UNIVERSIDADE DA CORUÑA

Facultad de Informática

TRABAJO DE FIN DE GRADO

GRADO EN INGENIERÍA INFORMÁTICA

Mención en Tecnologías de la Información

Detección de anomalías en redes IoT mediante Mapas Autooganizados

Autor: Miguel Ouviña Santamaría

Directores: Francisco Javier Nóvoa Manuel,
Manuel Fernández López-Vizcaíno

A Coruña, 6 de septiembre de 2019

A mi Madre

Agradecimientos

Sin tener muy claro cómo he llegado hasta aquí, a escribir estas lineas, solo me queda agradecer a todos las personas que me han ayudado en este camino. En especial a mi madre, por educarme, apoyarme y permitirme, con sus sacrificios, poder lograr algo que nunca hubiese imaginado.

También dar las gracias a mis compañeros y amigos por hacer todos estos años mucho más llevaderos, a todos esos profesores que disfrutan compartiendo sus conocimientos, y en especial a mis tutores, Fran y Manuel, por descubrirme esta temática y ayudarme con el proyecto pese a todos los contratiempos.

Resumen

El objetivo principal de este trabajo fin de grado es el estudio de las características del tráfico en las redes IoT mediante redes de neuronas artificiales de tipo SOM. Para ello se han recopilado diferentes *datasets* con información de tráfico de redes IoT, se han analizado las características del tráfico para su clasificación, se ha elaborado un preprocesado de datos para su adecuación al sistema de procesamiento, se establecieron diferentes configuraciones de SOM y se ha hecho un análisis y comparación de los resultados obtenidos. Finalmente se han planteado las conclusiones del trabajo realizado.

Entre la diversidad de entornos en los que están presentes las redes IoT, en este trabajo nos hemos centrado en los Sistemas de Control Industrial, y más concretamente en los formados por los sistemas SCADA. Éstos son importantes en procesos industriales e infraestructuras críticas. Tradicionalmente, la infraestructura de comunicaciones de los sistemas industriales ha sido independiente de la red de datos, estando completamente aisladas entre sí. Sin embargo en los últimos años con el advenimiento de "Internet de las Cosas" (Internet of Things, IoT) cada vez es más habitual integrar redes de sensores y actuadores en entornos IP para mejorar la eficiencia y la productividad. Esta integración con Internet ha traído varios problemas relacionados con la seguridad ya que cuando estos entornos fueron desarrollados la seguridad no fue un tema principal a tener en cuenta.

Los sistemas de detección de intrusiones podrían descubrir posibles ataques a estos entornos, resolviendo así algunos de los problemas relacionados con la seguridad. En este trabajo se propone un sistema de detección de anomalías, que utiliza el análisis de las características más relevantes de las cabeceras de los paquetes para la construcción del perfil de comportamiento normal de la red y de las anomalías que se puedan producir en las redes SCADA.

Palabras clave:

- IoT
- Detección de anomalías
- SOM
- Industria 4.0
- Sistemas de control industrial
- SCADA
- Modbus

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Contexto y motivación

Tradicionalmente, la infraestructura de comunicaciones de los sistemas industriales ha sido independiente de la red de datos, estando completamente aisladas entre sí, es decir, la red **IT** (Tecnologías de la Información) estaba separada de la red **OT** (Tecnologías de la Operación). Sin embargo en los últimos años con el advenimiento de "Internet de las Cosas" (**Internet of Things, IoT**) cada vez es más habitual integrar redes de sensores y actuadores en entornos IP. En 1999 Kevin Ashton [17] mencionó por primera vez el término IoT y desde entonces este paradigma se ha ido desplegando en multitud de organizaciones, siendo los últimos años los de mayor crecimiento. Según diferentes estudios, se estima que a lo largo del 2020 habrá más de veinte mil millones de estos dispositivos conectados a Internet [26].

Una de las aplicaciones fundamentales de IoT es conectar elementos con el fin de recolectar, almacenar, procesar y presentar grandes cantidades de datos desde sensores altamente dispersos y enviar señales a actuadores (válvulas, interruptores, etc.) IoT es también una de las tecnologías en las que se basa la **Industria 4.0**, también conocida cómo ***Smart Factories***. La Industria 4.0 es la completa digitalización de las cadenas de valor a través de la integración de tecnologías de procesamiento de datos, software inteligente y sensores. Haciendo hincapié en las **Infraestructuras Críticas**, parte de estas *Smart Factories*, tenemos servicios

esenciales para el funcionamiento de las sociedades modernas que se controlan mediante **Sistemas de Control Industrial**. Garantizar la seguridad de estos sistemas es primordial debido a las graves consecuencias que puede acarrear el éxito de un ataque en su contra. Además, la seguridad de las infraestructuras críticas está disminuyendo debido a la aparición de nuevas ciberamenazas dirigidas en contra de los sistemas **SCADA** (Supervisory Control and Data Acquisition). La conexión de todos estos sistemas críticos a Internet en consecuencia del uso de IoT y la evolución que han experimentado; el uso de componentes estándar de hardware y software o el aumento de dispositivos interconectados para reducir costes y mejorar la eficiencia, han contribuido a ello.

1.2. Objetivos

El objetivo principal del presente trabajo de fin de grado es estudiar las características del tráfico de las redes IoT mediante redes de neuronas artificiales del tipo SOM. Como resultado se proporcionará un método para establecer su línea base de funcionamiento, permitiendo al mismo tiempo disponer de un mecanismo de detección de anomalías adaptado a este entorno. Para alcanzar este objetivo principal, es necesario:

1. Recopilar diferentes *datasets* que contengan información de tráfico de redes IoT.
2. Analizar las características del tráfico relevantes para la clasificación de los flujos de tráfico.
3. Estudiar diferentes configuraciones de SOM para determinar cuáles son las que proporcionan mejores resultados
4. Comparar con otras técnicas existentes de detección de anomalías en el ámbito de IoT

1.3. Estructura de la memoria

La presente memoria ha sido estructurada en los siguientes capítulos:

1. **Introducción.**
2. **Fundamentos teóricos:** introducimos los conceptos necesarios para comprender los contenidos expuestos a lo largo de la memoria. A su vez se divide en seis secciones: Internet of Things, Industria 4.0, Sistemas de control industrial, el protocolo Modbus TCP, la Seguridad en este tipo de entornos y terminaremos el capítulo con una revisión del Estado del arte.
3. **Planificación:** detallamos la planificación del proyecto, la estimación de costes y el tiempo que transcurre desde el comienzo hasta el final de su realización.
4. **Trabajo realizado:** realizamos un estudio del entorno y características sobre las cuales se elaboró el *dataset* y, posteriormente, un análisis de los datos recolectados en las capturas del trafico de red. Además, describimos las diferentes fases del preprocesado de datos y la manera de abordar cada una de ellas
5. **Pruebas:** presentamos las métricas utilizadas para medir las diferentes pruebas realizadas y los resultados obtenidos en cada una de ellas.
6. **Conclusiones:** enumeramos una serie de reflexiones sobre el trabajo realizado, en comparación con los objetivos iniciales. También se abarcan algunas de las líneas de desarrollo más interesantes de cara a futuros trabajos relacionadas con el presente proyecto.
7. **Glosario de acrónimos y términos:** explicación de todos los términos específicos utilizados en la redacción de la presente memoria y que pueden ser desconocidos o generar duda al lector.
8. **Bibliografía:** referencias bibliográficas utilizadas a lo largo del proyecto.

Capítulo 2

Fundamentos teóricos

Este capítulo está enfocado a dar al lector una introducción teórica sobre los conceptos básicos más importantes sobre los que nos apoyaremos en esta memoria. Esto ayudará a facilitar la comprensión del contenido expuesto en los posteriores capítulos.

También, nos centraremos en la evolución sufrida en el campo de detección de anomalías en redes IoT y el actual estado del arte.

2.1. Internet of Things (IoT)

Internet evolucionó de una manera que pocos hubieran imaginado. Al principio, los avances se daban lentamente, pero hoy en día, la innovación y la comunicación se producen a gran velocidad.

Desde su humilde comienzo como ***Advanced Research Projects Agency Network***(ARPANET) en 1969, que interconectaba unos pocos **sitios**, a día de hoy se predice que Internet interconectará alrededor de veinte mil millones de **objetos** para el año 2020 [26]. En la actualidad, Internet proporciona conexiones globales que hacen posible que exista la navegación web, los medios sociales y los dispositivos móviles inteligentes.

Podemos dividir la evolución de Internet en cuatro fases distintas. Cada fase tiene un efecto más profundo en los negocios y en la sociedad que la fase anterior.

- **Conectividad**: La primera fase comenzó hace 30 años. Se basa en digitalizar el acceso a la información y destaca la aparición de servicios como el correo electrónico o la navegación web.
- **Economía interconectada**: La segunda etapa empezó a finales de la década de los 90. Aquí situamos el comienzo del comercio electrónico y de las cadenas de suministro conectadas digitalmente. Cambió la forma en la que hacemos compras y en que las empresas llegan a nuevos mercados.
- **Experiencia cooperativa**: La tercera fase comenzó a principios de la década del 2000. Esta fase se rige por el amplio uso de los medios sociales, la movilidad, los servicios de video y la computación en la nube.
- **Internet of Everything**: Esta es la fase actual. En esta se conectan personas, procesos, datos y objetos, transformando la información en acciones que crean nuevas capacidades, oportunidades y experiencias más valiosas.

2.1.1. Pilares en los que se basa IoT

Los cuatro pilares en los que se basa Iot son las **personas**, los **procesos**, los **datos** y los **objetos**.

2.1.1.1. Los objetos como un pilar

La idea de conectar objetos no es nueva. De hecho, **Internet of Things (IoT)** es un término ampliamente aceptado desde finales de la década de los 90 y hace referencia a la red de objetos físicos a los que se puede acceder mediante Internet.

Internet conecta distintos tipos dispositivos electrónicos además de las computadoras de escritorio y portátiles. A nuestro alrededor, hay dispositivos con los que quizá interactuemos todos los días y que también están conectados a Internet. Por ejemplo, día a día las personas utilizan cada vez más los dispositivos móviles para comunicarse y realizar tareas cotidianas, como revisar el pronóstico

del tiempo o realizar operaciones bancarias en línea. Entre estos dispositivos tenemos los smartphones, tablets, smartwaches, smartglasses, etc.

Es posible que muchos otros objetos también tengan conexión a Internet para que se puedan controlar y configurar de manera remota a través de Internet, tales como sistemas de seguridad, iluminación, controles de climatización, refrigeradores, hornos, calentadores de agua, coches, sensores de estado atmosférico, etiquetas de RFID, etc. Aún así, se estima que solo el 1 % de los objetos están acoplados actualmente a Internet. Para controlar dispositivos que actualmente no son conectables (objetos) a través de Internet es necesario dotarlos con sensores (para recoger información del entorno) y actuadores (para modificar el comportamiento del objeto). Estos sensores y actuadores disponen de interfaces de red que dotan de conectividad al objeto.

Los sensores son dispositivos electrónicos que permiten obtener datos de dispositivos que tradicionalmente no estaban conectados. Convierten los aspectos físicos de nuestro entorno en datos que las computadoras pueden procesar. Existen diferentes tipos de sensores, como por ejemplo un sensor que utiliza identificación por radiofrecuencia (RFID). RFID usa los campos electromagnéticos de radiofrecuencia para comunicar información entre pequeñas etiquetas codificadas (etiquetas) y un lector. En general, las etiquetas RFID se utilizan para identificar al portador, como una mascota o una caja de ropa, y hacerle un seguimiento. Debido a que las etiquetas son pequeñas, pueden fijarse a prácticamente cualquier elemento, incluidos ropa y dinero. Algunas etiquetas RFID no utilizan baterías. La energía que la etiqueta necesita para transmitir la información se obtiene de señales electromagnéticas que envía el lector de etiquetas RFID. La etiqueta recibe esta señal y utiliza parte de la energía en ella para enviar la respuesta.

Los sensores pueden programarse para que tomen mediciones, traduzcan esos datos en señales y después los envíen a un dispositivo principal denominado "controlador". El controlador es responsable de obtener los datos de los sensores y proporciona una conexión a Internet. Los controladores pueden tener la capacidad de tomar decisiones inmediatas o pueden enviar datos a una computadora más potente para su análisis. Esta computadora más potente puede estar en la misma

LAN que el controlador, o bien puede ser accesible únicamente por medio de una conexión a Internet.

2.1.1.2. Los datos como un pilar

Los datos son un valor asignado a todo lo que nos rodea; están en todas partes. Sin embargo, por sí solos, los datos no tienen sentido. Los datos se vuelven más útiles al interpretarlos, por ejemplo, mediante la correlación o la comparación. Esos datos útiles ahora son **información**. Cuando dicha información se aplica y se comprende se dice que la información se convierte en **conocimiento**.

Por lo general, las computadoras no tienen la conciencia contextual y la intuición de los seres humanos. Por lo tanto, es importante considerar los dos estados de datos siguientes: **estructurados** y **no estructurados**.

- **Datos estructurados**: Los datos estructurados son aquellos que se introducen y se mantienen en campos fijos dentro de un archivo o un registro. Las computadoras introducen, clasifican, consultan y analizan datos estructurados con facilidad. Por ejemplo, cuando enviamos nuestro nombre, dirección y datos de facturación a un sitio web, creamos datos estructurados. La estructura obliga al uso de cierto formato para la introducción de los datos, a fin de minimizar los errores y hacer que sea más fácil para la computadora interpretarlos.
- **Datos no estructurados**: Los datos no estructurados carecen de la organización de los datos estructurados; son datos sin procesar. No tienen la estructura que identifica el valor de los datos. No hay un método fijo para introducir o agrupar los datos no estructurados y, luego, analizarlos. Algunos ejemplos de datos no estructurados incluyen el contenido de fotos y archivos de audio y de vídeo.

Uno de los factores que impulsa el crecimiento de la información es la cantidad de dispositivos conectados a Internet y la cantidad de conexiones entre esos dispositivos. El volumen de datos que, hace una década, se producía en un año, ahora se produce en una semana. Eso equivale a la producción de más de 20

Exabytes (EB) de datos por semana [27]. A medida que más objetos no conectados se conectan, los datos siguen creciendo exponencialmente. Con esta cantidad de información, las organizaciones deben aprender cómo administrar los datos y también cómo tratar los datos masivos mediante técnicas de **Big Data**.

Cuando hablamos de **Big Data** nos referimos a conjuntos de datos o combinaciones de conjuntos de datos cuyo tamaño (**volumen**), complejidad (**variabilidad**) y velocidad de crecimiento (**velocidad**) dificultan su captura, gestión, procesamiento o análisis mediante tecnologías y herramientas convencionales, tales como bases de datos relacionales y estadísticas convencionales o paquetes de visualización, dentro del tiempo necesario para que sean útiles [2].

Para la administración, el almacenamiento y el análisis de Big Data, son necesarios nuevos productos y técnicas.

2.1.1.3. Las personas como un pilar

Los datos, por sí solos, no sirven de nada. Una gran cantidad de datos a la que nadie puede tener acceso no resulta útil. La organización de estos datos y su transformación en información útil permite que las personas tomen decisiones con buenos fundamentos e implementen las medidas adecuadas. Esto genera valor económico en una economía que se basa en **IoT**.

