *exploits* y puertas traseras. Puede avisar al administrador vía e-mail de cualquier detección encontrada [104]. En este caso, está configurado para lanzar está búsqueda diariamente y para actualizar la base de datos semanalmente.
- debsecan: Es una aplicación que analiza los paquetes instalados en el servidor e informa de las vulnerabilidades encontradas. Esta aplicación descarga la información sobre las vulnerabilidades de Internet y puede avisar vía e-mail cuando esta descubre nuevas vulnerabilidades en el sistema [105]. Actualmente no está implementada ningún tipo de herramienta que lance automáticamente esta aplicación con una cierta frecuencia, pero se recomienda implementar un programador de tareas, como *cron* [106], que permita lanzar esta aplicación diariamente en busca de vulnerabilidades.

4.2 Fortificación de la aplicación móvil

La aplicación va a ser el intermediario entre los usuarios y el servidor. Según se ha visto en la arquitectura del proyecto TwizyLine, esta aplicación se comunica con el servidor empleando el protocolo MQTT. Ahora que el sistema *back-end* implementa MQTT sobre TLS, también lo deberá de implementar la aplicación para que pueda existir una comunicación entre estos dos componentes.

La actual aplicación está desarrollada empleando el *framework* Flutter [16] y empleando el lenguaje de programación Dart [17]. Además, se hace uso del IDE Android Studio para el desarrollo de la aplicación. Más información sobre la aplicación en el proyecto de fin de grado de A. Mazaira [87].

Lo interesante de Flutter es que se pueden instalar y usar *plugins* para facilitar la programación. En este caso, para desarrollar la parte de la comunicación MQTT, se hizo uso del *plugin* mqtt_client. Este es un *plugin* certificado y dispone de una amplia documentación [107]. Además, todo el código encargado de manejar los eventos producidos por la comunicación MQTT están recogidos en un mismo fichero, llamado mqtt.dart, por lo que para implementar MQTT sobre TLS, hay que modificar este fichero.

Como en la anterior sección se han creado los certificados de la CA y del cliente y la clave del cliente, falta el desarrollo del código para que la aplicación implemente TLS empleando estos ficheros.

Lo primero que hay que hacer, es añadir unas líneas de código al fichero pubspec.yaml, para indicar los nuevos recursos que se van a usar. Estos recursos son los certificados y la clave del cliente.

```
# The following section is specific to Flutter.
flutter:
  assets:
    - assets/certs/ca.crt
    - assets/certs/client.crt
    - assets/certs/client.key
```

Después, en el fichero mqtt.dart, hay que indicar al cliente que tiene que comunicarse con el *bróker* MQTT usando el puerto 8883:

```
final client = MqttServerClient.withPort('192.168.0.110', '', 8883);
```

También es necesario especificar que el cliente va a usar una conexión segura.

```
client.secure = true;
```

Por último, se cargará en diferentes variables el contenido de los ficheros que almacenan los certificados y la clave, y se añadirán a la clase 'SecurityContext'. Esta clase está incluida con el *plugin* mqtt_client y es la encargada de contener los certificados en los que se debe confiar al realizar una conexión segura con el servidor. Esta clase también permite especificar la configuración de TLS que se va a implementar. A continuación, una captura del código con lo anteriormente comentado:

```
ByteData rootCA = await rootBundle.load('assets/certs/ca.crt');
ByteData clientCert = await rootBundle.load('assets/certs/client.crt');
ByteData clientKey = await rootBundle.load('assets/certs/client.key');

SecurityContext context = SecurityContext.defaultContext;
context.setClientAuthoritiesBytes(rootCA.buffer.asUint8List());
context.useCertificateChainBytes(clientCert.buffer.asUint8List());
context.usePrivateKeyBytes(clientKey.buffer.asUint8List(), password: 'cliente');
```

Con esta implementación, la aplicación estaría comunicándose con el servidor empleando MQTT sobre TLS. Un ejemplo de una traza de esta comunicación se comenta en el capítulo Resultados.

5 Resultados

En este capítulo se van a comentar los siguientes resultados obtenidos:

- Análisis del antes y después de la implementación de TLS al protocolo MQTT de una comunicación cliente-*broker*. En este análisis se observará como la comunicación se cifra con éxito y otras particularidades.
- Análisis de la implementación TLS a la comunicación entre el *broker* MQTT y la aplicación móvil. En este análisis se comentará el protocolo *handshake* y el protocolo de la capa de registro, en comparación con lo visto en el estado del arte.
- Se comentarán los problemas y dificultades encontrados durante la realización de este Trabajo de Fin de Grado.

5.1 Análisis antes vs después de implementar TLS

El escenario del análisis es el siguiente:

- El *broker* se ejecuta en una Raspberry Pi 3 modelo B+.
- En un ordenador con Windows, se lanzan dos terminales. En uno se ejecuta un cliente MQTT suscriptor y en otro terminal se ejecuta un cliente MQTT publicador.
- Se publica el mensaje “Mensaje de prueba” en el tópico “prueba”.
- La traza wireshark es capturada desde el ordenador donde es lanzado el cliente MQTT. Esta captura empieza cuando el cliente MQTT publicador publica el mensaje anterior.
- Tanto el ordenador como la Raspberry están en la misma red, por lo que los retardos de red se supondrán como despreciables.

En la Figura 34 se puede observar el intercambio de tramas cuando no se implementa TLS, y en la Figura 35 se puede observar el intercambio de tramas cuando sí se implementa TLS.

En el primer caso, si se abre el mensaje ‘*Publish Message*’ obtenemos el siguiente contenido:

```
Topic: prueba
Message: 4d656e73616a65206465207072756261
```

Este contenido nos indica en claro que el tópico escogido es “prueba” y el mensaje en ASCII es “Mensaje de prueba”. Pero si lo comparamos con la traza donde se emplea TLS, todos los datos de aplicación están cifrados y resulta imposible descifrar los mensajes intercambiados:

```
Encrypted Application Data: 341215df2162efced9f335df36831fa87345003a6c87cf20fb4e06aa6bf658b34ce8d95c…
```

1.548107	192.168.0.104	192.168.0.110	TCP	54	64541 → 1883 [ACK] Seq=1 Ack=1 Win=131328 Len=0
1.548287	192.168.0.104	192.168.0.110	MQTT	82	Connect Command
1.549981	192.168.0.110	192.168.0.104	TCP	54	1883 → 64541 [ACK] Seq=1 Ack=29 Win=64256 Len=0
1.551739	192.168.0.110	192.168.0.104	MQTT	58	Connect Ack
1.551912	192.168.0.104	192.168.0.110	MQTT	81	Publish Message [prueba]
1.552010	192.168.0.104	192.168.0.110	MQTT	56	Disconnect Req
1.553328	192.168.0.110	192.168.0.104	TCP	54	1883 → 64541 [ACK] Seq=5 Ack=56 Win=64256 Len=0
1.554388	192.168.0.110	192.168.0.104	TCP	54	1883 → 64541 [FIN, ACK] Seq=5 Ack=59 Win=64256 Len=0
1.554425	192.168.0.104	192.168.0.110	TCP	54	64541 → 1883 [ACK] Seq=59 Ack=6 Win=131328 Len=0

Figura 34.. Traza de una captura en la que un cliente publica un mensaje usando el protocolo MQTT.

2.356882	192.168.0.104	192.168.0.110	TLSv1.3	367	Client Hello
2.359428	192.168.0.110	192.168.0.104	TCP	54	8883 → 64528 [ACK] Seq=1 Ack=314 Win=64128 Len=0
2.383424	192.168.0.110	192.168.0.104	TLSv1.3	1514	Server Hello, Change Cipher Spec, Application Data, Application Data
2.383424	192.168.0.110	192.168.0.104	TLSv1.3	1150	Application Data, Application Data, Application Data
2.383506	192.168.0.104	192.168.0.110	TCP	54	64528 → 8883 [ACK] Seq=314 Ack=2557 Win=131328 Len=0
2.387361	192.168.0.104	192.168.0.110	TLSv1.3	1514	Change Cipher Spec
2.387361	192.168.0.104	192.168.0.110	TLSv1.3	890	Application Data, Application Data, Application Data
2.390912	192.168.0.110	192.168.0.104	TCP	54	8883 → 64528 [ACK] Seq=2557 Ack=1774 Win=64128 Len=0
2.390912	192.168.0.110	192.168.0.104	TCP	54	8883 → 64528 [ACK] Seq=2557 Ack=2610 Win=64128 Len=0
2.390973	192.168.0.104	192.168.0.110	TLSv1.3	104	Application Data
2.393372	192.168.0.110	192.168.0.104	TCP	54	8883 → 64528 [ACK] Seq=2557 Ack=2660 Win=64128 Len=0
2.396444	192.168.0.110	192.168.0.104	TLSv1.3	1205	Application Data
2.451276	192.168.0.104	192.168.0.110	TCP	54	64528 → 8883 [ACK] Seq=2660 Ack=3708 Win=130048 Len=0
2.454891	192.168.0.110	192.168.0.104	TLSv1.3	1231	Application Data, Application Data
2.455326	192.168.0.104	192.168.0.110	TLSv1.3	103	Application Data
2.457662	192.168.0.110	192.168.0.104	TCP	54	8883 → 64528 [ACK] Seq=4885 Ack=2709 Win=64128 Len=0
2.457717	192.168.0.104	192.168.0.110	TLSv1.3	78	Application Data
2.459307	192.168.0.104	192.168.0.110	TLSv1.3	78	Application Data
2.460673	192.168.0.110	192.168.0.104	TCP	54	8883 → 64528 [ACK] Seq=4885 Ack=2733 Win=64128 Len=0
2.460861	192.168.0.110	192.168.0.104	TLSv1.3	78	Application Data
2.460861	192.168.0.110	192.168.0.104	TCP	54	8883 → 64528 [FIN, ACK] Seq=4909 Ack=2733 Win=64128 Len=0

Figura 35. Traza de una captura en la que un cliente publica un mensaje usando el protocolo MQTT sobre TLS.