Por esta razón, las personas son uno de los cuatro pilares. Las personas son la figura central en cualquier sistema económico: interactúan como productores y consumidores en un entorno cuyo propósito es mejorar el bienestar satisfaciendo las necesidades humanas. Ya sean las conexiones de persona a persona (P2P), de máquina a persona (M2P) o de máquina a máquina (M2M), todas las conexiones y los datos generados a partir de ellas se utilizan para aumentar el valor para las personas.

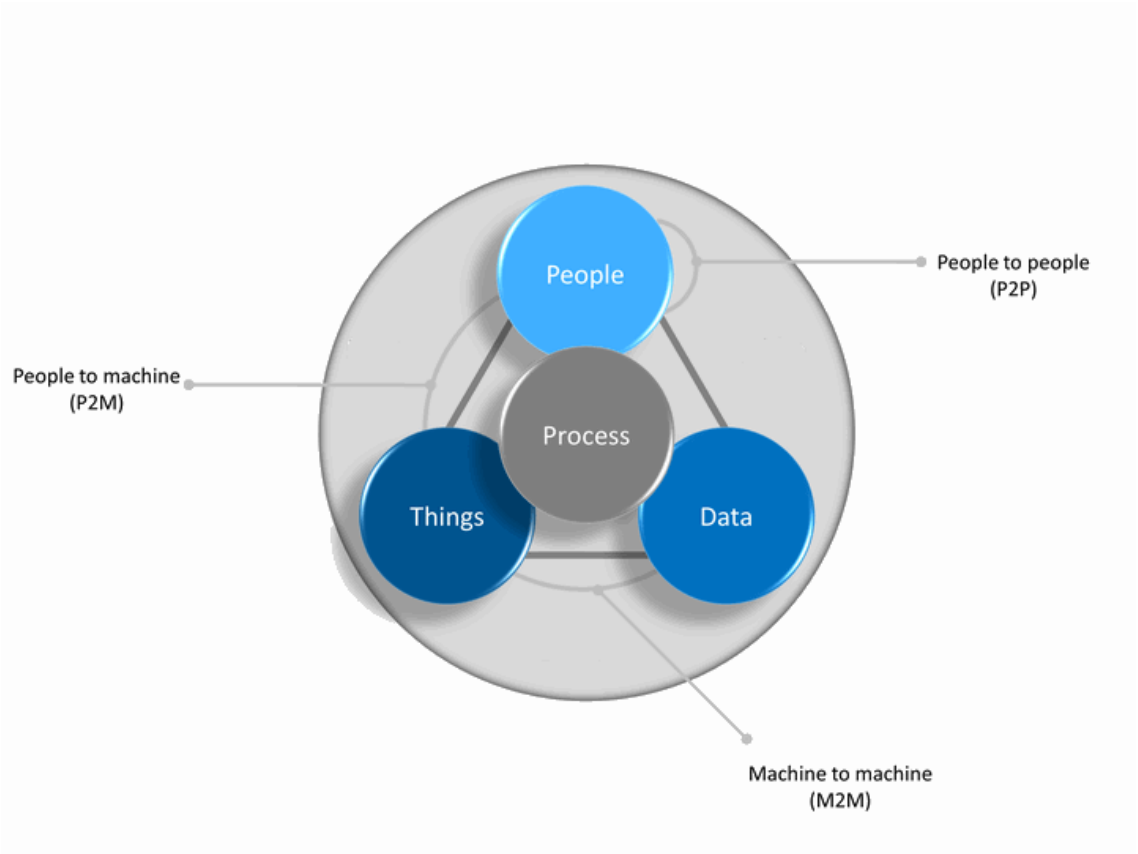


Figura 2.1: Tipos de conexiones en IoT (Cisco)

2.1.1.4. Los procesos como un pilar

Los procesos desempeñan una función fundamental en la manera en que los otros pilares -los objetos, los datos y las personas- operan juntos para ofrecer valor en el mundo conectado de IoT.

Con el proceso adecuado, las conexiones adquieren pertinencia y agregan valor, dado que se entrega la información correcta a la persona indicada en el momento adecuado y de la manera apropiada.

Los procesos facilitan las interacciones entre las personas, los objetos y los datos. En la actualidad, IoT los une mediante la combinación de conexiones de máquina a máquina (M2M), de máquina a persona (M2P) y de persona a persona (P2P).

- Conexiones M2M: Estas conexiones tienen lugar cuando se transfieren datos de una máquina u "objeto" a otro a través de una red. A menudo, el término **Internet of Things** (IoT) es considerado como sinónimo de M2M. En su más amplia conceptualización, IoT incluye cualquier tipo de objeto físico o virtual o entidad que pueda hacerse direccionable, y que tenga la capacidad de transmitir datos sin la comunicación de persona a máquina; esas son las cosas en IoT. Las cosas son, a menudo, elementos que no se han conectado en el pasado; la automatización de las comunicaciones de las cosas es también central al concepto de IoT.
- Conexiones M2P: Tienen lugar cuando la información se transfiere entre una máquina (como una computadora, un dispositivo móvil o un letrero digital) y una persona. Cuando una persona obtiene información de una base de datos o realiza un análisis complejo, tiene lugar una conexión M2P. Estas conexiones facilitan el movimiento, la manipulación y la información de datos de máquinas para ayudar a las personas a que tomen decisiones fundadas.
- Conexiones P2P: Ocurren cuando la información se transfiere de una persona a otra. Las conexiones P2P se producen cada vez más a través de vídeo, dispositivos móviles y redes sociales.

Como se muestra en la figura 2.2, el valor más alto de IoT se obtiene cuando el proceso facilita la integración de las conexiones M2M, M2P y P2P. [27]

2.2. Industria 4.0

El término **Industria 4.0** ha sido acuñado y desarrollado mayormente en Alemania, y se corresponde con una nueva manera de organizar los medios de producción. Los procesos industriales de hoy exigen una alta conectividad entre sus componentes sin renunciar a los requerimientos básicos de continuidad de negocio y alta disponibilidad. Por ello, es necesario crear nuevos procesos de fabricación inteligentes capaces de una mayor adaptabilidad a las necesidades y a los procesos de producción, así como a una asignación más eficaz de los recursos. También tendrá un efecto en la forma en que se producen las cosas, provocando otra revolución industrial. Con la primera revolución siendo la introducción de la

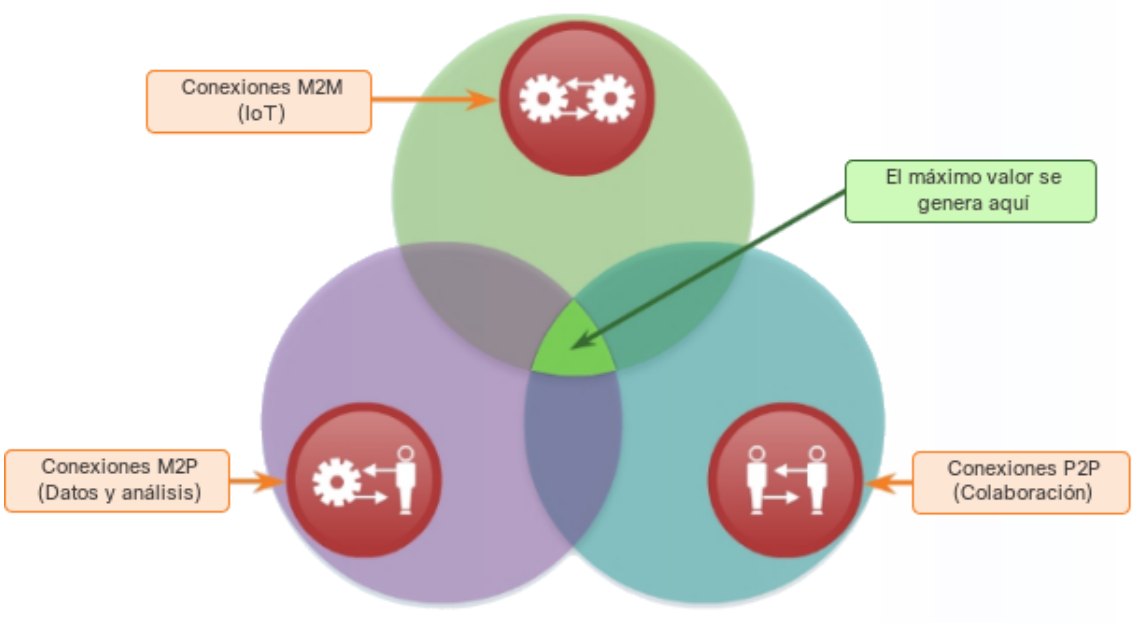


Figura 2.2: Generación de valor en IoT (Cisco)

máquina de vapor, la segunda trayendo la producción en serie en la línea de ensamblaje y la tercera el establecimiento de la robótica y automatización industrial (entre otras cosas), es ahora el momento para la cuarta revolución industrial.

La cuarta revolución industrial está allanando el camino hacia la **Smart Factory** (fábrica inteligente). Esta se basa en sistemas **ciber-físicos (CPS)**, máquinas y componentes interconectados con la inclusión de software "inteligente" y altamente flexible. Los sistemas ciber-físicos están intrínsecamente conectados con sistemas embebidos, que son partes de dispositivos completos que llevan a cabo funciones muy específicas que con frecuencia vienen con restricciones de computación en tiempo real. Los CPS enlazan estos sistemas embebidos con redes digitales facilitando el procesamiento independiente de datos. La asignación de una dirección IP permite a esos sistemas ser controlados y monitorizados en línea.

Debido a estos sistemas informáticos embebidos, sensores y actuadores, los sistemas ciber-físicos organizan la producción de forma automática y autónoma. El control central del proceso puede ser eliminado ya que puede ser asumido por

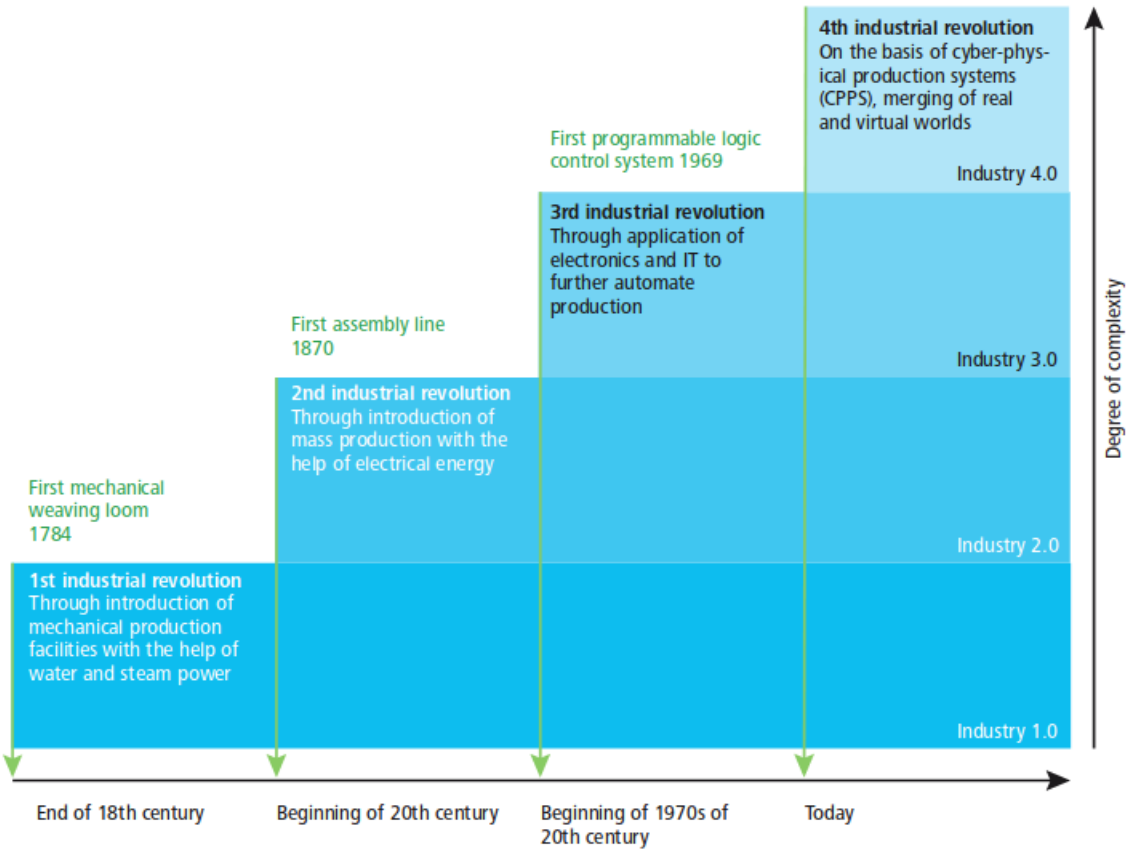


Figura 2.3: Revolución Industrial (Deloitte)

componentes basados en CPS. Este concepto de organización de la cadena de valor también se conoce como **Industria 4.0** o la **cuarta revolución industrial**.

IoT facilita los procesos de comunicación y cooperación de los sistemas ciber-físicos. Las tecnologías IoT más comunes utilizadas en la Industria 4.0 y, en consecuencia, en las fábricas inteligentes son, entre otras, redes inalámbricas, objetos "inteligentes" y elementos de sensorización y actuación [11], [14], [18].

2.3. Sistemas de Control Industrial

Un **Sistema de Control Industrial (SCI)** tiene por objetivo controlar de manera automática un proceso industrial (antes controlado manualmente, cabe aclarar que todavía existen en muchas fabricas) sin tener que depender de la

reacción o interpretación de un operador sobre una medida. Los procesos industriales manuales dependen en gran parte de la experiencia del operador, pero agregan variables difíciles de controlar: el nivel de cansancio, su estado de ánimo, las frecuencias en que el cerebro humano puede tener huecos o vacíos de atención ante un proceso de alta concentración, etc. La automatización de estos temas no es algo novedoso, los sistemas de control industrial existen desde hace muchísimos años y funcionan bien.

Uno de los SCI más desplegados son los sistemas **SCADA** (Supervisory Control And Data Acquisition). Los sistemas SCADA se utilizan para monitorizar y controlar infraestructuras críticas de toda clase, basándose en el uso de protocolos de comunicación, sistemas de sensores y mecanismos de computación. Su uso permite la intercomunicación y el control de todos los elementos que componen el sistema, posibilitando la compartición de datos y el desencadenamiento de acciones si es necesario. Este tipo de protocolos se ha venido desarrollando desde los años 60 en aplicaciones típicas del ámbito industrial, siendo especialmente importante en los sistemas de distribución de electricidad y agua, en la industria energética y en los sistemas de telecomunicaciones.