El intercambio de mensajes a causa del protocolo TLS resulta diferente al comentado en el estado del arte. Esto es porque durante la negociación TLS no se ha escogido un algoritmo de intercambio de claves. En este caso se ha escogido la *cipher suite* 'TLS_AES_256_GCM_SHA384'. Esto indica que:

- No se emplea ningún mecanismo para el intercambio de claves.
- Se emplea el cifrado simétrico AES 256 GCM. Esto implica usar una clave con una longitud de 256 bits.
- Se emplea el algoritmo de seguridad *hash* SHA384.

Debido a este *cipher suite* escogido, los datos intercambiado se cifran con una clave secreta que se obtiene directamente de los *nonces* intercambiados. Esto se puede observar en la primera trama que envía el servidor, dónde tras enviar el mensaje '*Server Hello'* este envía seguidamente el mensaje '*Change Cypher Spec*', el cual indica que los siguientes datos enviados serán cifrados.

Esta clase de conexiones TLS, donde no se especifica un algoritmo de intercambio de claves resultan un problema, ya que esta clase de conexiones son vulnerables a un ataque *man-in-the-middle*.

Respecto a los tiempos de procesamiento, donde se van a obviar los tiempos de retardo de la red, se puede observar que entre el inicio y el fin de la conexión TCP entre el ejemplo con cifrado y el de sin cifrar hay una diferencia de 97,661ms. Este tiempo correspondería con los tiempos de procesamiento del cifrado. Por lo que, en el caso del proyecto TwizyLine, donde en el caso extremo donde hay un pico en la red y existieran conexiones con todos los Twizy de un *parking* (20 Twizy), el retardo de procesamiento total sería de casi 2 segundos. Este margen habría que tenerlo en cuenta a la hora de escoger la velocidad a la que se mueven los Twizy dentro del *parking*. Esta velocidad es de un máximo de 10 km/h, por lo que habrá un margen de error de 5'5 metros debido al retardo de procesamiento.

5.2 Análisis comunicación real

En esta sección se va a hacer un análisis de una comunicación real entre la aplicación móvil y el servidor de TwizyLine. Se dice que es una comunicación real porque las condiciones en las que se ha capturado la traza son las siguientes:

- La traza mostrada pertenece al inicio de la conexión entre la aplicación móvil y el servidor. Este inicio de conexión ocurre cuando el usuario inicia la aplicación móvil e introduce las credenciales de forma correcta. Después de esto, el usuario recibe en su terminal las localizaciones de los *parking*s más cercanos a su localización.
- El *broker* se ejecuta en una Raspberry Pi 3 modelo B+, al igual que en la sección anterior.
- La aplicación móvil se está ejecutando en un emulador que ofrece Android Studio, en un ordenador Windows.
- La traza se ha capturado desde el mismo ordenador donde se está ejecutando el emulador de Android Studio.
- Tanto el ordenador Windows como la Raspberry, se encuentran en la misma red.

En la Figura 36, se puede observar la traza de tráfico generado con las condiciones anteriormente descritas.

11	2.606377	192.168.0.108	192.168.0.110	TCP	54 53106 → 8883 [ACK] Seq=1 Ack=1 Win=131328 Len=0
12	2.736314	192.168.0.108	192.168.0.110	TLSv1.2	211 Client Hello
13	2.738791	192.168.0.110	192.168.0.108	TCP	54 8883 → 53106 [ACK] Seq=1 Ack=158 Win=64128 Len=0
14	2.759015	192.168.0.110	192.168.0.108	TLSv1.2	1514 Server Hello
15	2.759015	192.168.0.110	192.168.0.108	TLSv1.2	981 Certificate, Server Key Exchange, Certificate Request, Server Hello Done
16	2.759193	192.168.0.108	192.168.0.110	TCP	54 53106 → 8883 [ACK] Seq=158 Ack=2388 Win=131328 Len=0
19	2.818847	192.168.0.108	192.168.0.110	TLSv1.2	1318 Certificate, Client Key Exchange, Certificate Verify, Change Cipher Spec, Encrypted Handshake Message
20	2.821060	192.168.0.110	192.168.0.108	TCP	54 8883 → 53106 [ACK] Seq=2388 Ack=1422 Win=64128 Len=0
21	2.825866	192.168.0.110	192.168.0.108	TLSv1.2	1192 New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
22	2.878824	192.168.0.108	192.168.0.110	TCP	54 53106 → 8883 [ACK] Seq=1422 Ack=3526 Win=130048 Len=0
23	2.928994	192.168.0.108	192.168.0.110	TLSv1.2	160 Application Data
24	2.931036	192.168.0.110	192.168.0.108	TCP	54 8883 → 53106 [ACK] Seq=3526 Ack=1528 Win=64128 Len=0
25	2.932496	192.168.0.110	192.168.0.108	TLSv1.2	87 Application Data
27	2.987080	192.168.0.108	192.168.0.110	TCP	54 53106 → 8883 [ACK] Seq=1528 Ack=3559 Win=130048 Len=0
59	5.743816	192.168.0.108	192.168.0.110	TLSv1.2	200 Application Data
60	5.839190	192.168.0.110	192.168.0.108	TLSv1.2	87 Application Data
61	5.890759	192.168.0.108	192.168.0.110	TCP	54 53106 → 8883 [ACK] Seq=1674 Ack=3592 Win=130048 Len=0
62	5.894817	192.168.0.110	192.168.0.108	TLSv1.2	189 Application Data, Application Data, Application Data, Application Data
63	5.919303	192.168.0.108	192.168.0.110	TLSv1.2	115 Application Data
64	5.921119	192.168.0.110	192.168.0.108	TLSv1.2	87 Application Data
65	5.968481	192.168.0.108	192.168.0.110	TCP	54 53106 → 8883 [ACK] Seq=1735 Ack=3760 Win=129792 Len=0
73	6.320920	192.168.0.108	192.168.0.110	TLSv1.2	87 Application Data
74	6.324135	192.168.0.110	192.168.0.108	TLSv1.2	87 Application Data
75	6.344137	192.168.0.108	192.168.0.110	TLSv1.2	87 Application Data
77	6.349706	192.168.0.110	192.168.0.108	TLSv1.2	107 Application Data
78	6.387994	192.168.0.108	192.168.0.110	TLSv1.2	87 Application Data
79	6.389729	192.168.0.110	192.168.0.108	TLSv1.2	87 Application Data
83	6.435803	192.168.0.108	192.168.0.110	TCP	54 53106 → 8883 [ACK] Seq=1834 Ack=3879 Win=131328 Len=0
84	6.442828	192.168.0.110	192.168.0.108	TLSv1.2	137 Application Data, Application Data
91	6.498604	192.168.0.108	192.168.0.110	TCP	54 53106 → 8883 [ACK] Seq=1834 Ack=3962 Win=131072 Len=0
93	6.500805	192.168.0.110	192.168.0.108	TLSv1.2	277 Application Data, Application Data
96	6.544649	192.168.0.108	192.168.0.110	TCP	54 53106 → 8883 [ACK] Seq=1834 Ack=4185 Win=131072 Len=0
98	6.546717	192.168.0.110	192.168.0.108	TLSv1.2	167 Application Data
101	6.590688	192.168.0.108	192.168.0.110	TCP	54 53106 → 8883 [ACK] Seq=1834 Ack=4298 Win=130816 Len=0
305	20.848976	192.168.0.108	192.168.0.110	TCP	54 53106 → 8883 [FIN, ACK] Seq=1834 Ack=4298 Win=130816 Len=0
308	20.892745	192.168.0.110	192.168.0.108	TLSv1.2	85 Encrypted Alert
309	20.892745	192.168.0.110	192.168.0.108	TCP	54 8883 → 53106 [FIN, ACK] Seq=4329 Ack=1835 Win=64128 Len=0
310	20.892843	192.168.0.108	192.168.0.110	TCP	54 53106 → 8883 [ACK] Seq=1835 Ack=4330 Win=130816 Len=0

Figura 36. Traza TLS de una comunicación entre la aplicación móvil y el servidor.

Como se puede observar, el intercambio de paquetes es generado por la negociación TLS que existe entre el servidor y la aplicación. Se puede observar en el intercambio de mensajes producidos por el protocolo *handshake*, como la aplicación y el servidor intercambian sus certificados y negocian los parámetros de la comunicación TLS, para después proceder con el intercambio de mensajes de aplicación empleando el protocolo de capa de registro.

El primer mensaje enviado es por parte de la aplicación móvil, que envía el mensaje '*Client Hello'* para iniciar el protocolo *handshake*. En la Figura 37 se puede observar

los algoritmos de cifrado soportados por la aplicación móvil junto al *nonce* enviado por el cliente.

```
    Random Bytes: 7d805d913ba004894e0e0f40b1e1469193953203f1274e5b14149c3e
  Session ID Length: 0
  Cipher Suites Length: 30
v Cipher Suites (15 suites)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca9)
    Cipher Suite: TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca8)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xc009)
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc00a)
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
    Cipher Suite: TLS_RSA_WITH_AES_128_GCM_SHA256 (0x009c)
    Cipher Suite: TLS_RSA_WITH_AES_256_GCM_SHA384 (0x009d)
    Cipher Suite: TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)
    Cipher Suite: TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)
    Cipher Suite: TLS_RSA_WITH_3DES_EDE_CBC_SHA (0x000a)
```

Figura 37. Contenido del mensaje Client Hello.

El siguiente mensaje enviado es enviado por el servidor y se trata del mensaje '*Server Hello*'. En la Figura 38 se puede observar el contenido de este mensaje. Este indica que no se usa ningún método de compresión y que los parámetros de cifrado son:

- Se emplea el mecanismo *Elliptic-curve Diffie–Hellman* para establecer la clave compartida empleada para cifrar los mensajes de aplicación.
- Se emplea el cifrado simétrico AES 256 GCM. Esto implica usar una clave con una longitud de 256 bits.
- Se emplea el algoritmo de seguridad *hash* SHA384.

```
v Handshake Protocol: Server Hello
    Handshake Type: Server Hello (2)
    Length: 61
    Version: TLS 1.2 (0x0303)
  v Random: 926b05ca04ce8a5306560e7e2502558dc2469980dd80b2b0444f574e47524401
      GMT Unix Time: Nov  4, 2047 14:36:10.000000000 Hora estándar romance
      Random Bytes: 04ce8a5306560e7e2502558dc2469980dd80b2b0444f574e47524401
    Session ID Length: 0
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
    Compression Method: null (0)
```

Figura 38. Contenido del mensaje Server Hello.

El siguiente mensaje enviado por el servidor incluye el certificado del servidor, una clave pública efímera y la solicitud del certificado del cliente. En la Figura 39 se puede observar los dos certificados que envía el servidor a la aplicación móvil. El primero corresponde con el certificado del *bróker*, el cual está firmado por la propia CA creada. El segundo es el certificado de la CA, el cual está auto firmado. El campo *issuer* hace referencia a la entidad que ha firmado el certificado y el campo *subject* a la entidad dueña del certificado.

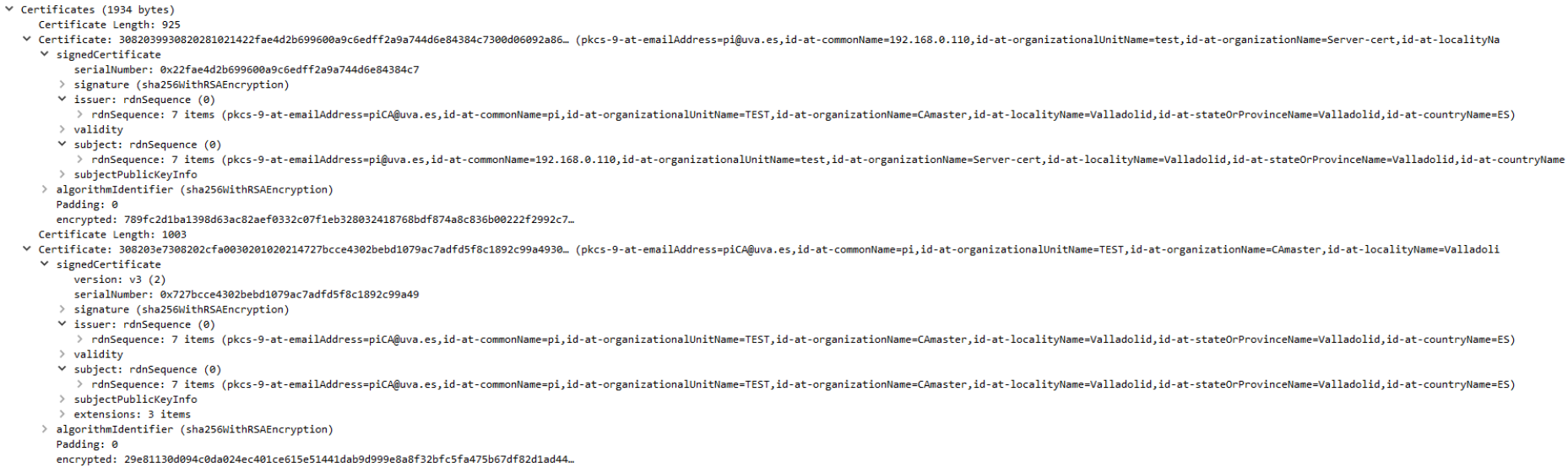

Figura 39. Contenido del mensaje Certificate donde se puede observar los certificados enviados por el servidor.

El siguiente mensaje intercambiado es enviado por el cliente. Este envía su certificado, junto a una clave pública efímera y el mensaje '*Change Cipher Spec',* que indica que los próximos datos enviados por el cliente (la aplicación móvil) serán encriptados con las claves de sesión generadas. Respecto a los certificados que envía el cliente, este envía el certificado de la CA y el certificado del propio cliente firmado por la misma CA que el certificado del *bróker*.

Por último, el servidor envía un mensaje '*New Session Ticket'* el cual indica que se actualiza el ticket de sesión, y el mensaje '*Change Cipher Spec'* que indica que los siguientes mensajes enviados por el servidor (*broker*) serán encriptados usando las claves de sesión.

Después de que finalice el protocolo *handshake*, se empiezan a enviar los datos de aplicación cifrados con las claves de sesión empleando para ello el protocolo de capa de registro, tal y como se puede ver en la Figura 40.

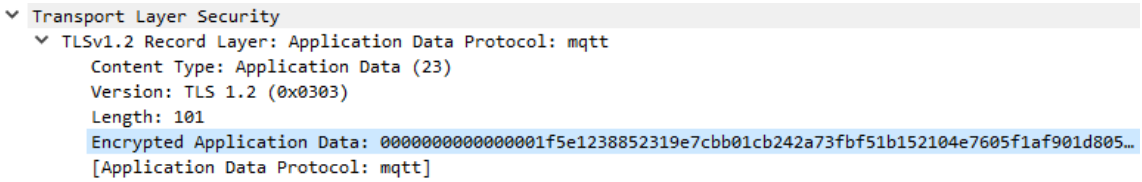

Figura 40. Contenido de un mensaje del protocolo de capa de registro.

5.3 Problemas encontrados

Respecto a la realización del *script* de *backup* que permite reducir el RTO se han encontrado los siguientes problemas:

- Complejidad para la instalación de alguna dependencia, como la base de datos empleada por el proyecto TwizyLine "MariaDB", la cual requiere de una instalación con respuestas a comandos y el *script* no es capaz de contestar a estos comandos.
- No todos los ficheros de configuración han sido modificados al completo. Esto es debido a que estos ficheros ya son creados cuando se instalan las dependencias (Como en el caso de mosquitto, viene incluido con el fichero 'mosquitto.conf'), por lo que desde la programación en *Shell,* esta se vuelve complicada cuando se necesita cargar un fichero y recorrer este fichero para cambiar los campos que modifican el comportamiento de las diferentes dependencias. Por ello, en los casos donde sí se ha realizado un cambio en la configuración, ha sido porque estos ficheros permiten la llamada a otro fichero el cual podría crear el propio usuario (en el caso de mosquitto, el fichero 'default.conf'), por lo que a través de este, se pueden introducir las configuraciones necesarias.
- Este *script* tampoco crea ni los certificados ni las claves de los clientes y el bróker. Esto es por la misma razón que no se instalan las dependencias de MariaDB. Como se ha visto en la sección 4.1.5, para crear los certificados y claves se emplea la herramienta 'openssl'. Esta herramienta necesita de introducir respuestas por parte del usuario, por lo que no se ha podido implementar esta funcionalidad.

- Tampoco revisa la versión que tiene instalada por defecto el sistema operativo ni actualiza a la versión necesaria para que el *software back.end* funcione correctamente.