El catálogo de protocolos SCADA disponibles ofrece muchas alternativas, tales como DNP3, Modbus, OPC o ICCP entre otros. Todos estos protocolos son diferentes entre sí y están orientados a distintas aplicaciones. Sin embargo, tienen una serie de características en común:

- **Control de acceso:** Los usuarios se encuentran dentro de grupos con privilegios de lectura y escritura específicos sobre los parámetros del proceso y las funcionalidades del producto.
- **Gestión de alarmas:** Los sistemas SCADA disponen de alarmas cuyo manejo se basa en la comprobación del estado en los diferentes nodos y en la comparación con un valor límite. Las alarmas se gestionan de manera centralizada, de tal forma que la información que manejan solo existe en un lugar y todos los usuarios ven el mismo estado. La mayoría de implementaciones permiten establecer distintos niveles de alarma y generar

un aviso, ya sea por correo electrónico u otro servicio, en respuesta a la información procesada en un evento.

- **Archivado de *logs*:** Una de las funcionalidades básicas de estos protocolos es el archivado de *logs*. Durante el tiempo de ejecución de un programa SCADA se van guardando datos relevantes de la comunicación acompañados de una marca de tiempo y un identificador que permite su filtrado y estudio posterior.
- **Generación de informes:** La mayoría de las aplicaciones SCADA disponen de herramientas de reporte compatibles con hojas de cálculo o sistemas de bases de datos.
- **Automatización:** Una de las capacidades más potentes de SCADA es la de ejecutar acciones automáticamente cuando se produzca un evento determinado. Estos desencadenantes se programan mediante lenguajes de scripting propios de cada protocolo y pueden dar lugar a secuencias de acción muy complejas en uno o varios dispositivos del sistema.

Los sistemas SCADA tienen una arquitectura que puede diferir en función del protocolo utilizado y de la aplicación para la que han sido diseñados, pero todos pueden englobarse dentro del modelo que puede verse de manera esquemática en la Figura 2.4. La función de cada una de las partes se pormenoriza en los siguientes apartados [24].

2.3.1. Unidad Terminal Remota

La **Unidad Terminal Remota** (**RTU**, por sus siglas en inglés, Remote Terminal Unit), es el dispositivo que, desplegado a lo largo de toda la infraestructura de red, se ocupa de adquirir información útil para su monitorización y control. Estos aparatos, que en la jerga tradicional de las redes se consideran como **esclavos**, adquieren datos de su entorno y los traducen al lenguaje o protocolo utilizado en la comunicación SCADA para su envío a un centro de procesamiento, también denominado **maestro**.

Podemos diferenciar los RTU en función de la aplicación que desarrollan, siendo los más usuales los siguientes tipos:

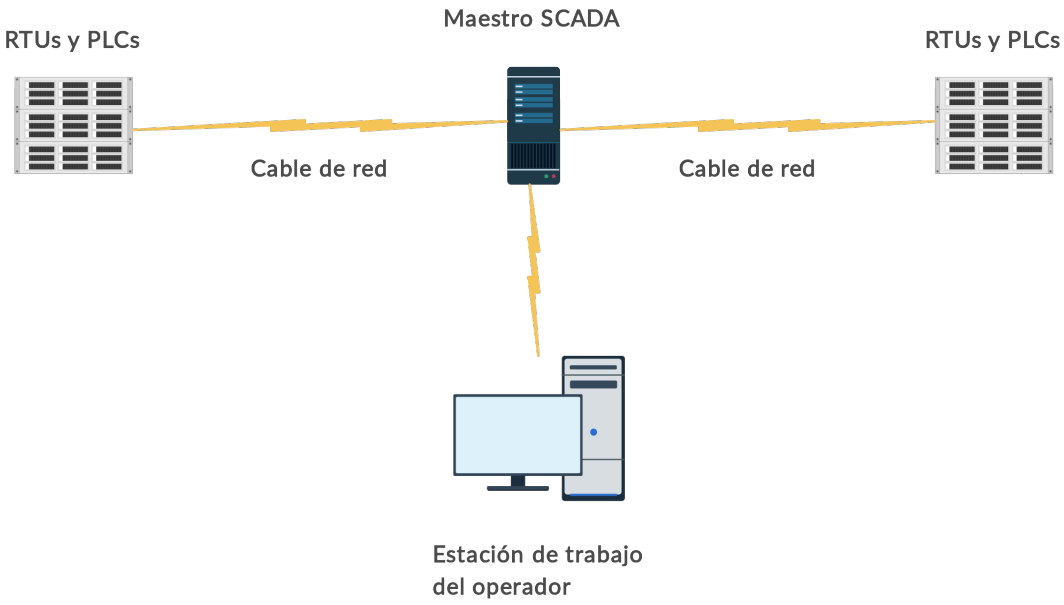


Figura 2.4: Esquema de la arquitectura SCADA

- **Módulos analógicos de entrada/salida:** Se ocupan de la detección y la actuación en el ámbito de las variables continuas. Los módulos de entrada tienen un número de interfaces cuyo valor típico puede ser de 4, 8, 16 o 32. Los módulos de salida toman valores digitales de la CPU y lo convierten a representaciones analógicas para su envío a los actuadores.
- **Módulos digitales de entrada/salida:** Estos módulos se utilizan en el display de estados y en la gestión de señales de alarma.

Con frecuencia los RTU son instalados en sitios remotos y por esta razón se necesita que tengan algunas características tales como: capacidad de comunicación a grandes distancias (radio modem o GSM/SMS entre otras), bajo consumo, capacidad para almacenar datos, funcionamiento en amplio rango de temperatura, alimentación DC, etc.

A modo de ejemplo, un RTU colocado en un transformador eléctrico podría tomar valores de voltajes o corrientes instantáneas, empaquetarlo en un mensaje de un protocolo de red y enviarlos al maestro para su procesamiento.

2.3.2. Controlador Lógico Programable

Los **Controladores Lógico Programables** (**PLC**, por sus siglas en inglés, Programmable Logic Controller), son los dispositivos que se utilizan para automatizar y controlar procesos electromecánicos, desencadenando acciones físicas como puede ser mover un pistón, abrir una válvula o encender una turbina. Esta clase de dispositivos también se consideran **esclavos**. Estos aparatos, surgidos como sustitutos del relé, se han venido utilizando en la industria en los últimos cuarenta años y están diseñados para trabajar en condiciones térmicas, electromagnéticas y mecánicas desfavorables. A día de hoy, los PLCs se han ganado el papel de estándar en hardware en aplicaciones de control.

Aunque para respetar los tecnicismos, en esta revisión se han separado en dos categorías distintas a los RTU de los PLC, lo cierto es que en los últimos años las características técnicas de ambos dispositivos se han ido solapando y a día de hoy es difícil establecer un límite claro entre los dos.

2.3.3. Unidad Terminal Maestra

La **Unidad Terminal Maestra** (**MTU**, por sus siglas en inglés, Master Terminal Unit), hace la función de servidor central de todo el sistema, recibiendo datos de los RTU, extrayendo y almacenando la información relevante para la aplicación y procesando los datos de forma que sean entendibles para un operador humano. Desde el MTU también se gestionan las órdenes que los PLC deberán ejecutar en el mundo físico, siguiendo la política de eventos que el sistema tenga configurada.

La comunicación entre MTU y RTU o PLC es bidireccional, lo que quiere decir que los esclavos podrán en la mayoría de los casos iniciar una comunicación con el maestro sin que este lo haya hecho previamente. A pesar de esto, la mayor parte del tráfico en las redes SCADA se basa en peticiones de datos -lecturas- que el MTU solicita al RTU, por lo que se suele denominar como **maestro** al **MTU**. Dependiendo del protocolo SCADA utilizado en el sistema puede haber uno o varios maestros, lo que puede derivar en distintas topologías de red.

2.3.4. Red de Comunicaciones

La **red de comunicaciones** de un sistema SCADA se encarga de asegurar la transmisión de los distintos mensajes y datos que se generan entre los elementos que conforman la red, como los RTU o los PLC. La gran parte de las implementaciones industriales utilizan un soporte físico como el cable por tener una robustez mayor frente a fenómenos electromagnéticos adversos aunque también pueden establecerse otros tipos de comunicación, como el enlace inalámbrico.

Desde sus orígenes las redes SCADA han evolucionado desde un modelo de comunicación típicamente pensado para puerto serie a uno IP, debido en gran parte a la influencia de los conceptos de IoT. Esta evolución proporciona a los sistemas SCADA de una flexibilidad mayor, pero a su vez implica la existencia de nuevas oportunidades disponibles para los hackers y es sin duda una de los motivos que ha elevado el número de ciberataques sobre estos sistemas.

2.3.5. Interfaz Hombre Máquina

El operador humano necesita tener acceso en todo momento a información relativa al sistema que está administrando. Para ello, es preciso que disponga de una máquina con conexión remota al MTU y que esta sea capaz de mostrarle los datos en un formato amigable.

Para ello se emplea un sistema de **Interfaz Hombre Máquina** (**HMI**, por sus siglas en inglés, Human-Machine Interface), que muestra al operador información en tiempo real y generalmente de forma gráfica. Las funciones más importantes de los sistemas HMI se listan a continuación:

- Dar accesibilidad a la información técnica de alta complejidad a personal que disponga de pocos conocimientos técnicos simplificando el tratamiento de la información.
- Monitorizar la disponibilidad del sistema.
- Mantener un histórico de los parámetros de la red.

- Ayudar a los procesos de depuración y corrección de errores aportando información útil.
- Permitir el análisis del sistema para determinar potenciales acciones que mejoren su rendimiento [25].

2.4. Modbus TCP

Diseñado inicialmente por **Modicon** a finales de los años 70, **Modbus** es un protocolo de comunicaciones serie para conectar dispositivos industriales. Más adelante se convirtió en un estándar abierto gestionado por la organización Modbus [6].

Modbus es un protocolo maestro/esclavo, es decir, solo uno de los hosts que participan en la comunicación, denominado maestro, puede iniciar el intercambio de mensajes. Por cada solicitud realizada por el maestro, el esclavo responde con una respuesta normal o con una excepción, indicando que ha ocurrido algún error. Un esclavo nunca envía un mensaje, a menos que sea solicitado por el maestro.

Más recientemente, Modbus se ha ampliado para soportar la pila TCP/IP (Modbus TCP). Nótese que, incluso en su versión TCP/IP, Modbus sigue siendo un protocolo maestro/esclavo. En este caso, el esclavo se denomina "servidor". El servidor escucha las peticiones TCP entrantes en el puerto 502. El maestro, denominado "cliente", abre activamente las conexiones TCP. En una conexión TCP dada, los roles maestro y esclavo son fijos, sin embargo, pueden ser invertidos en otra conexión entre los mismos hosts.

La cabecera Modbus TCP se muestra en la Tabla 2.1. Consiste en los siguientes campos [4]:

- **Transaction identifier** es usado para hacer coincidir cada respuesta con su consulta.
- **Protocol identifier** siempre se fija a 0 para los mensajes Modbus. Los otros valores se reservan para futuras ampliaciones.

- **Length field** es el tamaño en bytes de todos los campos a su derecha, es decir, unit identifier, function code y data.
- **Unit identifier** se utiliza normalmente para comunicarse con varios esclavos serie (es decir, no compatibles con TCP/IP) a través de un gateway TCP/IP. Cada uno de los llamados "esclavos encadenados" puede ser direccionado a través de su unit identifier único. En caso de que no se utilice esta función, el campo se debe establecer a 0.
- **Function code** identifica diferentes operaciones, como la lectura y la escritura. El estándar define tres tipos de códigos de función:

  i) **Códigos de función públicos** que tienen su significado definido en el estándar.

  ii) **Códigos de función definidos por el usuario** que pueden ser utilizados por los proveedores para operaciones no estándar.

  iii) **Códigos de función reservados** que se conservan para operaciones existentes.

- **Data** incluyen el contenido del mensaje y su formato y tamaño específicos dependen del código de función. Por ejemplo, una solicitud de lectura de un registro específico implica indicar el campo inicial que se va a leer y la cantidad de registros que se van a solicitar, ya sea uno o más. La respuesta contendrá para cada registro solicitado un campo con el valor correspondiente a dicho registro.

Transaction Identifier	Protocol Identifier	Length Field	Unit Identifier	Function Code	Data
2 Bytes	2 Bytes	2 Bytes	1 Byte	1 Byte	252 Bytes (Máximo)

Tabla 2.1: Unidad de datos del protocolo Modbus (Protocol Data Unit, PDU)

La mayoría de los códigos de función públicos se refieren a operaciones de lectura y escritura en los diferentes tipos de datos definidos en el estándar, y a funciones de diagnóstico. La única excepción es el código de función 43, que se utiliza para encapsular otros protocolos a través de Modbus.

Modbus define solo cuatro tipos de datos básicos (Tabla 2.2); *discrete inputs*, *coils*, *input registers* y *holding registers*. Los dos tipos de "inputs" son de solo lectura, mientras que los dos restantes también pueden ser escritos por el cliente. Los dos tipos de "register" tienen una longitud de 16 bits, mientras que los dos restantes son de un solo bit.

Nombre	Acceso	Tamaño
Discrete Input	solo - lectura	1 bit
Coil	lectura - escritura	1 bit
Input Register	solo - lectura	16 bits
Holding Register	lectura - escritura	16 bits

Tabla 2.2: Tipos de datos Modbus

Cada tipo de datos se trata por separado por medio de un campo de 16 bits, lo que permite 65.536 entradas para casa tipo de datos. Diferentes códigos de función operan con diferentes tipos de datos. Por ejemplo, el código de función 1 lee los *coils* y el código de función 2 lee las *discrete inputs*. Las operaciones de lectura se definen sobre una secuencia contigua de valores (hasta 2.000 para los tipos de 1 bit y 125 para los tipos de 16 bits). Para leer un valor, el cliente determina la dirección de inicio y la cantidad total de valores que se deben leer. También es posible escribir varios valores, pero en ese caso se utilizan diferentes tipos de función. Por ejemplo, el código de función 5 escribe un solo *coil* mientras que el código de función 15 escribe varios *coils*. Hay que tener en cuenta que la asignación de direcciones depende completamente del proveedor.

Por último, decir que Modbus no proporciona ninguna característica de seguridad. Los mensajes se intercambian sin ningún tipo de cifrado (en texto plano) y no se realiza ninguna autenticación; las peticiones realizadas desde cualquier origen serán aceptadas por un servidor Modbus [23].

2.5. Seguridad en las redes OT

Los sistemas SCADA se desarrollaron originalmente utilizando dispositivos embebidos con fines específicos, que se comunicaban con protocolos propietarios.

Estos fueron diseñados para operar de forma aislada a otros sistemas, por lo que los vendedores y aperadores creían que podían confiar en este aislamiento para su protección. Es decir, como la red SCADA estaría físicamente aislada de cualquier otra red, los atacantes no podrían acceder a ella (**separación entre la red de control y la red de datos**). Además, los proveedores y los operadores confiaban en la seguridad a través de la oscuridad, o sea, la creencia de que la falta de información pública disponible sobre los sistemas SCADA los hacía seguros.

Sin embargo, en los despliegues modernos de redes SCADA observamos que la interconexión con otros sistemas se está convirtiendo en algo común. Se puede acceder a las redes SCADA desde las redes corporativas, los ingenieros tienen acceso remoto a través de **Redes Privadas Virtuales** (VPN, por sus siglas en inglés, Virtual Private Network) e incluso, en algunos casos, es posible el acceso desde Internet. Además, se están substituyendo los dispositivos empotrados de propósito especial por los denominados dispositivos **COTS** (Commercial Off-The-Shelf, que en su traducción literal significa Componente sacado del estante) y el conjunto de protocolos de comunicación TCP/IP. Todos estos cambios tienen un **profundo impacto** en la seguridad de las redes de los **Sistemas de Control Industrial** y entre ellos los sistemas **SCADA**.