Para solucionar estos problemas, se ha podido ver en algún ejemplo de Internet, que mediante ordenes 'echo' se podría realizar las funcionalidades de respuesta, como si fuese el usuario el que introdujera por línea de comandos estas respuestas, pero aún está por estudiar dicha implementación.

Respecto al despliegue de los certificados, se han tenido bastantes problemas ya que ni se encontraba con la manera adecuada de generar los certificados ni con la manera adecuada de configurar el fichero de configuración del bróker para que funcionase correctamente la implementación TLS. Se pudo observar que el problema se corregía cuando el certificado del bróker se generaba con el campo '*Common Name'* con la IP de la Raspberry.

Los certificados han sido autogenerados a través de una propia CA creada con la herramienta 'openssl'. Esto se decidió para facilitar la implementación de TLS sobre el actual sistema de TwizyLine. Pero para que este proyecto pueda usarse de una manera pública, se tendría que enviar los certificados del *broker* y un certificado por cada aplicación móvil instalada a una CA real. Esto implicaría que cuando un usuario se instalase la aplicación móvil, también debería de descargarse un certificado firmado por una CA real.

6 Conclusiones

Durante el curso de este proyecto se han logrado cumplir con los dos principales objetivos comentados en la sección 1.2. El primero ha sido el del análisis del proyecto TwizyLine, análisis realizado en el capítulo 3. Este análisis no ha sido un análisis completo del sistema, ya que solo se ha centrado en dos de los tres componentes que componen el proyecto TwizyLine: servidor y aplicación móvil. Además, este análisis no ha ido un análisis dedicado a cada componente y mediante el uso de herramientas de análisis, sino que ha sido un análisis en base a los recursos de información que cada un maneja. El segundo objetivo del proyecto ha sido el de la implementación de medidas de seguridad en base análisis anteriormente comentado. Esta implementación se ha realizado en el capítulo 4. Al igual que en el análisis, se ha llevado a cabo sobre los mismos dos componentes.

El objetivo principal de este proyecto ha sido la de tratar de fortificar el sistema del que se compone el proyecto TwizyLine. Teniendo en cuenta las diferentes necesidades planteadas en el análisis, se ha decidido implementar el protocolo TLS sobre el protocolo MQTT con el fin de establecer un canal de comunicación seguro entre el servidor y la aplicación móvil. Este canal seguro ofrece: cifrado de la comunicación, para que ningún otro usuario ajeno sea capaz de distinguir la comunicación entre el servidor y la aplicación; legitimidad del servidor frente a la aplicación y viceversa, mediante el despliegue de certificados. En resumen, la implementación de este canal seguro ofrece confianza, verificación e integridad.

A parte de ofrecer un canal seguro, se ha decidido proteger el servidor mediante la implementación de un *firewall*. También, se han decidido implementar otras herramientas para monitorizar el estado del servidor, para ofrecer una puerta de acceso remota al administrador del sistema de una forma segura y para facilitar un nuevo despliegue del servidor en otro dispositivo.

Aunque no se han logrado conseguir todas las necesidades planteadas en el análisis, se ha llegado a sentar las bases para ofrecer un sistema seguro. Tampoco se ha logrado analizar el estado del vehículo ni implementar ninguna medida en este, pero tras los conocimientos obtenidos en este proyecto y sabiendo que el módulo de comunicaciones emplea el protocolo MQTT y que este módulo emplea un sistema operativo Linux, no queda lejos poder aplicar los mismos pasos seguidos en el servidor con este módulo de comunicaciones.

En el curso de la realización de este proyecto se han adquirido conocimientos que no se habían aprendido a lo largo del grado, como la implementación del protocolo MQTT, el despliegue de certificados, el uso del *framework* Flutter o el despliegue del protocolo TLS sobre otro protocolo.

6.1 Líneas futuras

La línea futura fundamental es seguir implementando medidas de seguridad que protejan al sistema TwizyLine, ya que la ciberseguridad es un tópico que no deja de avanzar y actualizarse, y cada día que pasa se encuentran nuevos fallos en los sistemas, y por lo tanto, nuevas brechas de seguridad de las que protegerse. A continuación, se describen una serie de objetivos futuros:

- Análisis del sistema completo en búsqueda de brechas de seguridad mediante el uso de herramientas específicas de análisis.
- Implementación de las necesidades a cubrir descritas en el capítulo 3, como el cifrado de las credenciales del usuario dentro del terminal móvil.
- Acoplado de todas las medidas de seguridad implementadas con el trabajo desarrollado por C. Gómez en su Trabajo de Fin de Grado [108].
- Realizar un *pentesting* a todos los componentes del sistema para poder analizar las medidas de seguridad implementadas y poder actuar en consecuencia. El *pentesting* consiste en realizar un ataque desde la posición de un atacante potencial al sistema con el fin de encontrar vulnerabilidades, configuraciones insuficientes o deficiencias de seguridad.

Como dijo Richard Clarke, antiguo Subsecretario de Inteligencia de EE.UU., "Si gastas más en café que en seguridad informática, serás hackeado. Es más, mereces ser hackeado".

7 Referencias

[1] «Twizy Line,» [En línea]. Available: http://twizyline.com/.

[2] «Twizy E-Tech eléctrico - Renault,» [En línea]. Available: https://www.renault.es/electricos/twizy.html. [Último acceso: 18 Junio 2022].

[3] «Electromovilidad,» [En línea]. Available: http://electromovilidad.net/que-es-el-carsharing/. [Último acceso: 18 Junio 2022].

[4] «Know more about TwizyLine - TwizyLine,» [En línea]. Available: http://twizyline.com/abouttwizyline. [Último acceso: 18 Junio 2022].

[5] «Cyber Attack - Glossary | CSRC,» [En línea]. Available: https://csrc.nist.gov/glossary/term/cyber_attack. [Último acceso: 05 Junio 2022].

[6] «What Is Petya and NotPetya Ransomware? | Trellix,» [En línea]. Available: https://www.trellix.com/en-us/security-awareness/ransomware/petya.html. [Último acceso: 16 Julio 2022].

[7] J. Cook, «What are the most expensive cyber attacks of all time? | Business Leader,» 6 Mayo 2022. [En línea]. Available: https://www.businessleader.co.uk/what-are-the-most-expensive-cyber-attacks-of-all-time/. [Último acceso: 16 Julio 2022].

[8] I. Corporation, Cost of a Data Breach Report 2021, Armonk: IBM Security, 2021.

[9] «Twizycontest | GitHub,» [En línea]. Available: https://github.com/GCOdeveloper/Twizycontest. [Último acceso: 20 Julio 2022].

[10] «Wireshark,» [En línea]. Available: https://www.wireshark.org/. [Último acceso: 17 Julio 2022].

[11] d. (. user), «Open Vehicle Monitoring System, Renault Twizy, CANBUS-Objektverzeichnis,» GitHub, [En línea]. Available: https://github.com/openvehicles/Open-Vehicle-Monitoring-System/blob/master/docs/Renault-Twizy/CANBUS-Objektverzeichnis.ods. [Último acceso: 20 Julio 2022].

[12] I. R. González, Four level autonomous vehicle for an automatized parking, Valladolid: Universidad de Valladolid, 2020.

[13] MariaDB, «MariaDB Enterprise Open Source Database & SkySQL MariaDB Cloud | MariaDB,» [En línea]. Available: https://mariadb.com/. [Último acceso: 20 Julio 2022].

[14] E. Foundation, «Eclipse Paho | The Eclipse Foundation,» [En línea]. Available: https://www.eclipse.org/paho/. [Último acceso: 20 Julio 2022].

[15] Android, «Download Android Studio & App Tools - Android Developers,» [En línea]. Available: https://developer.android.com/studio. [Último acceso: 20 Julio 2022].

[16] «Flutter,» [En línea]. Available: https://flutter.dev/. [Último acceso: 15 Julio 2022].

[17] «Dart,» [En línea]. Available: https://dart.dev/. [Último acceso: 15 Julio 2022].

[18] OASIS, «MQTT Version 3.1.1 Plus Errata 01,» 10 Diciembre 2015. [En línea]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html. [Último acceso: 1 Julio 2022].

[19] HiveMQ, «The Ultimate Guide to MQTT for Beginners and Experts | HiveMQ,» [En línea]. Available: https://www.hivemq.com/mqtt-essentials/. [Último acceso: 17 Julio 2022].

[20] «Publis & Suscribe - MQTT Essential: Part 2,» 19 Enero 2015. [En línea]. Available: https://www.hivemq.com/blog/mqtt-essentials-part2-publish-subscribe/. [Último acceso: 3 Julio 2022].

[21] «MQTT Essentials: Part 3 | HiveMQ,» 17 Julio 2019. [En línea]. Available: https://www.hivemq.com/blog/mqtt-essentials-part-3-client-broker-connection-establishment/. [Último acceso: 4 Julio 2022].