2.5.1. Diferencias entre las Tecnologías de la Operación (OT) y las Tecnologías de la Información tradicionales (IT)

Los requisitos de seguridad para las redes de IT tradicionales se resumen como la terna: **Confidencialidad**, **Integridad** y **Disponibilidad**. De hecho, para una empresa típica, estos requisitos están ordenados en importancia:

- Confidencialidad: La propiedad de un sistema informático por el cual su información se divulga solo a las partes autorizadas, tiene la más alta prioridad.
- Integridad: La característica de que las alteraciones en los activos de un sistema pueden ser de una manera autorizada, sigue como segunda prioridad.

- Disponibilidad: La prioridad de que un sistema está listo para ser utilizado inmediatamente, tiene la más baja prioridad.

La seguridad tiene la más alta prioridad en un entorno SCADA. Mientras que un entorno TI tradicional las interrupciones normalmente solo causan pérdidas monetarias, en los sistemas SCADA, una interrupción puede suponer un riesgo para la seguridad humana y pública, así como daños a los equipos. Por lo tanto, el orden de prioridad se invierte: la disponibilidad tiene la mayor prioridad; la integridad se acerca lo más posible, ya que la operación segura depende de la veracidad de los datos; y la confidencialidad tiene menor prioridad, ya que los datos deben ser analizados dentro del contexto para tener valor.

Además de las diferentes prioridades, existen otras diferencias clave entre estos entornos, entre ellas:

- **Recursos limitados**: Los equipos SCADA suelen tener recursos limitados, lo que supone un reto para la implementación de algoritmos criptográficos tradicionales desarrollados para entornos IT. Este problema se ve agravado por los requisitos en tiempo real de las aplicaciones SCADA.
- **Información frente a activos**: Los entornos IT tienen como prioridad la protección de la información durante el almacenamiento, la transmisión y la computación. Sin embargo, en los entornos OT, el enfoque se centra en la protección de los dispositivos de la red de campo, por ejemplo, PLCs y RTUs, ya que son los responsables de controlar el proceso físico. Proteger el servidor OT también es importante ya que puede influir en los dispositivos de control.
- **Problema de actualización de software**: La aplicación de parches de seguridad es uno de los principales mecanismos para evitar la explotación de las vulnerabilidades de los dispositivos usados en los entornos IT. Sin embargo, esta práctica no es común en entornos OT, ya que cada cambio en los sistemas requiere pruebas exhaustivas.
- **Vida útil de los componentes**: La vida útil de los componentes OT es normalmente mucho más larga que la de los componentes de los entornos IT. Los cambios en IT son rápidos, del orden de 2 a 5 años, debido a la

	TO	TI
Objetivo principal	Control de equipos físicos	Procesamiento y transferencia de datos
Gravedad de fallo	Alta	Baja
Tiempo ida-vuelta (RTT)	250μ - 20ms	50ms +
Determinismo	Alto	Bajo
Composición de los datos	Pequeños paquetes de tráfico periódico y aperiódico	Paquetes grandes y aperiódicos
Entornos de funcionamiento	Duros, a veces adversos (ruido, polvo, vibraciones...)	Limpios
Vida del sistema	10 - 20 años	2 - 5 años
Complejidad de nodos	Baja	Alta

Tabla 2.3: Diferencias entre TO y TI

constante innovación. Por el contrario, los cambios en OT son más lentos, del orden de 15 a 20 años, a medida que se desarrollan soluciones para escenarios específicos. Conectado al problema de los parches, esta larga vida útil suele dar como resultado componentes antiguos que están operativos pero que ya no son mantenidos por los proveedores.

- **Determinismo**: En las redes OT el tráfico de red es generado por procesos automáticos (PLCs, RTUs) cada medio segundo, cada segundo, de forma continuada y, aunque cada cierto tiempo el operador también tendrá que realizar alguna operación manual, este tipo de redes son mucho más estáticas y deterministas que las redes IT.

En la Tabla 2.3 podemos ver un resumen de las diferencias más importantes entre los entornos TI y TO [9] [16].

2.5.2. Incidentes de seguridad en los últimos años

En los últimos años se han detectado multitud de ciberataques a infraestructuras críticas, las cuales siempre están en el punto de mira de gobiernos, cibercriminales y hacktivistas.

La evolución de los ciberataques ha ido desde el típico malware orientado a los sistemas IT con distintos fines, hasta las grandes operaciones contra

infraestructuras críticas donde se utiliza malware modular avanzado que se aprovecha de vulnerabilidades en protocolos industriales. Este tipo de software malicioso se denomina amenaza persistente avanzada, más conocida por sus siglas en inglés **APT** (Advanced Persistent Threat). Esto lleva a suponer que, en los próximos años, los ciberataques contra la industria y servicios críticos irán en aumento, por lo que es necesario contar con mecanismos y contramedidas para proteger los sistemas críticos. En la Figura 2.5 podemos ver las amenazas más destacas en contra de los SCI de los últimos años y los vectores de infección junto con las técnicas utilizadas para la explotación de estos sistemas.

La detección de estos ataques es de extrema dificultad y constituyen uno de los retos más importantes a los que se enfrentan los profesionales de seguridad.

En muchas ocasiones, se utilizan vectores de ataque únicos y novedosos, capaces de evadir los sistemas de defensa tradicionales e incluso es posible que permanezcan ocultos durante largos periodos de tiempo dentro de las redes de control sin que ningún sistema de seguridad alerte sobre ello.

Sin embargo, como se puede ver en la Figura 2.5, uno de los principales medios de infección es mediante correo electrónico (spear-phishing) o empleando dispositivos externos como USB infectados con malware. Esto significa que, una vez más, el vector de infección afecta al eslabón más débil, las personas. Los agentes maliciosos aprovechan este hecho para aumentar las probabilidades de tener éxito, realizando ataques dirigidos a personas que trabajan en la compañía.

Partiendo de la base que **no podemos proteger una compañía al 100 % de ciberataques** [1] debemos hacer uso de un conjunto de técnicas, conceptos y sistemas de seguridad que nos ayuden a reducir al máximo la probabilidad de ser comprometidos y en el caso de serlo, detectarlo lo antes posible. Algunas de las medidas que se pueden adoptar son las siguientes:

- Aplicar ciberseguridad de manera transversal a toda la organización, desde formación específica de seguridad a todos los empleados de la empresa,

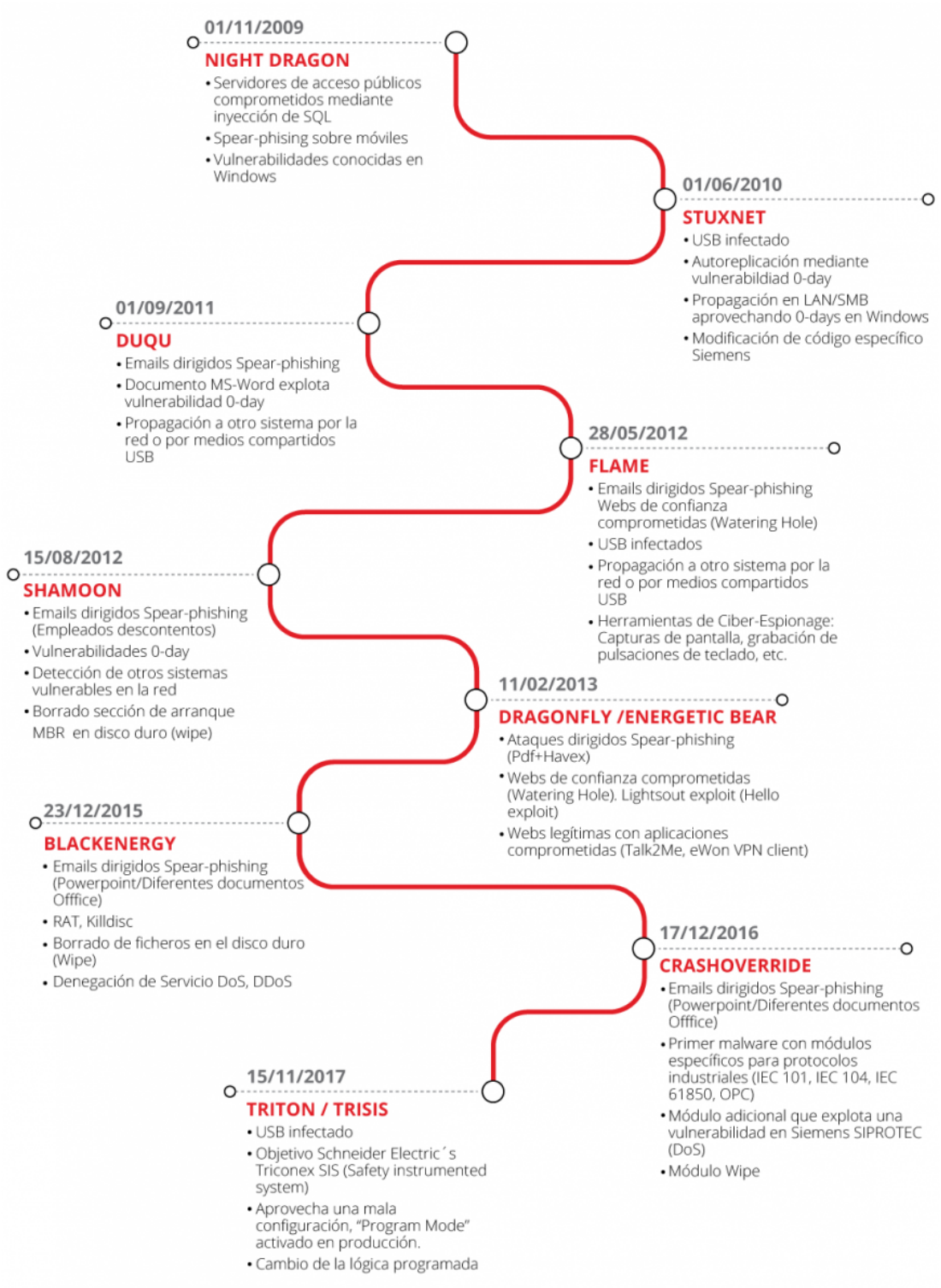


Figura 2.5: Principales ataques conocidos en contra de los sistemas de control industrial [19]

concienciación, así como conceptos y técnicas de seguridad como defensa activa y threat intelligence, ya que es clave para prevenir futuros ataques.

- Compartir información, y lo más importante conocer cómo tratarla, para convertirla en medidas de seguridad aplicable contra los vectores de ataque utilizados por los diferentes actores en el pasado, debería ser parte de la ciberseguridad industrial.

- Igualmente, es esencial el correcto bastionado de dispositivos, monitorización, segmentación y control de los sistemas y redes, accesos de todas las personas, etc., mediante los correspondientes análisis de valoración de riesgos.

En el presente trabajo de fin de grado estudiaremos las características del tráfico de este tipo de redes mediante redes de neuronas artificiales del tipo SOM. Como resultado se proporcionará un método para establecer su línea base de funcionamiento, permitiendo al mismo tiempo disponer de un mecanismo de detección de anomalías adaptado a este entorno y que permita dar un extra de seguridad a este tipo de redes [7], [19].

2.6. Estado del arte

En este apartado vamos a exponer el estado de la investigación actual en el campo de la detección de ataques en redes IoT y más concretamente, en el subconjunto formado por las redes industriales.

Hay un hecho importante que aumenta el interés de los investigadores en este campo y es el descubrimiento de **Stuxnet** (2010). Podemos considerar a Stuxnet el ataque que marcó un antes y un después tanto en temas referentes a la investigación en entornos SCI como la concienciación sobre la necesidad de seguridad especifica en este tipo de redes.

Stuxnet es un gusano informático que afecta a equipos con Windows, descubierto en junio de 2010 por **VirusBlokAda** [5], una empresa de seguridad ubicada en Bielorrusia. Se dice que es la pieza de software, y en este caso de

malware, más sofisticada jamás escrita ya que explotaba 4 vulnerabilidades de día 0 (0-days en inglés) con el objetivo de sabotear ciertas centrifugadoras de Uranio en plantas nucleares de Irán. Lo realmente peligroso de este software es que a los operadores no se les mostraba el estado real del proceso. Las centrifugadoras de las plantas infectadas giraban a más velocidad de la que debieran para desgastarse, pero los datos que se observaban a través de la HMI eran completamente correctos, se mostraban datos erróneos que hacían creer a los operadores que esas centrifugadoras funcionaban correctamente. El mal uso de estos dispositivos originó que estos se averiasen. Esto produjo un problema bastante grande para el programa nuclear iraní.

La complejidad de Stuxnet se puede ver en los ataques posteriores realizados en contra de este tipo de entornos (Figura 2.5) ya que la gran mayoría de ellos utilizan módulos de software desarrollados para Stuxnet.

2.6.1. Estado de la investigación antes de Stuxnet (2010)

Antes de Stuxnet ya se trabajaba en la detección de ataques en redes SCI pero el campo de investigación era bastante limitado comparado con hoy en día o con las redes IT del momento.

La gran parte de las soluciones de seguridad adoptadas consistían en aplicar las ya existentes del campo IT y trasladarlas al entorno industrial, como por ejemplo la utilización de IDS basados en firmas. Por otra parte, la mayor parte de la investigación del momento eran pruebas de concepto, no se llevaban a cabo en producción porque no había o no se sentía una necesidad real por parte de la industria. También existía algún workshop dedicado al tema pero todo de forma muy aislada.

En el trabajo realizado por Zhu y Sastry [28], en el que realizan una revisión de los trabajos llevados a cabo hasta entonces, se observan ya algunos términos que actualmente toman más importancia, entre los que destacamos los siguientes:

- **Detección basada en modelos:** Este enfoque se fundamenta en construir un modelo de lo que debería ser la red o el proceso industrial en sí mismo y así poder detectar *cosas* que se salgan del modelo construido.
- **Detección basada en especificaciones:** Esta aproximación se basa en la creación de reglas simples, **administrativamente**, indicando comportamientos que no deberían producirse, ya que podrían estar ocasionado por actores maliciosos. Esta es una solución bastante más simple que la detección basada en modelos.

Además,la mayoría de las buenas practicas que se utilizan hoy en día no existían o estaban muy poco desarrolladas hace apenas 10 años, como la segmentación de la red IT/OT, la utilización de firewalls industriales o las listas blancas, altamente recomendadas a día de hoy.

2.6.2. Enfoques actuales

Lo anteriormente expuesto es el estado de investigación en el campo hasta 2010. Con el el descubrimiento de Stuxnet y su publicación en prensa, se genera un miedo y una concienciación hacia los ciberataques en contra de este tipo entornos, ya que pueden perturbar y dañar las operaciones críticas de infraestructura, contaminar el medio ambiente, causar grandes pérdidas económicas y, lo que es aún más peligroso, causar la perdida de vidas humanas.