[22] «¿Qué es una casa inteligente? | BBVA,» [En línea]. Available: https://www.bbva.mx/educacion-financiera/blog/que-es-una-casa-inteligente.html. [Último acceso: 7 Julio 2022].

[23] A. Munshi, «Improved MQTT Secure Transmission Flags in Smart Homes,» Sensors 2022, 2022. [En línea]. Available: https://www.mdpi.com/1424-8220/22/6/2174. [Último acceso: 7 Julio 2022].

[24] S. Jaloudi, «MQTT for IoT-based Applications in Smart Cities,» marzo 2019. [En línea]. Available: https://www.researchgate.net/publication/331529805_MQTT_for_IoT-based_Applications_in_Smart_Cities. [Último acceso: 7 Julio 2022].

[25] «Application of MQTT protocol in oil & gas industry | EMQ,» 30 Julio 2021. [En línea]. Available: https://www.emqx.com/en/blog/application-of-mqtt-protocol-in-oil-and-gas-industry. [Último acceso: 7 Julio 2022].

[26] «Advanced Authentication Mechanisms - MQTT Security Fundamentals | HiveMQ,» 27 Abril 2015. [En línea]. Available: https://www.hivemq.com/blog/mqtt-security-fundamentals-advanced-authentication-mechanisms/. [Último acceso: 7 Julio 2022].

[27] «What Is an X.509 Certificate & How Does It Work? | Sectigo,» 7 Enero 2021. [En línea]. Available: https://sectigo.com/resource-library/what-is-x509-certificate. [Último acceso: 7 Julio 2022].

[28] «Authorization - MQTT Security Fundamentals | HiveMQ,» 4 Marzo 2015. [En línea]. Available: https://www.hivemq.com/blog/mqtt-security-fundamentals-authorization/. [Último acceso: 7 Julio 2022].

[29] «TLS/SSL - MQTT Security Fundamentals | HiveMQ,» 11 Mayo 2015. [En línea]. Available: https://www.hivemq.com/blog/mqtt-security-fundamentals-tls-ssl/. [Último acceso: 7 Julio 2022].

[30] «Seguridad de la información: Aspectos a tener en cuenta | ayudaley,» [En línea]. Available: https://ayudaleyprotecciondatos.es/2020/07/14/seguridad-de-la-informacion/. [Último acceso: 19 Junio 2022].

[31] «¿Qué es la seguridad informática? | Dasit,» 22 Agosto 2018. [En línea]. Available: https://www.dasit.es/que-es-la-seguridad-informatica/. [Último acceso: 19 Junio 2022].

[32] ISO/IEC, «27001 Information technology — Security techniques — Information security management systems - Requirements».

[33] «Algo que sabes, algo que tienes y algo que eres. ¿Sabes de qué estamos hablando? | INCIBE,» 15 Diciembre 2014. [En línea]. Available: https://www.incibe.es/protege-tu-empresa/blog/algo-sabes-tienes-eres-contrasenas-identidad-online. [Último acceso: 19 Junio 2022].

[34] «Qué es el doble factor de autenticación y para qué sirve | a3SIDES,» 2 Julio 2020. [En línea]. Available: https://www.a3sides.es/blog/que-es-doble-factor-autenticacion/. [Último acceso: 19 Junio 2022].

[35] «5 pilares de la ciberseguridad | Blog Fast Lane LATAM,» [En línea]. Available: https://www.flane.com.pa/blog/5-pilares-de-la-seguridad-en-la-informacion/. [Último acceso: 21 Junio 2022].

[36] «¿Cuáles son los pilares de la seguridad de la información? | DocuSign,» [En línea]. Available: https://www.docusign.mx/blog/seguridad-de-la-informacion. [Último acceso: 21 Junio 2022].

[37] E. Mifsud, «Ministerio deEducación, Cultura y Deporte | Observatorio Tecnológico,» 26 Mayo 2012. [En línea]. Available: recursostic.educacion.es/observatorio/web/ca/software/software-

general/1040-introduccion-a-la-seguridad-informatica?start=1. [Último acceso: 21 Junio 2022].

[38] B. S. y. T. K. Niels Ferguson, Cryptography Engineering: Design Principles and Practical Applications, Indianapolis: Wiley Publishing, Inc., 2010.

[39] A. M. y. D. S. Colin Boyd, Protocols for Authentication and Key Establishment, Berlin, Heidelberg: Springer Berlin Heidelberg, 2020.

[40] M. R. Cayetano, *Seguridad en redes de comunicaciones: Protocolos de seguridad en el nivel de trasnporte,* Valladolid: E.T.S. de Ingenieros de Telecomunicación, Universidad de Valladolid, 2022.

[41] INCIBE, «Glosario de términos de ciberseguridad: una guía de aproximación para el empresario,» 2022. [En línea]. Available: https://www.incibe.es/sites/default/files/contenidos/guias/doc/guia_glosario_ciberseguridad_2021.pdf. [Último acceso: 7 Julio 2022].

[42] A. B. &. V. Sapra, *Security Incidents & Response Against Cyber Attacks,* Ghent, Belgium: EAI/Springer Innovations in Communication and Computing, 2021.

[43] C. Point, «CYBER ATTACK TRENDS - Mid Year Report 2021,» [En línea]. Available: https://www.checkpoint.com/downloads/resources/cyber-attack-trends-report-mid-year-2021.pdf. [Último acceso: 10 Julio 2022].

[44] E. Georgescu, «Top 10 Most Dangerous Banking Malware That Can Empty Your Bank Account | Heimdal Security,» 15 Julio 2021. [En línea]. Available: https://heimdalsecurity.com/blog/banking-malware-trojans/. [Último acceso: 10 Julio 2022].

[45] «Cryptojacking – What is it? | Malwarebytes,» [En línea]. Available: https://www.malwarebytes.com/cryptojacking. [Último acceso: 10 Julio 2022].

[46] J. Santaella, «¿En qué consiste la minería de criptomonedas y qué usos tiene? Te lo explicamos | Economia3,» 14 Mayo 2022. [En línea]. Available: https://economia3.com/que-es-mineria-criptomonedas-y-que-usos-tiene/. [Último acceso: 10 Julio 2022].

[47] S. Moore, «7 Top Trends in Cybersecurity for 2022 | Gartner,» 13 Abril 2022. [En línea]. Available: https://www.gartner.com/en/articles/7-top-trends-in-cybersecurity-for-2022. [Último acceso: 10 Julio 2022].

[48] «Cyber-Physical Systems | National Science Foundation,» [En línea]. Available: https://www.nsf.gov/news/special_reports/cyber-physical/. [Último acceso: 18 Julio 2022].

[49] P. E. y. e. C. d. l. U. Europea, REGLAMENTO (UE) 2016/679 DEL PARLAMENTO EUROPEO Y DEL CONSEJO, Bruselas: Diario Oficial de la Unión Europea, 2016.

[50] Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales, Boletín Oficial del Estado, 2018.

[51] «Agencia Española de Protección de Datos,» [En línea]. Available: https://www.aepd.es/es. [Último acceso: 10 Julio 2022].

[52] «¿Qué es la seguridad de las aplicaciones? | vmware,» [En línea]. Available: https://www.vmware.com/latam/topics/glossary/content/application-security.html. [Último acceso: 6 Julio 2022].

[53] «Seguridad de las aplicaciones móviles: ¿Cómo garantizarla? | NTS SEIDOR,» 13 Septiembre 2021. [En línea]. Available: https://www.nts-solutions.com/blog/seguridad-aplicaciones-moviles.html. [Último acceso: 6 Julio 2022].

[54] A. K. Tatyana Shishkova, «Virología móvil 2021 | Securelist,» 1 Marzo 2022. [En línea]. Available: https://securelist.lat/mobile-malware-evolution-2021/96289/. [Último acceso: 6 Julio 2022].

[55] L. Fernández, «Aplicaciones móviles: un gran fallo de seguridad | redes zone,» 8 Febrero 2020. [En línea]. Available: https://www.redeszone.net/tutoriales/seguridad/aplicaciones-moviles-seguridad/. [Último acceso: 6 Julio 2022].

[56] «Qué es el hacking ético | ticnegocios,» [En línea]. Available: https://ticnegocios.camaravalencia.com/servicios/tendencias/que-es-el-hacking-etico/. [Último acceso: 7 Julio 2022].

[57] «About the OWASP Foundation | OWASP,» [En línea]. Available: https://owasp.org/about/. [Último acceso: 7 Julio 2022].

[58] AEPD, «ANÁLISIS DE LOS FLUJOS DE INFORMACIÓN EN ANDROID - HERRAMIENTAS PARA EL CUMPLIMIENTO DE LA RESPONSABILIDAD PROACTIVA,» 2019. [En línea]. Available: https://www.aepd.es/sites/default/files/2019-09/estudio-flujos-informacion-android.pdf. [Último acceso: 18 Julio 2022].