Desde entonces, este campo se convierte en un área de investigación más de moda de lo que fue anteriormente, se empiezan a publicar más, se crean más workshops y conferencias dedicadas a la ciberseguridad industrial, nuevos enfoques y productos finales para intentar prevenir, detectar y mitigar ataques, etc.

Actualmente, existen dos corrientes principales en las que se centra la mayor parte de trabajos, y son **la detección de ataques a nivel de red** y la **detección de ataques a nivel de campo**.

2.6.2.1. Detección de ataques a nivel de red

Esta aproximación se basa en observar cómo se comporta la red, detectando paquetes o tráfico que no cumplen ciertas reglas, características, comportamientos extraños, etc. En este apartado nos encontramos con los IDS (Intrusion Detection Systems), sus variantes y las diferentes técnicas de detección empleadas.

Existen multitud de métodos para realizar la detección de intrusiones. Una posibilidad son los **sistemas basados en firmas (signature-based IDS)**, que se centran en buscar patrones específicos en las tramas transmitidas por la red, aunque por ese motivo les sea imposible detectar nuevos tipos de ataques cuyo patrón desconozcan [35].

Otra posibilidad son los **sistemas de detección basados en anomalías(anomaly-based IDS)**, que comparan el estado actual del sistema y sus datos generados con el comportamiento habitual del mismo, para identificar desviaciones que lleven a pensar en una posible intrusión. No obstante, en el contexto de los sistemas de control, hay que tener en cuenta restricciones tales como la heterogeneidad de los datos recogidos en un ambiente industrial en producción, el ruido presente en las mediciones, y la naturaleza de las anomalías (ataques vs. averías).

Por este motivo, se han propuesto numerosas técnicas de detección procedentes de ramas como la estadística o la inteligencia artificial [6], cada una con un nivel de adaptación distinto según el escenario de la aplicación a proteger[20].

Entre estas técnicas destacan las **técnicas de minería de datos (Data mining-based detection)** encontrando en esta categoría *técnicas de clasificación*, *clustering* y *técnicas basadas en reglas de asociación*; **técnicas estadísticas (statistical anomaly detection)** como *modelos paramétricos y no paramétricos*, *análisis de series temporales* o *técnicas de detección basadas en la información*; **técnicas basadas en el conocimiento (knowledge based detection)** ejemplos de estas técnicas incluyen las *técnicas basadas en transiciones de estados*, *redes de Petri* o *sistemas expertos*; **técnicas de aprendizaje automático**

(machine learning based detection) entre las que se encuentran las *redes de neuronas artificiales*, *redes bayesianas*, *máquinas de vectores soporte*, *lógica difusa* o *algoritmos genéticos* [12].

Un trabajo interesante desde el punto de vista personal y que no se halla dentro de ninguna de las categorías anteriormente comentadas, es el realizado por *Iturbe, Mikel et al.* [15]. Ante la ausencia de herramientas de visualización de seguridad para redes industriales, en este proyecto se crea un útil basado en diagramas de cuerdas, como podemos ver en la Figura 2.6

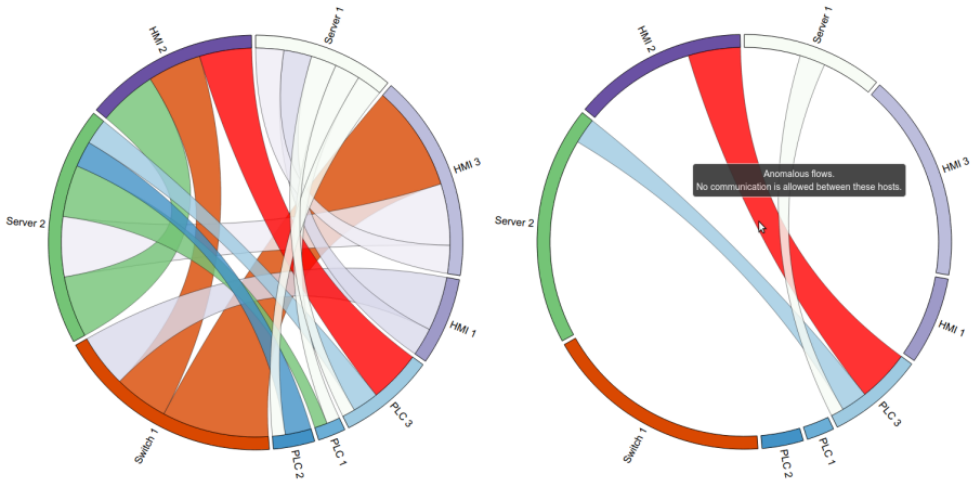


Figura 2.6: Representación de un flujo de red anómalo

La manera de interpretarla es la siguiente: En el circulo están representados todos los host industriales pertenecientes a la red de forma ordenada (PLCs, HMIs, Servidores, Switches, etc.). Como ya hemos comentado en el apartado 2.5.1, debido al determinismo de las redes industriales si un host se comunica siempre con otro host, este comportamiento no va a cambiar, por ejemplo, si un PLC tiene 2 servidores de control asociados va a enviar la información a esos servidores, no tiene que enviar información a otro dispositivo (dentro o fuera de la red) ya que el PLC es un proceso automático y estático. Aprovechándose de esta característica de este tipo de redes, se representan los flujos de red. Las cuerdas que se observan en la figura representan a los flujos de red entre los distintos dispositivos y la anchura de la base de estas cuerdas muestra el número de paquetes en cada cuerda, es

decir, cuerdas más anchas flujo con más cantidad de paquetes y cuerdas más estrechas flujos menos activos.

Por último se debe elaborar una lista blanca de que dispositivos a los que se les permita comunicarse entre si y todo lo que no cumpla esos parámetros se señala en rojo. A modo de ejemplo, en el Figura 2.6, en el diagrama de cuerdas de la derecha se puede observar un flujo no permitido entre un PLC y un HMI.

En este trabajo también se han elaborado otros casos de uso como la detección de ataques de denegación de servicio (DOS) y no solo la existencia de flujos anómalos, si no la ausencia de flujos de red, que puede estar provocado por el mal funcionamiento o avería de algún dispositivo y que por lo tanto merece atención.

2.6.2.2. Detección de ataques a nivel de campo

Este enfoque se centra en los dispositivos de campo, obvia los paquetes, los flujos, servicios de red ejecutándose... el tráfico de red en general y su objetivo es conocer que esta ocurriendo en el proceso físico, si su funcionamiento no está siendo el adecuado o no está operando como debiera. En esta alternativa pasamos a medir magnitudes físicas tales como el ruido, vibraciones, temperaturas, humedad, etc.

Esta rama de investigación surge debido a que muchas veces, a nivel de red, el equipamiento existente en entornos OT, ya sean switches, routers, etc. no permite realizar técnicas de recolección de paquetes o flujos, como un *port mirroring*, y mucho menos obtener información de flujos como sFlow o NetFlow. La causa de este tipo de limitaciones se pueden observar en la Tabla 2.3 y más concretamente se debe a larga vida útil de la electrónica de red, provocando que los sistemas con mucha antigüedad no tengan este tipo de capacidades, y a su baja complejidad, limitando las capacidades a implementar.

Partimos de la base de que los datos ya están presentes en los dispositivos de campo, ya que los datos se monitorizan, el operador monitoriza cómo van las temperaturas, las presiones, niveles de ruido, etc. por lo que los datos ya existen y

no hay la necesidad de comprar electrónica de red nueva que nos permita hacer otras funciones.

La mayor parte de los trabajos relacionados con este rama de investigación se basan en la realización de modelos físicos, es decir, describir mediante código lo que hace el proceso físico en si mismo. Esta tarea es extremadamente costosa y compleja, además de requerir de expertos en el campo capaces e realizarla. Además, si se produce algún cambio en el proceso físico, el modelo queda obsoleto.

Otras aproximaciones relacionadas con este enfoque pero más abordables son las comentadas a continuación. Estas se basan en la detección de ataques guiadas por datos, no se construye ningún modelo físico ya que los datos nos pueden aportar la información que necesitamos. En el trabajo realizado por *Iturbe, Mikel et al.* [22] se presenta una solución para el diagnostico de ataques. Puesto que ya existían técnicas de detección de anomalías, en este trabajo se busca dar la respuesta al porqué de esas anomalías. Utilizando un modelo de análisis de componentes principales han concluido que los resultados obtenidos pueden utilizarse para distinguir, hasta cierto punto, entre perturbaciones e intrusiones. Otro ejemplo es el trabajo realizado por *Aoudi, Wissam et al.* [8] centrado en la detección de ataques más avanzados a nivel de campo. Esta propuesta la han bautizado como ***PASAD*** y esta basada en el Análisis de Espectro Singular. Como se puede observar en la Figura 2.7, a la izquierda tenemos una señal (de una vibración, temperatura, presión, etc.) y a la derecha el cluster. La parte azul se corresponde con el aprendizaje del sistema, la zona negra es el proceso en sí. Como podemos ver en el cluster, la parte negra queda dentro de la zona azul, lo que nos dice que es un comportamiento normal. Una vez comienza el ataque, la estructura cambia y los puntos rojos están contenidos fuera del cluster, indicándonos que ese comportamiento puede ser malicioso.

Aunque la detección de ataques a nivel de campo va más allá del objetivo de este trabajo, es interesante comentarla ya que nos pueden permitir detectar ataques realizados en el proceso físico y que no necesariamente tienen que poder verse a nivel de red. Un ejemplo de esto sería la manipulación manual de algún tipo de actuador, ya que esto no se realiza a través de una acción desde la HMI y por lo tanto, no

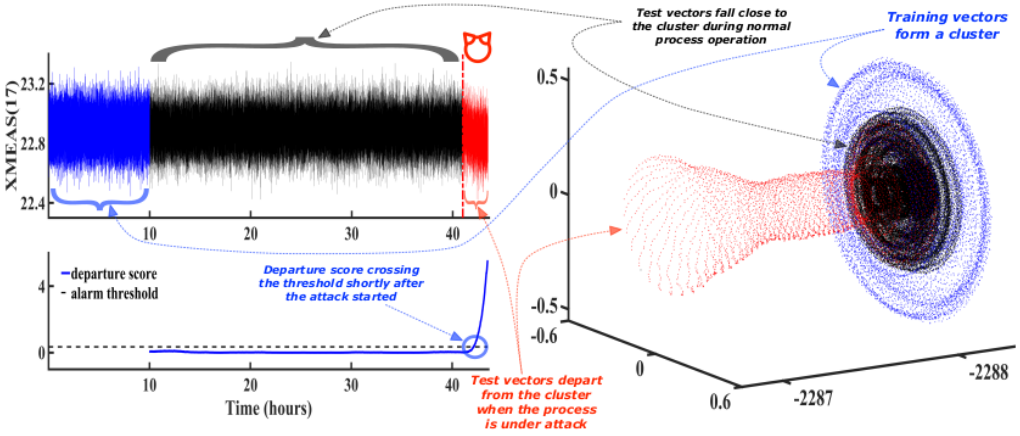


Figura 2.7: Representación de la señal y el cluster

quedaría registrada en el trafico de red, no pudiendo detectar el comportamiento malicioso.

Capítulo 3

Planificación

En este capítulo se procederá a explicar la planificación previa y estimaciones realizadas para el desarrollo de este proyecto. Además, se expondrán los recursos hardware y software empleados junto con los costes derivados de los mismos.

3.1. Planificación previa

Como paso previo, hemos definido la división de tareas en las que se compone el proyecto. Para llevar a cabo los objetivos explicados anteriormente, se ha dividido el proyecto en las siguientes fases:

1. Análisis de requisitos mediante entrevistas con los expertos en este campo.
2. Búsqueda extensa de documentación bibliográfica sobre el problema a tratar, desde enfoques médicos y técnicos, así como de las herramientas que se van a emplear.
3. Revisión del estado de la cuestión
4. Diseño de diferentes configuraciones de SOM y selección de diferentes conjuntos de características del tráfico
5. Definición de métricas de precisión.
6. Prueba de los algoritmos seleccionados en el cuarto punto.

7. Extracción de conclusiones y redacción de la memoria.

3.2. Modelo de desarrollo

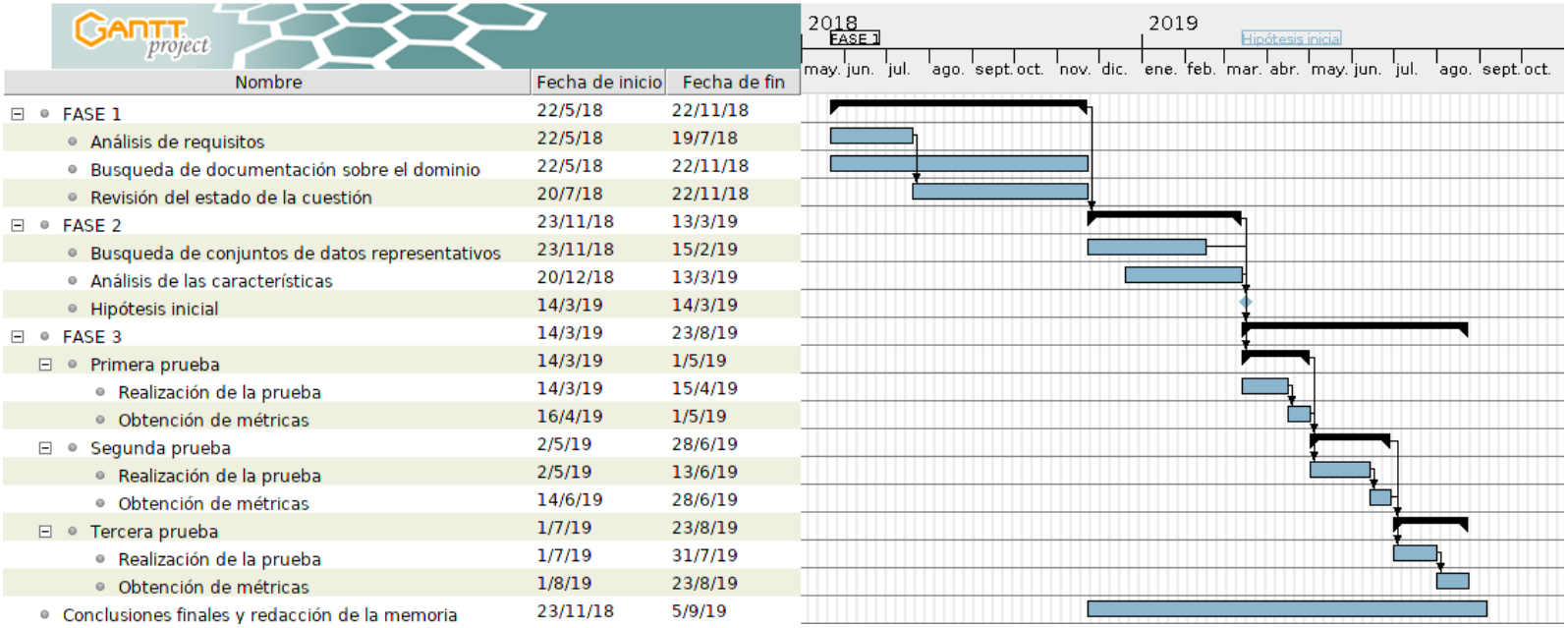
El método de desarrollo elegido es el método científico experimental. Podemos dividir este método en varias fases explicadas a continuación:

- **Fase 1:** Esta primera fase comprende las etapas de Observación y Planteamiento del problema del método experimental. En ella hemos englobado el análisis de la cuestión en técnicas de detección de anomalías basadas en SOM en redes de comunicaciones, especialmente en redes IoT y los requisitos que deben cumplir las técnicas a utilizar. Para ello hemos buscado documentación de referencia sobre el dominio a tratar.