[59] «Los 20 mejores consejos de seguridad al usar apps móviles | Grupo Atico34,» [En línea]. Available: https://protecciondatos-lopd.com/empresas/seguridad-apps-moviles/#7_apps_de_seguridad_para_tu_smartphone. [Último acceso: 7 Julio 2022].

[60] D. Véliz, «Top 10: las apps más descargadas del mundo (2022) | Marketing4ecommerce,» 1 Febrero 2022. [En línea]. Available:

https://marketing4ecommerce.net/top-apps-mas-descargadas-en-espana-en-mundo/. [Último acceso: 7 Julio 2022].

[61] I. Fanego, «Cuántos usuarios tiene WhatsApp en 2021 y otros datos interesantes | AppCritic,» 1 Julio 2021. [En línea]. Available: https://appcritic.es/usuarios-whatsapp/. [Último acceso: 18 Junio 2022].

[62] R. Fernández, « Número de usuarios de smartphones a nivel mundial desde 2016 hasta 2021 | Statista,» 14 Febrero 2022. [En línea]. Available: https://es.statista.com/estadisticas/636569/usuarios-de-telefonos-inteligentes-a-nivel-mundial/. [Último acceso: 18 Junio 2022].

[63] R. F. García, «Aplicaciones de mensajería más populares: Top Messaging Apps 2021 | respond.io,» 14 Septiembre 2021. [En línea]. Available: https://es.respond.io/blog/top-messaging-apps. [Último acceso: 18 Junio 2022].

[64] «Información sobre el cifrado de extremo a extremo | Servicio de ayuda de WhatsApp,» [En línea]. Available: https://faq.whatsapp.com/791574747982248/?locale=es_ES. [Último acceso: 19 Junio 2022].

[65] G. Ryles, «Is WhatsApp safe? We asked the experts if you should be using it in 2022 | Trusted Reviews,» 25 Enero 2022. [En línea]. Available: https://www.trustedreviews.com/news/is-whatsapp-safe-2-4199326. [Último acceso: 8 Julio 2022].

[66] G. Á. Marañon, «Si toda tu vida privada está en WhatsApp, ¿quién más la conoce? | ThinkBig,» 13 Marzo 2020. [En línea]. Available: https://blogthinkbig.com/cifrado-vida-privada-esta-en-whatsapp-quien-mas-la-conoce. [Último acceso: 19 Junio 2022].

[67] Z. Doffman, «Why You Should Stop Using Facebook Messenger | Forbes,» 25 Julio 2020. [En línea]. Available: https://www.forbes.com/sites/zakdoffman/2020/07/25/why-you-should-stop-using-facebook-messenger-encryption-whatsapp-update-twitter-hack/?sh=35bb08ea69ad. [Último acceso: 8 Julio 2022].

[68] M. Tillman, «Pocket-lint,» 28 Enero 2022. [En línea]. Available: https://www.pocket-lint.com/es-es/aplicaciones/noticias/facebook/159846-facebook-messenger-implementa-el-cifrado-para-todos-como-habilitarlo-en-los-chats. [Último acceso: 19 Junio 2022].

[69] E. Malo, «Facebook Messenger añade el cifrado de extremo a extremo a sus chats de texto | MuyComputer,» 28 Enero 2022. [En línea]. Available: https://www.muycomputer.com/2022/01/28/facebook-messenger-cifrado-chats/. [Último acceso: 19 Junio 2022].

[70] H. Davies, «Is Telegram safe? Here's what security experts have to say about the app | Trusted Reviews,» 26 Enero 2022. [En línea]. Available: https://www.trustedreviews.com/news/is-telegram-safe-4130553. [Último acceso: 8 Julio 2022].

[71] C. Pastorino, «Cómo configurar la privacidad y seguridad en Telegram | WeLiveSecurity,» 1 Junio 2021. [En línea]. Available: https://www.welivesecurity.com/la-es/2021/06/01/como-configurar-privacidad-seguridad-telegram/. [Último acceso: 19 Junio 2022].

[72] K. How, «¿Qué es un servidor? | IONOS,» 15 Septiembre 2020. [En línea]. Available: https://www.ionos.es/digitalguide/servidores/know-how/que-es-un-servidor-un-concepto-dos-definiciones/. [Último acceso: 24 Junio 2022].

[73] «¿Cuáles son los servidores web más utilizados? | Stacksale,» 7 Junio 2022. [En línea]. Available: https://www.stackscale.com/es/blog/top-servidores-web/. [Último acceso: 25 Junio 2022].

[74] S. P. Arranz, Back-end implementation for an automatized car parking, Valladolid: Universidad de Valladolid, 2022.

[75] «Tipos de bases de datos y las mejores bases de datos | Pandora FMS,» 29 Marzo 2022. [En línea]. Available: https://pandorafms.com/blog/es/tipos-de-bases-de-datos-y-las-mejores-bases-de-datos/. [Último acceso: 25 Junio 2022].

[76] Á. D. León, «Servidor de Correo: ¿Qué es? ¿Para qué sirve? | infranetworking,» 12 Marzo 2019. [En línea]. Available: https://blog.infranetworking.com/servidor-de-correo/. [Último acceso: 6 Julio 2022].

[77] «¿Qué es un servidor DNS? | dinahosting,» [En línea]. Available: https://dinahosting.com/ayuda/que-es-un-servidor-dns/. [Último acceso: 6 Julio 2022].

[78] «¿Qué tipos de servidores hay? | claranet,» [En línea]. Available: https://www.claranet.es/blog/que-tipos-de-servidores-hay. [Último acceso: 25 Junio 2022].

[79] «Cluster de servidores, ¿qué es y como funciona? | Solingest,» [En línea]. Available: https://www.solingest.com/cluster-de-servidores-que-es-y-como-funciona. [Último acceso: 25 Junio 2022].

[80] «Ventajas y Desventajas de Linux | Ciberaula,» [En línea]. Available: https://linux.ciberaula.com/articulo/ventajas_inconvenientes_linux/. [Último acceso: 26 Junio 2022].

[81] «La importancia de la seguridad en un vehiculo | ITV.com.es,» [En línea]. Available: https://itv.com.es/la-importancia-de-la-seguridad-en-un-vehiculo. [Último acceso: 6 Julio 2022].

[82] «SISTEMAS DE SEGURIDAD EN EL AUTOMÓVIL | AprendEmergencias,» [En línea]. Available: https://www.aprendemergencias.es/seguridad-vial/sistemas-de-seguridad-en-el-veh%C3%ADculo/. [Último acceso: 26 Junio 2022].

[83] «What is connected car cyber security | Synopsis,» [En línea]. Available: https://www.synopsys.com/glossary/what-is-connected-car-cyber-security.html. [Último acceso: 10 Julio 2022].

[84] «The CAN Bus Protocol Tutorial | kvaser,» [En línea]. Available: https://www.kvaser.com/can-protocol-tutorial/. [Último acceso: 10 Julio 2022].

[85] « Protecting Cars, Trucks and Commercial Vehicles from Hacking – an Overview | Argus Cyber Security,» [En línea]. Available: https://argus-sec.com/car-hacking/. [Último acceso: 10 Julio 2022].

[86] «OMSP - Open Mobility Security Project,» OMSP, 30 Abril 2020. [En línea]. Available: https://github.com/zerolynx/omsp. [Último acceso: 27 Junio 2022].

[87] A. M. Hernández, Front-end implementation for an automatized car parking, Valladolid: Universidad de Valladolid, 2022.

[88] «OWASP Mobile Top 10 | OWASP,» [En línea]. Available: https://owasp.org/www-project-mobile-top-10/. [Último acceso: 11 Julio 2022].

[89] «OWASP Top Ten | OWASP,» [En línea]. Available: https://owasp.org/www-project-top-ten/. [Último acceso: 11 Julio 2022].

[90] J. Umawing, «What is a Man-in-the-Middle attack? MitM attacks explained | Malwarebytes Labs,» 27 Abril 2021. [En línea]. Available: https://blog.malwarebytes.com/101/2018/07/when-three-isnt-a-crowd-man-in-the-middle-mitm-attacks-explained/. [Último acceso: 18 Junio 2022].

[91] «Mobile Security Framework | MobSF,» [En línea]. Available: https://github.com/MobSF/Mobile-Security-Framework-MobSF. [Último acceso: 20 Julio 2022].

[92] C. Burdova, «¿Qué es un rootkit y cómo se elimina? | Avast,» 21 Marzo 2022. [En línea]. Available: https://www.avast.com/es-es/c-rootkit. [Último acceso: 13 Julio 2022].

[93] I. Belcic, «¿Qué es un exploit en seguridad informática? | AVG,» 6 Mayo 2021. [En línea]. Available: https://www.avg.com/es/signal/computer-security-exploits. [Último acceso: 13 Julio 2022].

[94] «Puerta Trasera | Panda Security,» [En línea]. Available: https://www.pandasecurity.com/es/security-info/back-door/. [Último acceso: 13 Julio 2022].

[95] M. R. Cayetano, Seguridad en redes de comunicaciones: Sistemas de defensa perimetral, Universidad de Valladolid: E.T.S. de Ingenieros de Telecomunicación, 2022.

[96] J. Strandboge, «UFW,» [En línea]. Available: https://launchpad.net/ufw. [Último acceso: 20 Julio 2022].

[97] G. Domantas, «What Is SSH: Understanding Encryption, Ports and Connection | Hostinger Tutorials,» 1 Junio 2022. [En línea]. Available: https://www.hostinger.com/tutorials/ssh-tutorial-how-does-ssh-work. [Último acceso: 14 Julio 2022].

[98] «What is Shell Script and How Does It Work | TechTarget,» 01 Septiembre 2019. [En línea]. Available: https://www.techtarget.com/searchdatacenter/definition/shell-script. [Último acceso: 16 Julio 2022].

[99] «Know Your RTO & RPO For Disaster Recovery Time | Intrust IT,» [En línea]. Available: https://www.intrust-it.com/know-your-rto-rpo-for-disaster-recovery/. [Último acceso: 18 Julio 2022].

[100] «Eclipse Foundation,» [En línea]. Available: https://www.eclipse.org. [Último acceso: 15 Julio 2022].

[101] «openssl(1) - Linux man page,» [En línea]. Available: https://linux.die.net/man/1/openssl. [Último acceso: 15 Julio 2022].

[102] «Hardcoded/Embedded Passwords | BeyondTrust,» [En línea]. Available: https://www.beyondtrust.com/resources/glossary/hardcoded-embedded-passwords. [Último acceso: 18 Julio 2022].

[103] «fail2ban,» [En línea]. Available: https://www.fail2ban.org/wiki/index.php/Main_Page. [Último acceso: 15 Julio 2022].

[104] «The Rootkit Hunter project,» [En línea]. Available: http://rkhunter.sourceforge.net/. [Último acceso: 15 Julio 2022].

[105] «debsecan - the Debian Security Analyzer,» 16 Septiembre 2020. [En línea]. Available: https://wiki.debian.org/DebianSecurity/debsecan. [Último acceso: 15 Julio 2022].

[106] M. Kerrisk, «cron(8) — Linux manual page,» Linux, 27 Agosto 2021. [En línea]. Available: https://www.man7.org/linux/man-pages/man8/cron.8.html. [Último acceso: 21 Julio 2022].

[107] «mqtt_client | GitHub,» shamblett, [En línea]. Available: https://github.com/shamblett/mqtt_client. [Último acceso: 15 Julio 2022].

[108] C. G. Diego, Guiado de vehículo autónomo mediante tecnología LiDAR, Valladolid: Universidad de Valladolid, 2022.

[109] «MQTT Version 3.1.1,» [En línea]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html. [Último acceso: 18 Junio 2022].

[110] L. Cavagnis, «Android and MQTT: A Simple Guide | Medium,» 4 Diciembre 2020. [En línea]. Available: https://medium.com/swlh/android-and-mqtt-a-simple-guide-cb0cbba1931c. [Último acceso: 18 Junio 2022].

[111] S. Fuentes, «Los SMS tienen un precio desorbitado frente a su coste | Xataka,» 6 Abril 2009. [En línea]. Available: https://www.xatakamovil.com/mercado/los-sms-tienen-un-precio-desorbitado-frente-a-su-coste. [Último acceso: 18 Junio 2022].

[112] T. Belaire, «¿Estamos ante el fin de los SMS de las tarifas móviles? | HelpMyCash,» 14 Agosto 2018. [En línea]. Available: https://www.helpmycash.com/blog/estamos-ante-el-fin-de-los-sms-de-las-tarifas-moviles/. [Último acceso: 18 Junio 2022].

[113] R. Fernández, « Cuota de mercado de las empresas de telefonía móvil en España a diciembre de 2021 | Statista,» 18 Febrero 2022. [En línea]. Available: https://es.statista.com/estadisticas/553819/cuota-de-mercado-de-los-operadores-de-telefonia-movil-en-espana/. [Último acceso: 18 Junio 2022].

[114] «Todas las estadísticas sobre móviles que deberías conocer | ditrendia,» [En línea]. Available: https://mktefa.ditrendia.es/blog/todas-las-estad%C3%ADsticas-sobre-m%C3%B3viles-que-deber%C3%ADas-conocer-mwc19. [Último acceso: 18 Junio 2022].

[115] «¿Existe un límite de caracteres para los mensajes SMS? | Genesys Cloud,» [En línea]. Available: https://es-help.mypurecloud.com/faqs/character-limit-sms-messages/. [Último acceso: 24 Junio 2022].

[116] A. Salinas, «Historia de WhatsApp : qué es, quién la creó y cómo funciona | mott,» [En línea]. Available: https://mott.marketing/informacion-sobre-que-es-quien-creo-y-como-funciona-la-aplicacion-whatsapp/. [Último acceso: 24 Junio 2022].

[117] «Firebase Cloud Messaging | Firebase Documentation,» [En línea]. Available: https://firebase.google.com/docs/cloud-messaging/. [Último acceso: 1 Julio 2022].

[118] «Is MQTT better than HTTP for mobile app (iOS, Android, etc) client-server communication? | Quora,» [En línea]. Available: https://www.quora.com/Is-MQTT-better-than-HTTP-for-mobile-app-iOS-Android-etc-client-server-communication. [Último acceso: 1 Julio 2022].

[119] «Infraestructura de claves públicas (PKI) | IBM MQ,» 20 Abril 2021. [En línea]. Available: https://www.ibm.com/docs/es/ibm-mq/7.5?topic=ssfksj-7-5-0-com-ibm-mq-sec-doc-q009900--htm. [Último acceso: 8 Julio 2022].

[120] K. Panetta, «The Top 8 Security and Risk Trends We're Watching | Gartner,» 15 Noviembre 2021. [En línea]. Available: https://www.gartner.com/smarterwithgartner/gartner-top-security-and-risk-trends-for-2021. [Último acceso: 10 Julio 2022].

[121] M. Azad, «What Is iptables and How to Use It? | Medium,» 7 Mayo 2020. [En línea]. Available: https://medium.com/skilluped/what-is-iptables-and-how-to-use-it-781818422e52. [Último acceso: 14 Julio 2022].