- **Fase 2:** La segunda fase ha consistido en la selección de las propiedades de la SOM y las características del tráfico a usar y comparar. Para tal fin, hemos realizado una búsqueda de *datasets* de interés y hemos estudiado las características del trafico perteneciente a estos. A partir de ellas hemos realizado la hipótesis principal, es posible detectar anomalías en redes IoT a través de redes de neuronas artificiales de tipo SOM.

- **Fase 3:** La tercera etapa consistió en en el desarrollo y pruebas de los algoritmos de segmentación, así como el establecimiento de las métricas correspondientes, para finalmente extraer las conclusiones del trabajo realizado y redactar la memoria. Estos trabajos constituyen las fases de experimentación, registro y análisis e interpretación de los datos del método experimental, siendo esta última la que nos ha permitido confirmar la hipótesis inicial.

Así, con esta segmentación de las tareas por realizar, se ha sentado el inicio del proyecto a día 22 de Mayo de 2018 y con fin el día 6 de Septiembre de 2019. En la Figura 3.1 podemos ver en el diagrama con la distribución en tareas.



Nombre	Fecha de inicio	Fecha de fin
FASE 1	22/5/18	22/11/18
Análisis de requisitos	22/5/18	19/7/18
Busqueda de documentación sobre el dominio	22/5/18	22/11/18
Revisión del estado de la cuestión	20/7/18	22/11/18
FASE 2	23/11/18	13/3/19
Busqueda de conjuntos de datos representativos	23/11/18	15/2/19
Análisis de las características	20/12/18	13/3/19
Hipótesis inicial	14/3/19	14/3/19
FASE 3	14/3/19	23/8/19
Primera prueba	14/3/19	1/5/19
Realización de la prueba	14/3/19	15/4/19
Obtención de métricas	16/4/19	1/5/19
Segunda prueba	2/5/19	28/6/19
Realización de la prueba	2/5/19	13/6/19
Obtención de métricas	14/6/19	28/6/19
Tercera prueba	1/7/19	23/8/19
Realización de la prueba	1/7/19	31/7/19
Obtención de métricas	1/8/19	23/8/19
Conclusiones finales y redacción de la memoria	23/11/18	5/9/19

Figura 3.1: Diagrama de Gantt del proyecto

3.2.1. Recursos materiales

Durante el desarrollo de este proyecto han sido necesarios los recursos hardware y software listados continuación, además de todas las fuentes bibliográficas indicadas en la sección referente a la bibliografía:

- **Recursos hardware**:
  - **Ordenador personal**: Acer Aspire

    CPU: Intel® Core™ i5-3230M CPU @ 2.60GHz × 4

    SO: Ubuntu 18.04.2 LTS

    Memoria RAM: 8 Gb
  - **Clúster**: Servidores del laboratorio de investigación LIA2
- **Recursos software**:
  - **Anaconda**: Plataforma de desarrollo libre y abierta de los lenguajes Python y R, utilizada en ciencia de datos, y aprendizaje automático (machine learning). Está orientado a simplificar el despliegue y administración de los paquetes de software.
  - **Bash**: Bash es un programa informático, cuya función consiste en interpretar órdenes, y un lenguaje de consola. Es una shell de Unix compatible con POSIX y el intérprete de comandos por defecto en la mayoría de las distribuciones GNU/Linux, además de macOS.
  - **Python**: Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, dinámico y multiplataforma.
  - **Wireshark**: Wireshark es un analizador de protocolos utilizado para realizar análisis y solucionar problemas en redes de comunicaciones, para desarrollo de software y protocolos, y como una herramienta didáctica.

3.3. Estimación de costes

Tanto los **costes estimados del hardware** como los del **software** podemos considerarlos nulos, ya que el todo el material empleado estaba en posesión antes ya del comienzo del proyecto.

En cuanto a los costes humanos, Tabla 3.1 asumimos dos puestos principales: El de analista (yo, Miguel Ouviña) y el trabajo de supervisión de los dos directores de proyecto. Así, y teniendo en cuenta 5 días laborables de trabajo con una jornada de 4h, estimamos un total de 950 horas * hombre para el cargo de analista. Además, para la dirección y revisión del proyecto por parte de los directores estimamos un esfuerzo de 65 h * hombre repartidas entre los dos. Así, teniendo en cuenta el sueldo de un analista de 20 e/h y 30 e/h para los cargos de directores de proyecto, estimamos los costes presentes en la tabla 3.2.

	Sueldo	**Horas * Hombre**	**Inversión**
Director de proyecto (x2)	30 € / h	65	1.950 €
Analista	20 € / h	950	19.000 €
		Total:	**20.950 €**

Tabla 3.1: Estimación del coste de los recursos humanos

Capítulo 4

Trabajo realizado

En este capítulo se realizará un estudio del entorno y las características sobre las cuales se elaboró el conjunto de datos y, posteriormente, un análisis de los datos más representativos recolectados en las capturas del trafico de red.

4.1. Problemática

Debido a la naturaleza propietaria y a la sensibilidad potencial de las operaciones realizadas en las redes industriales, los datos reales de este tipo de redes rara vez son revelados a los investigadores. De hecho, no conocemos ningún conjunto de datos Modbus/TCP de acceso público que contenga un conjunto de paquetes extenso y debidamente etiquetados. Por lo tanto, la mayoría de los investigadores se basan en conjuntos de datos extraídos de los bancos de pruebas o simulaciones. Este problema es señalado por muchos autores en las investigaciones realizadas en este tipo de entornos [20] .

La falta de conjuntos de datos representativos y públicos dificulta la evaluación y la validación independiente de las nuevas soluciones de seguridad y frena el avance hacia soluciones eficaces y reutilizables.

4.2. Dataset

En el trabajo realizado por Antoine Lemay y José M. Fernandez [20] se presenta un dataset para la investigación sobre seguridad en redes SCADA, que es un subconjunto de la investigación sobre seguridad en redes SCI. El conjunto de datos incluye capturas de tráfico para una serie de parámetros de configuración y ataques diferentes que requieren de distintos grados de sofisticación para su detención.

Para la generación de datos de la red SCADA se implementó una pequeña red de laboratorio en una Sandbox SCADA. El protocolo de comunicaciones elegido en este trabajo fue Modbus, debido a la mejor disponibilidad de herramientas de código abierto que soportan el protocolo. Además, se eligió la variante Modbus que cierra las conexiones TCP después de cada petición en lugar de la variante con un largo tiempo de espera. Esta red está compuesta por un número variable de MTU y controladores (RTU). El número exacto de MTU y RTU es variable para proporcionar una visión del impacto de las diferentes implementaciones en el tráfico. Un ejemplo de configuración de red que incluye 2 MTU y 3 controladores se puede ver en la Figura 4.1.

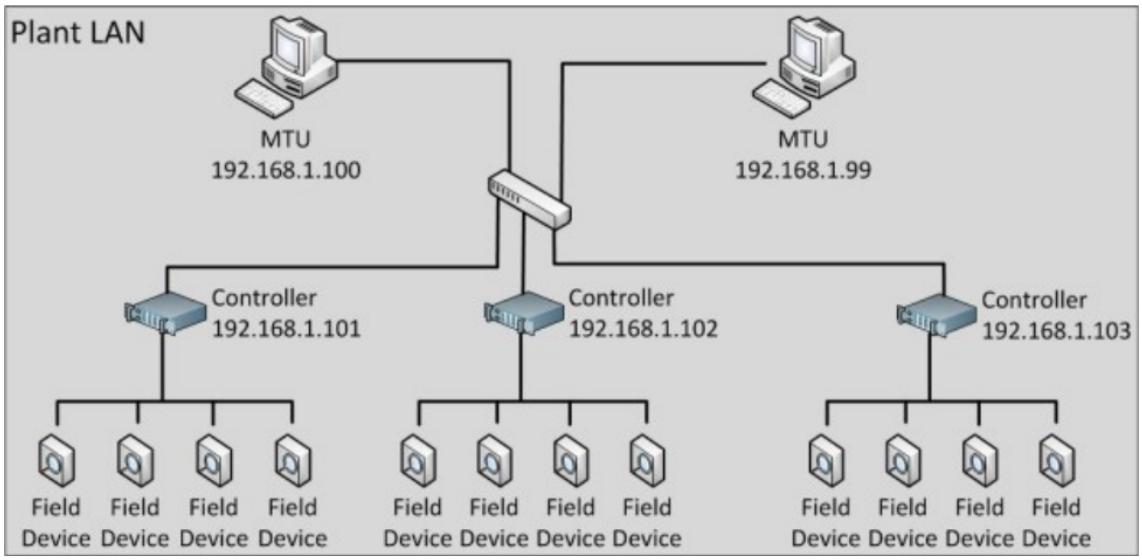


Figura 4.1: Topología de red de uno de los escenario

Cada controlador funciona con el mismo sistema físico. Como tal, cada sistema tiene el mismo número de puntos. Hay cuatro puntos que reportan el estado de los interruptores, implementados como *Binary Inputs*, cuatro puntos que representan la tensión en cada línea, implementados como *Holding Register* y cuatro puntos de control para accionar los interruptores, implementados como *Input Coils*. Estos puntos representan los dispositivos de campo conectados al controlador.

A la hora de etiquetar el tráfico en malicioso o no malicioso surge el quebradero de cabeza proveniente de cómo se define el término "malicioso". Consideramos un ataque que se conecta a un servicio FTP y envía un exploit. Esto es claramente un comportamiento malicioso y debe ser etiquetado como tal. Si embargo, cuando profundizamos en la captura de paquetes, pueden surgir preguntas. Por ejemplo, sería posible argumentar que el *handshake* TCP necesario para establecer una conexión con el servicio y enviar el exploit no es malicioso. Después de todo, no es diferente de los otros *handshake* realizados por clientes legítimos. En este caso, solo el paquete que contiene la vulnerabilidad real debe ser etiquetado como malicioso. Sin embargo, en ese caso, ¿cómo deben etiquetarse los ataques que abusan de la funcionalidad? Por ejemplo, en una disciplina de etiquetado en la que solo los paquetes que contienen código de ataque o explotación se etiquetan como maliciosos, ninguno de los paquetes relacionados con un análisis de puertos debe etiquetarse como malicioso, porque un escaneo de puertos solo realiza *handshake* legítimos.

Esta ambigüedad es particularmente importante con algunos de los tipos de ataques que se implementan en el conjunto de datos. Estos ataques abusan de la falta de autenticación en los protocolos SCADA para inyectar comandos o para realizar un *fingerprint* del contenido de un controlador usando paquetes correctamente formados.

Para abordar esta ambigüedad, los autores utilizaron la siguiente disciplina de etiquetado: si un paquete es parte de un flujo que incluye actividad maliciosa, se denomina malicioso. De lo contrario, es etiquetado como no malicioso.

Por ejemplo, el paquete de *polling* Modbus de la MTU se etiqueta como no malicioso, pero un paquete que contiene una vulnerabilidad se etiqueta como malicioso. Sin embargo, con la disciplina de etiquetado que han seguido los autores, el SYN que establece la conexión que contiene el paquete de exploit también es etiquetado como malicioso, incluso si no contiene ningun *payload*.

4.2.1. Análisis

Este conjunto de datos consiste en un total de once ficheros de datos en formato pcap, con sus respectivos archivos de etiquetado de paquetes, representando el número cero a paquetes de tráfico normal y el número uno a paquetes considerados parte de un ataque. La topología de red usada en cada captura es diferente, así como los ataques introducidos. En la Tabla 4.1 se pude ver una descripción del conjunto de datos que forma la red SCADA.

Se han tenido en cuenta algunas características para representar la información sobre el tráfico de la red. Realizamos el análisis sobre tráfico TCP y UDP (hay otros protocolos en las capturas pero son minoritarios, menos del 0,05 %), clasificado en ataque y normal. Bajo estas propiedades, el número de paquetes analizados es de 890.760. El tráfico considerado como anómalo se compone de 1.405 paquetes, lo que representa el 0,158 % de los datos utilizados para el análisis. Los detalles del número de paquetes observados por fichero, protocolo y etiqueta (legítimo/malicioso) se describen en la Tabla 4.2.

Nombre	Descripción	Actividad Maliciosa?	Número de Paquetes
Run 8	1 hora de tráfico Modbus regular incluyendo sondeo y operaciones manuales - 2 MTU, 3 RTU e intervalo de sondeo de 10 segundos.	No	72.186
Run 11	1 hora de tráfico Modbus regular incluyendo sondeo y operaciones manuales - 2 MTU, 3 RTU e intervalo de sondeo de 10 segundos.	No	72.498
Run1 6RTU	1 hora de tráfico Modbus regular incluyendo sondeo y operaciones manuales - 2 MTU, 6 RTU e intervalo de sondeo de 10 segundos.	No	134.690
Run1 12RTU	1 hora de tráfico Modbus regular incluyendo sondeo y operaciones manuales - 2 MTU, 12 RTU e intervalo de sondeo de 10 segundos.	No	238.360
Run1 3RTU 2s	1 hora de tráfico Modbus regular incluyendo sondeo y operaciones manuales - 2 MTU, 3 RTU e intervalo de sondeo de 2 segundos.	No	305.932
Modbus polling only 6RTU	1 hora de tráfico Modbus regular incluyendo solo sondeo - 1 MTU, 3 RTU y 10 segundos de intervalo de sondeo.	No	58.325

Table 4.1 continued from previous page

Nombre	Descripción	Actividad Maliciosa?	Número de Paquetes
Moving two files Modbus 6RTU	3 minutos de tráfico Modbus regular incluyendo solo sondeo - 1 MTU, 6 RTU y 10 segundos de intervalo de sondeo.	Sí	3.319
Send a fake command Modbus 6RTU with operate	11 minutos de tráfico Modbus regular incluyendo sondeo y operaciones manuales - 1 MTU, 6 RTU intervalo de sondeo de 10 segundos. También incluye el envío de una operación de escritura Modbus desde un RTU comprometido utilizando la funcionalidad de proxy de Metasploit y la herramienta proxychains.	Sí	11.166
CnC uploading exe modbus 6RTU with operate	1 minuto de tráfico Modbus regular incluyendo sondeo y operaciones manuales - 1 MTU, 6 RTU e intervalo de sondeo de 10 segundos. También incluye el envío de un archivo EXE desde un RTU comprometido a otro RTU comprometido a través de un canal de meterpreter de Metasploit.	Sí	1.426

Table 4.1 continued from previous page

Nombre	Descripción	Actividad Maliciosa?	Número de Paquetes
6RTU with operate	5 minutos de tráfico Modbus regular incluyendo polling y operación manual - 1 MTU, 6 RTU y 10 segundos de intervalo de sondeo. También incluye el uso de un exploit (ms08_netapi) de un RTU comprometido para comprometer otro RTU usando Metasploit.	Sí	1.856

Tabla 4.1: Descripción del conjunto de datos de la red SCADA

Capturas	TCP		UDP		Paquetes
	Normal	Ataque	Normal	Ataque	
Run8	71.750	0	200	0	71.950
Run11	72.024	0	186	0	72.210
Run1 6RTU	133.853	0	485	0	134.338
Run1 12RTU	234.001	0	201	0	234.202
Run1 3RTU 2s	302.048	0	140	0	302.188
Modbus polling only 6RTU	58.044	0	103	0	58.147
Moving two files Modbus 6RTU	3.240	72	0	3	3.315
Send a fake command Modbus 6RTU with operate	11.121	10	5	0	11.136
CnC uploading exe modbus 6RTU with operate	1.305	119	0	2	1.426
6RTU with operate	649	1199	0	0	1.848
Total	888.035	1.400	1.320	5	890.760

Tabla 4.2: Estadísticas generales del tráfico

Para el análisis tomamos algunas características relevantes de tráfico de red como: Direcciones IP origen y destino, puertos origen y destino, bytes totales por paquetes y bytes medios por paquetes.

4.2.1.1. Direcciones IP origen y destino

Las direcciones IP origen y destino pueden ser unas de las características más relevantes, ya que permiten identificar los dispositivos que realizan una comunicación. La Figura 4.2 muestra cómo se distribuyen estas direcciones teniendo en cuenta que se dividen en broadcast, multicast, desconocidas (0.0.0.0/8), direcciones IP correspondientes a las MTU y RTU, y WAN (que incluye todas las direcciones públicas). Para una mayor claridad en la visualización de las características, se ha realizado una normalización por etiquetas. De esta forma, la suma de las cuatro barras del tráfico de ataque es el 100 %, y lo mismos para las barras del tráfico normal.

La Figura 4.2 muestra la distribución de los paquetes normales y de ataques entre las diferentes direcciones IP origen. Cerca del 95 % del tráfico anómalo proviene de las RTU y el 5 % restante de las MTU.

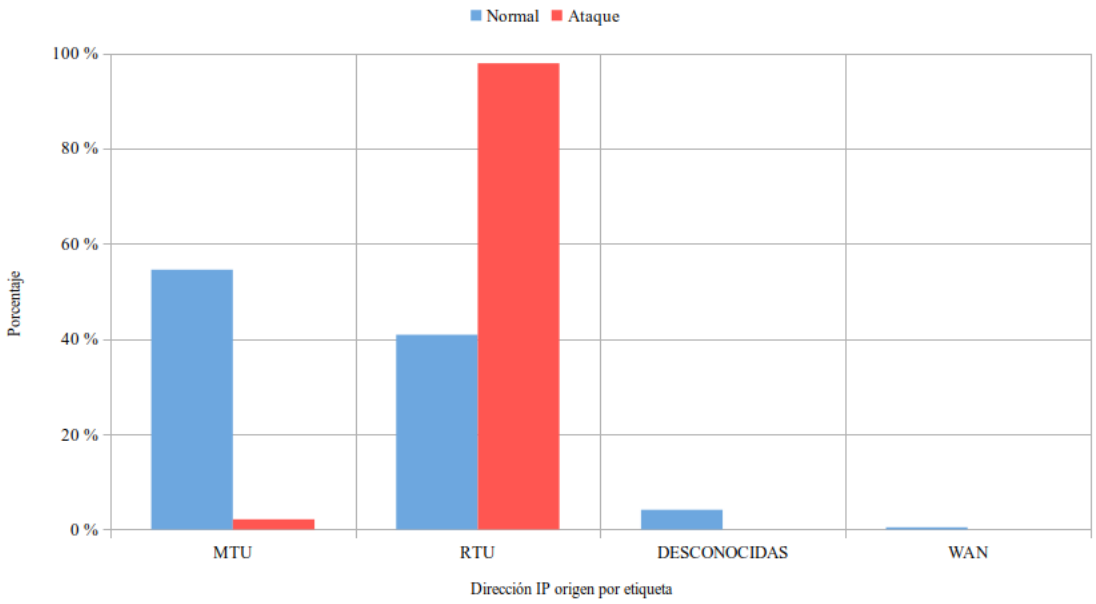


Figura 4.2: Estadísticas Dirección IP origen

La Figura 4.3 muestra las distribución de los paquetes normales y de ataques entre las diferentes direcciones IP destino. La mayor parte del tráfico de ataque, cerca del 97 %, está destinado a las RTU. Los paquetes de ataques con direcciones de broadcast, multicast, desconocidas y de MTU representan la minoría del tráfico anómalo, alrededor del 3 %.

Por lo tanto, determinar las direcciones de origen y destino puede ser importante para predecir el tipo de tráfico. Con el análisis mostrado anteriormente sobre las características IP origen y destino, podemos observar como la mayor parte del tráfico considerado como ataque se produce en comunicaciones entre RTU.

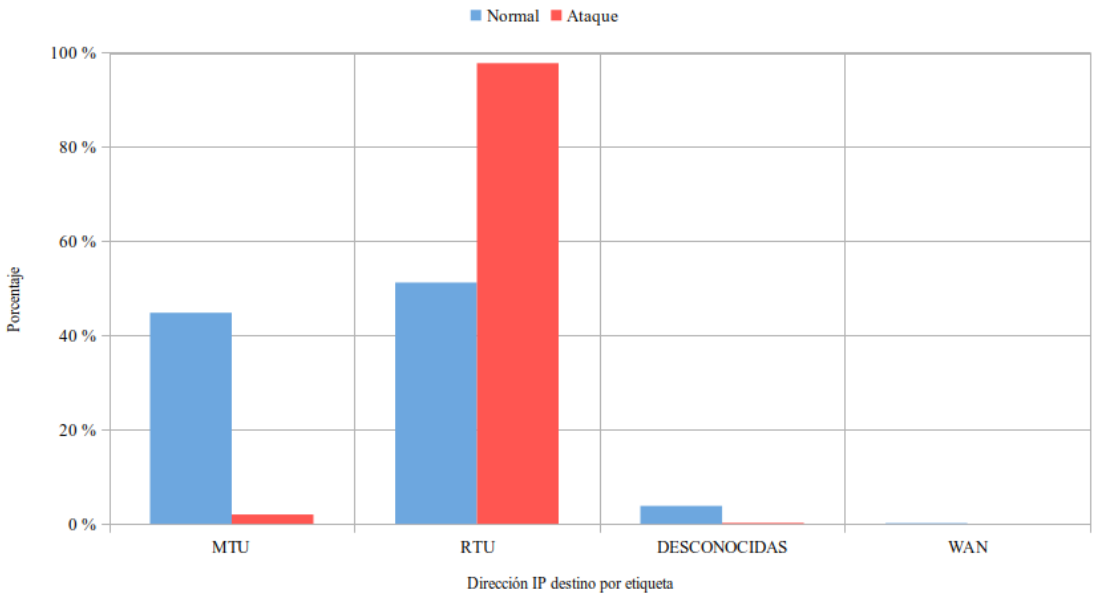


Figura 4.3: Estadísticas Dirección IP destino

4.2.1.2. Puertos origen y destino

Algunos puertos son comúnmente utilizados por algún tipo de servicio específico, mientras que otros son de uso público. Hemos realizado el análisis de los datos haciendo la clasificación de los puertos por puertos conocidos (números de puerto inferiores a 1024), puertos registrados (números de puerto entre 1024 y 49151) y puertos privados (números de puerto superiores a 49152).

La Figura 4.4 muestra la distribución de los puertos origen, en los cuales cerca del 55 % de los puertos origen utilizados son registrados y alrededor del 40 % son conocidos, para paquetes clasificados como normales. Los menos utilizados son los puertos privados. Para los paquetes clasificados como ataques el tema cambia. La mayor parte de los paquetes, próximos al 95 %, son puertos clasificados como registrados y el 5 % restante, como conocidos.

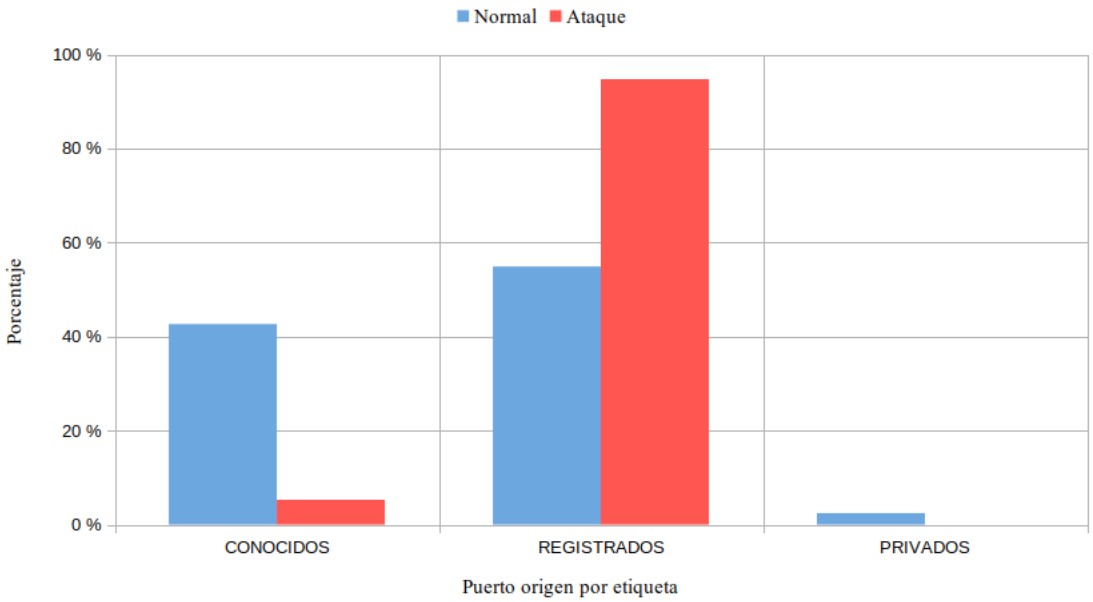


Figura 4.4: Estadísticas puerto origen

Por otro lado, la Figura 4.5 muestra la distribución de los puertos destino. A diferencia de los puertos origen, hay una permuta entre el número de puertos registrados y conocidos para el tráfico normal. El tráfico etiquetado como anómalo se mantiene de forma similar.

4.2.1.3. Bytes promedio

El tipo de protocolo utilizado y el tamaño del paquete puede ayudarnos a distinguir el tipo de tráfico con el que estamos tratando. La Tabla 4.3 presenta los valores mínimos y máximos, la mediana y la media de los bytes promedio por paquete, por protocolo y global. Este análisis nos permite entender mejor la distribución de los datos.

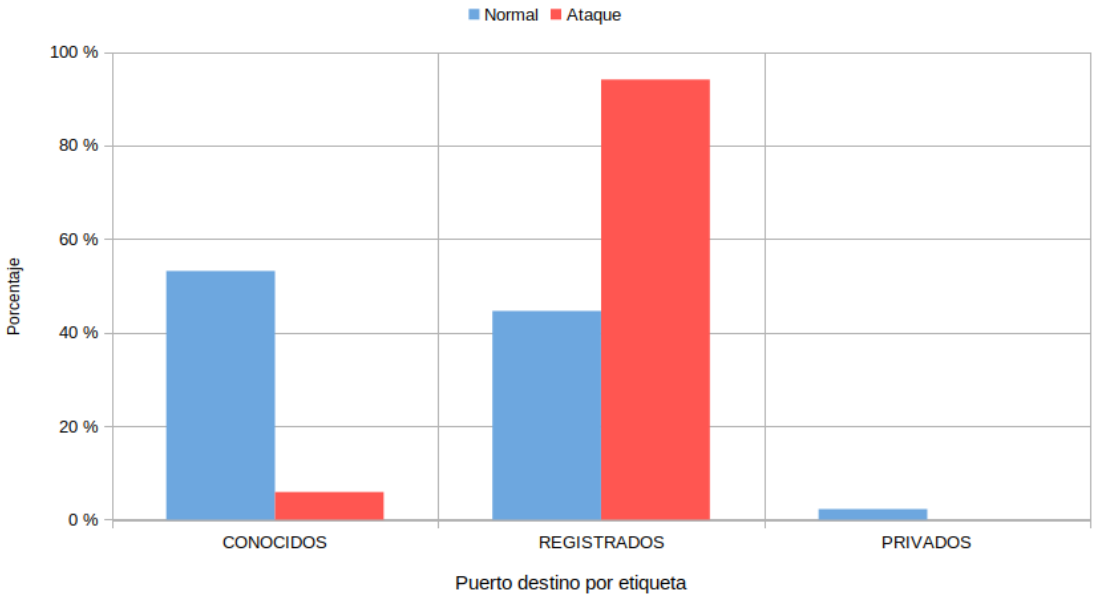


Figura 4.5: Estadísticas puerto destino

Con respecto al protocolo TCP, encontramos que más de la mitad del tráfico normal tiene un tamaño de paquete de 40 bytes. Por otro lado, para el tráfico de ataque encontramos que más del 50 % de los paquetes tienen un tamaño de 1500 bytes. Si nos centramos en la media para los dos tipos de tráfico (normal/ataque), observamos que los resultados obtenidos tienen una diferencia bastante marcada. Sin embargo, para el protocolo UDP, nos encontramos con características bastante similares entre los paquetes de tráfico normal y de ataque. Esto puede deberse a que el número de paquetes UDP es muy bajo, representando un 0,15 % del total. Además, el protocolo que estamos estudiando es TCP Modbus, que se basa en el envío de datos a través del servicio orientado a la conexión proporcionado por TCP. Debido a esto, no podemos sacar conclusiones importantes para este protocolo.

	TCP		UDP		Global
	Normal	Ataque	Normal	Ataque	
Mínimo	40	40	50	78	40
Mediana	40	1.500	78	90	40
Máximo	1.500	1.500	328	229	1.500
Media	58,78	832,78	100,57	113	60,21

Tabla 4.3: Estadísticas del número de bytes promedio

4.2.2. Limitaciones del dataset

A continuación presentamos las limitaciones percibidas tanto por los investigadores que han realizado el proyecto como las que nosotros mismos hemos observado a medida avanzábamos en análisis del conjunto de datos.

1. Una primera limitación es la configuración elegida por los autores. Con ella no podemos observar el tráfico entre el controlador y los dispositivos de campo.

2. El nivel de similitud entre los controladores es representativo de algunas redes SCI reales en la industria, donde los diseños se copian para acelerar la instalación. Sin embargo, muchas redes de SCI incluyen una mayor diversidad de dispositivos de campo y valores detectados por estos. Esto representa una limitación de los conjuntos de datos generados y debe tenerse en cuenta al utilizar el conjunto de datos para la experimentación.