Anexo I

El siguiente anexo muestra el contenido de las tablas de 'iptables' generadas a través de la herramienta UFW. Se puede observar que usando 4 comandos de esta herramienta y modificando 4 palabras de un fichero de configuración, se obtiene el siguiente resultado:

```
root@raspberrypi:~# iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination
ufw-before-logging-input  all  --  anywhere             anywhere
ufw-before-input  all  --  anywhere             anywhere
ufw-after-input  all  --  anywhere             anywhere
ufw-after-logging-input  all  --  anywhere             anywhere
ufw-reject-input  all  --  anywhere             anywhere
ufw-track-input  all  --  anywhere             anywhere

Chain FORWARD (policy DROP)
target     prot opt source               destination
ufw-before-logging-forward  all  --  anywhere             anywhere
ufw-before-forward  all  --  anywhere             anywhere
ufw-after-forward  all  --  anywhere             anywhere
ufw-after-logging-forward  all  --  anywhere             anywhere
ufw-reject-forward  all  --  anywhere             anywhere
ufw-track-forward  all  --  anywhere             anywhere

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
ufw-before-logging-output  all  --  anywhere             anywhere
ufw-before-output  all  --  anywhere             anywhere
ufw-after-output  all  --  anywhere             anywhere
ufw-after-logging-output  all  --  anywhere             anywhere
ufw-reject-output  all  --  anywhere             anywhere
ufw-track-output  all  --  anywhere             anywhere
```

```
Chain ufw-after-forward (1 references)
target     prot opt source               destination


Chain ufw-after-input (1 references)
target     prot opt source               destination
ufw-skip-to-policy-input   udp  --  anywhere             anywhere
udp dpt:netbios-ns
ufw-skip-to-policy-input   udp  --  anywhere             anywhere
udp dpt:netbios-dgm
ufw-skip-to-policy-input   tcp  --  anywhere             anywhere
tcp dpt:netbios-ssn
ufw-skip-to-policy-input   tcp  --  anywhere             anywhere
tcp dpt:microsoft-ds
ufw-skip-to-policy-input   udp  --  anywhere             anywhere
udp dpt:bootps
ufw-skip-to-policy-input   udp  --  anywhere             anywhere
udp dpt:bootpc
ufw-skip-to-policy-input   all  --  anywhere             anywhere
ADDRTYPE match dst-type BROADCAST


Chain ufw-after-logging-forward (1 references)
target     prot opt source               destination
LOG        all  --  anywhere             anywhere            limit:
avg 3/min burst 10 LOG level warning prefix "[UFW BLOCK] "


Chain ufw-after-logging-input (1 references)
target     prot opt source               destination
LOG        all  --  anywhere             anywhere            limit:
avg 3/min burst 10 LOG level warning prefix "[UFW BLOCK] "


Chain ufw-after-logging-output (1 references)
target     prot opt source               destination


Chain ufw-after-output (1 references)
```

```
target     prot opt source               destination


Chain ufw-before-forward (1 references)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere             ctstate
RELATED,ESTABLISHED
ACCEPT     icmp --  anywhere             anywhere             icmp
destination-unreachable
ACCEPT     icmp --  anywhere             anywhere             icmp time-
exceeded
ACCEPT     icmp --  anywhere             anywhere             icmp
parameter-problem
ACCEPT     icmp --  anywhere             anywhere             icmp echo-
request
ufw-user-forward  all  --  anywhere             anywhere


Chain ufw-before-input (1 references)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere             ctstate
RELATED,ESTABLISHED
ufw-logging-deny  all  --  anywhere           anywhere           ctstate
INVALID
DROP       all  --  anywhere             anywhere             ctstate
INVALID
DROP       icmp --  anywhere             anywhere             icmp
destination-unreachable
DROP       icmp --  anywhere             anywhere             icmp time-
exceeded
DROP       icmp --  anywhere             anywhere             icmp
parameter-problem
DROP       icmp --  anywhere             anywhere             icmp echo-
request
ACCEPT     udp  --  anywhere             anywhere             udp
spt:bootps dpt:bootpc
ufw-not-local  all  --  anywhere             anywhere
```

```
ACCEPT      udp   --   anywhere               224.0.0.251            udp
dpt:mdns

ACCEPT      udp   --   anywhere               239.255.255.250        udp
dpt:1900

ufw-user-input  all   --   anywhere             anywhere


Chain ufw-before-logging-forward (1 references)

target     prot opt source               destination


Chain ufw-before-logging-input (1 references)

target     prot opt source               destination


Chain ufw-before-logging-output (1 references)

target     prot opt source               destination


Chain ufw-before-output (1 references)

target     prot opt source               destination

ACCEPT     all   --   anywhere             anywhere

ACCEPT     all   --   anywhere              anywhere             ctstate
RELATED,ESTABLISHED

ufw-user-output  all   --   anywhere             anywhere


Chain ufw-logging-allow (0 references)

target     prot opt source               destination

LOG        all   --   anywhere              anywhere              limit:
avg 3/min burst 10 LOG level warning prefix "[UFW ALLOW] "


Chain ufw-logging-deny (2 references)

target     prot opt source               destination

RETURN     all   --   anywhere              anywhere             ctstate
INVALID limit: avg 3/min burst 10

LOG        all   --   anywhere              anywhere              limit:
avg 3/min burst 10 LOG level warning prefix "[UFW BLOCK] "
```

```
Chain ufw-not-local (1 references)
target     prot opt source               destination
RETURN     all  --  anywhere             anywhere             ADDRTYPE
match dst-type LOCAL
RETURN     all  --  anywhere             anywhere             ADDRTYPE
match dst-type MULTICAST
RETURN     all  --  anywhere             anywhere             ADDRTYPE
match dst-type BROADCAST
ufw-logging-deny  all  --  anywhere           anywhere            limit:
avg 3/min burst 10
DROP       all  --  anywhere             anywhere

Chain ufw-reject-forward (1 references)
target     prot opt source               destination

Chain ufw-reject-input (1 references)
target     prot opt source               destination

Chain ufw-reject-output (1 references)
target     prot opt source               destination

Chain ufw-skip-to-policy-forward (0 references)
target     prot opt source               destination
DROP       all  --  anywhere             anywhere

Chain ufw-skip-to-policy-input (7 references)
target     prot opt source               destination
DROP       all  --  anywhere             anywhere

Chain ufw-skip-to-policy-output (0 references)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere
```

```
Chain ufw-track-forward (1 references)
target     prot opt source               destination

Chain ufw-track-input (1 references)
target     prot opt source               destination

Chain ufw-track-output (1 references)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere             ctstate
NEW
ACCEPT     udp  --  anywhere             anywhere             ctstate
NEW

Chain ufw-user-forward (1 references)
target     prot opt source               destination

Chain ufw-user-input (1 references)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere             tcp
dpt:1883
ACCEPT     udp  --  anywhere             anywhere             udp
dpt:1883
ACCEPT     tcp  --  anywhere             anywhere             tcp
dpt:8883
ACCEPT     udp  --  anywhere             anywhere             udp
dpt:8883
ACCEPT     tcp  --  anywhere             anywhere             tcp
dpt:22445
ACCEPT     udp  --  anywhere             anywhere             udp
dpt:22445

Chain ufw-user-limit (0 references)
target     prot opt source               destination
LOG        all  --  anywhere             anywhere             limit:
avg 3/min burst 5 LOG level warning prefix "[UFW LIMIT BLOCK] "
```

```
REJECT     all  --  anywhere             anywhere             reject-
with icmp-port-unreachable

Chain ufw-user-limit-accept (0 references)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere

Chain ufw-user-logging-forward (0 references)
target     prot opt source               destination

Chain ufw-user-logging-input (0 references)
target     prot opt source               destination

Chain ufw-user-logging-output (0 references)
target     prot opt source               destination

Chain ufw-user-output (1 references)
target     prot opt source               destination
```

Anexo II

En este anexo se va a mostrar una captura de pantalla de un escaneado de puertos realizado con la herramienta 'nmap' al servidor de TwizyLine una vez se ha implementado el cortafuegos. Primero se escanean los *well-known ports*, después a los primeros 4096 puertos y después un escaneo hasta el puerto 23000.

```
pi@raspberrypi:~ $ nmap -p 0-1024 10.0.103.46
Starting Nmap 7.80 ( https://nmap.org ) at 2022-07-15 16:07 CEST
Nmap scan report for 10.0.103.46
Host is up (0.00082s latency).
All 1025 scanned ports on 10.0.103.46 are closed

Nmap done: 1 IP address (1 host up) scanned in 0.42 seconds
pi@raspberrypi:~ $ nmap -p 0-4096 10.0.103.46
Starting Nmap 7.80 ( https://nmap.org ) at 2022-07-15 16:08 CEST
Nmap scan report for 10.0.103.46
Host is up (0.0011s latency).
Not shown: 4096 closed ports
PORT     STATE SERVICE
1883/tcp open  mqtt

Nmap done: 1 IP address (1 host up) scanned in 1.07 seconds
pi@raspberrypi:~ $ nmap -p 0-23000 10.0.103.46
Starting Nmap 7.80 ( https://nmap.org ) at 2022-07-15 16:08 CEST
Nmap scan report for 10.0.103.46
Host is up (0.0011s latency).
Not shown: 22998 closed ports
PORT      STATE SERVICE
1883/tcp  open  mqtt
8883/tcp  open  secure-mqtt
22445/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 5.01 seconds
```

Anexo III

Contenido del fichero de *backup* implementado:

```
#!/bin/bash

# -*- ENCODING: UTF-8 -*-

# Author: Alvar Alonso Gonzalez <alvar.alonso@alumnos.uva.es>

#

# Backup script for raspbian server dependencies

# Update & Upgrade dependencies

apt-get update

apt-get upgrade -y

# Install mosquitto dependencies

apt-get install -y mosquitto

cd /etc/mosquitto/conf.d

touch default.conf

echo -e "allow_anonymous false\npassword_file /etc/mosquitto/passwd\nlistener 1883" > default.conf

cd ../passwd

echo -e "samuel:1234\n" > passwd

mosquitto_passwd -U passwd

systemctl restart mosquitto

apt-get install -y mosquitto-clients

# Python 3.9.2 is installed by default at Raspbian

#

# Install MariaDB dependencies

# Edit ssh's configuration file for better security

# First, a new user is needed

useradd -m -s /bin/bash remoto

echo remoto:conexionRemoto22 | chpasswd

cd etc/ssh/sshd_config.d

touch default.conf

echo -e "# Default Configuration: Use Port 22445, don't allow empty passwd, only allow remoto user and don't allow root to login\nPort
```

```
22445\nPermitEmptyPasswords no\nAllowUsers remoto\nPermitRootLogin no\nLoginGraceTime 30\nMaxAuthTries 3\nMaxStartups 3\n" > default.conf

systemctl restart ssh

# Install fail2ban

apt-get install -y fail2ban

cd /etc/fail2ban/jail.d

echo -e "[DEFAULT]\nignoreip=127.0.0.1/8\nbantime=1800\nmaxretry=3\n\n[sshd]\nenabled=true\n" > defaults-debian.conf

# Install rkhunter, which detect rootkits

apt-get install -y rkhunter

# Update rkhunter data base

rkhunter --propupd

rkhunter --update

# Install debsecan - Evaluates system's security and show the most common vulnerabilities

apt-get install -y debsecan

# Install ufw firewall

apt-get install -y ufw

ufw enable

ufw default deny incoming

ufw allow in 22445 comment "allows ssh connections"

ufw allow in 1883 comment "allows MQTT connections"

ufw allow in 8883 comment "allows MQTT over TLS connections"

exit
```