3. Un nivel similar de reutilización se empleó para la configuración de los sistemas MTU. En particular, se utilizó una configuración de *polling* uniforme en todas las MTU y en todos los controladores. Al igual que con el número de nodos, se ha variado la periodicidad del *polling* entre conjuntos de datos para ilustrar el impacto de la modificación del intervalo de *polling* en el tráfico. Sin embargo, el intervalo de *polling* permanece constante para todos los controladores y todas las MTU en una captura de paquetes determinada. Esta configuración de *polling* uniforme se observa más comúnmente que la configuración de controlador de alta similitud descrita anteriormente. Esto se debe a que el intervalo de *polling* depende de los requisitos de actualización de datos de las MTU. Los autores reconocen que algunos operadores SCADA pueden tener configuraciones de intervalos de *polling* mixtos. Esto representa una limitación de los conjuntos de datos generados que debería ser tomada en cuenta.

4. Otro parámetro es el número de operaciones realizadas por los operadores a través de la MTU. En algunas de las capturas, se simula la presencia de operadores humanos en el sistema enviando paquetes de control a intervalos aleatorios. Estos paquetes de control consisten en paquetes Write_Single_Coil (Modbus código de función 5) que instruyen a un controlador seleccionado

aleatoriamente a establecer el valor de un interruptor seleccionado aleatoriamente a un valor aleatorio (ON u OFF). Esto se hace tanto para variar el tipo de tráfico en el sistema como para introducir entropía en las mediciones, ya que alterar el estado de los interruptores alterará los resultados de las mediciones. Obviamente, este método de generación de tráfico no capta completamente el comportamiento humano. Esta limitación también debe tenerse en cuenta al analizar los resultados obtenidos del conjunto de datos.

5. Durante el análisis anterior, expuesto en la Sección 4.2.1, hemos detectado un problema relacionado con el fichero de tráfico llamado *Characterization Modbus 6RTU with operate* y que explicamos a continuación. Como se puede observar en la Tabla 4.4, en el documento elaborado por los autores, se dice que este fichero posee actividad maliciosa. En nuestro análisis, Tabla 4.5, apreciamos que hay un número de paquetes TCP y UDP de 12.275 y 1, respectivamente, pero no tenemos constancia de paquetes maliciosos. Podemos ver que el número de paquetes es diferente en los dos análisis. Así, en el análisis realizado por los autores, el número de paquetes es de 12.296, frente a 12.276 paquetes que aparecen en nuestro análisis, por lo que podemos estar cometiendo algún fallo o simplemente, esos 20 paquetes de desajuste, no pertenecen al protocolo TCP ni UDP. Para asegurarnos de que no comentemos ningún error, analizamos el fichero de etiquetado relacionado con esta captura de tráfico, en busca de paquetes considerados como anómalos. La sintaxis del fichero es *número de paquete*;*etiquetado*, por lo que deberemos buscar cadenas de caracteres que cumplan el patrón ";1", para ello usamos el siguiente comando en Linux, el cual devuelve el número de coincidencias del patrón que le indiquemos con la opción *-w* sobre el fichero deseado con la opción *-c*:

```
$ grep -w ";1" -c
   characterization_modbus_6RTU_with_operate_labeled.csv
$ 0
```

Figura 4.6: Bash script

El resultado devuelto es *0*, indicándonos que no hay ninguna coincidencia con el patrón. Con esto llegamos a la conclusión de que no hay ningún paquete

etiquetado como anómalo. Sin embargo, como en la descripción sí se refleja que se ha introducido este tipo de tráfico, decidimos descartar el uso de este paquete para las pruebas posteriores por las inconsistencias mostradas.

Nombre	**Descripción**	**Actividad maliciosa?**	**Número de paquetes**
Characterization Modbus 6RTU with operate	5,5 minutos de tráfico Modbus regular incluyendo sondeo y operaciones manuales - 1 MTU, 6 RTU e intervalo de sondeo de 10 segundos. También incluye el envío de una serie de comandos de lectura Modbus para describir los registros disponibles desde un RTU comprometido.	Sí	12.296

Tabla 4.4: Descripción del dataset Characterization Modbus 6RTU with operate

Capturas	**TCP**		**UDP**		**Paquetes**
	Normal	**Ataque**	**Normal**	**Ataque**	
Characterization Modbus 6RTU with operate	12.275	0	1	0	12.276

Tabla 4.5: Estadísticas de tráfico para el dataset Characterization Modbus 6RTU with operate

4.3. Preprocesado de datos

El preprocesamiento de datos es una etapa esencial del proceso de descubrimiento de información. En esta etapa realizamos la limpieza de datos, su integración, transformación y reducción para la siguiente fase, la fase de procesamiento.

En el presente apartado se describirán los pasos realizados para, partiendo de ficheros con capturas de tráfico de red en formato *pcap*, obtener los archivos con el formato exigido por el sistema para la etapa posterior, el procesamiento.

4.3.1. Introducción

Debido a los dos motivos expuestos a continuación, en el momento de la planificación de este proyecto, se decidió realizar la fase de análisis y preprocesado de datos de forma simultanea en el tiempo:

- El formato inicial de los datos requiere de una transformación preparatoria para integrar estos en la solución elegida para su análisis y que expondremos a continuación, ya que el formato *pcap* es binario y no podemos tratar los datos con facilidad.
- Debido a que los datos han sido creados de forma artificial en un entorno de laboratorio, cabe la posibilidad de que estos pueden contener errores, y es algo que queremos detectar lo antes posible para evitar introducir datos no fiables en los análisis y procesamiento.

Para analizar datos de capturas de tráfico de red, a día de hoy existen multitud de herramientas, ya sean de pago o software libre, de las que destacamos NetworkMiner, Packet Analyzer o Wireshark y su forma en línea de comandos Tshark, entre otros.

Las características de este proyecto (capturas de tráfico de red, dimensión de los datos utilizados, protocolos poco conocidos, etc.) requieren de una herramienta o conjunto de ellas que nos permita realizar un análisis especializado, automatizando extracciones de datos y manejar toda la tarea de preprocesado de datos y análisis de una forma mucho más ágil. Para tal fin, hemos optado por utilizar la herramienta Wireshark, un programa de software libre y posiblemente el más utilizados a la hora de realizar capturas de tráfico de red y analizar su contenido. Además hemos empleado scripts hechos tanto en Python como en Bash, ayudándonos de estructuras propias del lenguaje y librerías de tratamiento de datos, como pueden ser DataFrames, Series, etc.

Wireshark posee multitud de funcionalidades para hacer análisis de capturas de tráfico de red, como la utilización de filtros, que permiten cribar el tráfico por características como IP origen o destino, puertos, tamaño de paquetes, protocolos, etc. o el uso de la característica *Follow TCP Stream*, en la que se puede seguir todo

el tráfico perteneciente a un flujo TCP concreto, entre otras. Además, permite exportar los datos de un formato binario como es el *pcap* a uno de texto sencillo como JSON, formato que será de gran utilidad para la integración con Python.

4.3.2. Integración de datos

Después de realizar un análisis particular de cada captura de tráfico de forma individual, Tabla 4.2, necesitamos integrar todos los datos de los diferentes ficheros (fuentes de información) en una misma estructura de datos para crear información homogénea. Este paso permite tener una visión global del conjunto de datos y evitar futuros problemas de representación y codificación a la hora de normalizar los datos o codificarlos.

Para tal fin, hemos utilizado Python y una estructura de datos llamada DataFrame. El DataFrame es una estructura de datos tabular, bidimensional y potencialmente heterogénea, que se compone de ejes (columnas y filas) ordenados y etiquetados. Se pueden realizar una gran cantidad de operaciones alineadas a ambos ejes lo que nos permite manipular y visualizar las características del conjunto de datos con suma facilidad. La estructura de datos primaria del DataFrame son las Series, que a su vez se basa en los Pandas.

Al finalizar el proceso de integración de datos, obtenemos la estructura deseada que se compone de la siguiente forma:

- En las **filas**, se representan los paquetes obtenidos de la diferentes capturas de tráfico de red.
- En las **columnas**, se visualizan los atributos o características relacionadas con cada paquete en concreto.

En el momento de realizar la integración de todos los datos observamos algo que ya habíamos podido intuir: la necesidad de una potencia de cómputo y un almacenamiento mayor de la que puede aportar un ordenador personal, ya que la estructura de datos en su totalidad alcanza cerca de los 30 GB de espacio en memoria.

Debido a estas circunstancias, todo el código necesario fue desarrollado de manera local en el ordenador personal del alumno pero, tanto la ejecución de este como las acciones posteriores de extracción de estadísticas y realización de pruebas, se llevaron a cabo en el clúster LIA2.

4.3.3. Transformación y limpieza de datos

Una vez unificados todos los datos en una misma estructura, debemos seleccionar aquellos atributos o características de interés y que creamos más relevantes para nuestro estudio. En este caso particular, estamos tratando con capturas de tráfico de red y debemos adaptarnos a ello.

Las **características** más representativas que definen una comunicación de red las podemos encontrar en la definición de flujo de red. Un flujo de red es una serie de comunicaciones entre dos puntos finales que están limitadas por la apertura y el cierre de sesiones. Cada paquete perteneciente a un flujo de red se produce en un momento de tiempo determinado (*timestamp*) y entre dos hosts (IP origen y destino). A su vez, el paquete tendrá un tamaño, un protocolo utilizado en la capa de transporte (*generalmente TCP o UDP*) y unos puertos lógicos que se corresponden con la aplicación utilizada. Además, hemos optado por incluir alguna característica del campo de aplicación, Modbus en este caso.

Otro tema a tratar en este apartado es la clasificación de los atributos seleccionados. El sistema en el cual clasificaremos los datos exige que hagamos una clasificación de estos entre numéricos y categóricos. Podemos definir los datos estadísticos es estas dos categorías:

- **Categóricos:** Los datos categóricos son datos que provienen de resultados de experimentos en los que sus resultado se miden en escalas categóricas. Medir en una escala categórica consiste en observar el resultado de un experimento y asignarle una clase o categoría, de entre un número finito de clases posibles. Los datos categóricos también pueden tomar valores numéricos (como "1"que indica hombre y "2"que indica mujer), pero esos números no tienen significado matemático. No se pueden sumar, por ejemplo.

- **Numéricos:** Los datos numéricos, por su parte, son aquellos cuyo resultado es un número y son obtenidos a través de muestreos o mediciones. Estos datos tienen un significado como medida, como la estatura, el peso, el coeficiente intelectual o la presión arterial de una persona; o son un recuento, como el número de acciones que posee una persona o cuántos dientes tiene un perro.

4.3.4. Imputación de valores perdidos

La presencia de información faltante (o ausencia de información) es un problema constante con el que se debe lidiar. La misma puede originarse por un registro defectuoso de la información o, directamente, por la ausencia natural de la información.

En este trabajo hemos optado por seguir dos aproximaciones para resolver este problema. El enfoque se centra en el tipo de dato al que nos enfrentamos, por lo que según sean numéricos o categóricos, afrontamos el problema de la siguiente manera:

- Para atributos clasificados como **numéricos**, hemos calculado la **media** de entre todos lo valores pertenecientes al atributo en cuestión y substituir la información faltante por este valor.
- Ante los atributos **categóricos** la forma de operar es distinta. Simplemente, si nos encontramos con ausencia de información lo que haremos es reemplazar dicho valor por la cadena ***No_Aplica***.

Se podía haber optado por otro enfoque y eliminar los datos relacionados con la ausencia de información, pero debido al tipo de datos que se están tratando, evitamos eliminar toda la información que no este completa ya que en este campo en particular, muchos paquetes pertenecientes a ciertos protocolos de red no contienen la misma información que muchos otros y tendríamos una perdida de información considerable.

4.3.5. Normalización

El objetivo de la normalizacion es cambiar los valores de los atributos numéricos a una escala común, sin distorsionar las diferencias en los rangos de valores. No siempre se requiere de esta fase, solo cuando las características del conjunto de datos tienen rangos diferentes, como es el caso del conjunto de datos utilizado en el proyecto. De este modo, tenemos la certeza de que cada atributo tiene el mismo peso dentro del análisis y no que un atributo por tener mayor valor, influirá más en el resultado, ya que no significa necesariamente que sea más importante por que dicho valor sea mayor.

Entre todas las aproximaciones existentes nos hemos decantado por la normalización ***MinMax***, una estrategia que transforma linealmente de x a y, donde:

$$y = \frac{x - min}{max - min}$$

Figura 4.7: Normalización

Donde min y max son los valores mínimos y máximos en X, donde X es el conjunto de valores observados de x. Esto significa que el valor mínimo en X se asigna a 0 y el valor máximo en X se fija a 1. Por lo tanto, todo el rango de valores de X de mín. a máx. se asigna al rango de 0 a 1.

Por otro lado, también hemos aplicado una normalización a los atributos categóricos. Debido a que estamos trabajando con datos que pueden no ser muy fáciles de interpretar, por ejemplo las direcciones IP, hemos normalizando los datos de la siguiente forma: codificamos los valores correspondientes a cada atributo de la forma 0..N, siendo N el numero máximo de elementos distintos para ese atributo. De esta manera obtenemos valores más fáciles de formatear para una máquina. Este proceso también nos evita trabajar con los elementos No_Aplica, comentados en el apartado anterior, dándole un valor numérico.

4.3.6. Archivos y formato exigido

La última fase del preprocesado de los datos consiste en adaptar el formato con el que hemos trabajado a la entrada necesaria para la realización de las pruebas o procesamiento. Los ficheros y el formato exigido por el sistema para procesar los datos es el definido a continuación:

- **Identifiers**: Este fichero debe contener un identificador único por linea del fichero y un número de identificadores igual al de paquetes utilizados.
- **Numericals**: Se compondrá de una linea de cabecera precedida por una # y los identificadores de las características de los atributos numéricos elegidas y separadas por un espacio. Cada linea del archivo se compondrá de las mediciones para el paquete en cuestión, separadas también por un espacio.
- **Categoricals**: Misma estructura que el fichero ***Numericals*** pero con los datos de las características seleccionadas como categóricas.
- **Target**: Este fichero almacena en cada linea la clasificación para dicho paquete entre ataque(1) y tráfico legítimo(0).

Capítulo 5

Resultados y discusión

En este capítulo serán presentados y discutidos los resultados obtenidos y las pruebas realizadas. En primer lugar explicaremos las métricas utilizadas para medir el rendimiento de los resultados obtenidos. En segundo lugar presentaremos las características del tráfico utilizadas y los resultados obtenidos apoyándonos en las métricas y, posteriormente, presentaremos un análisis de los resultados.

5.1. Información sobre las pruebas

Toda la parte de entrenamiento, y posterior fase de detección, se realizó en base al conjunto de datos propuesto por Lemay y Fernández [20]. El dataset presentado contiene archivos que no contienen ataques, por lo que estos fueron utilizados en la parte de entrenamiento, y posteriormente, los archivos que presentan ataques se usaron para la fase de detección del sistema.

Después de ejecutar la fase de detección, el sistema genera una serie de ficheros que deberemos analizar y así determinar los resultados obtenidos. El número de ficheros generados varia en función del tamaño del mapa utilizado. Por ejemplo, si el tamaño del mapa es de 10x10 el número máximo de archivos que obtendremos será 100 y, cada uno de ellos representan las observaciones hechas por cada neurona del mapa. Es posible que alguna neurona no tenga ninguna observación, representándose esta situación con la ausencia del fichero correspondiente a esa neurona.

Cada fichero contiene los identificadores de las observaciones que pertenecen a esa neurona, por lo que, juntando el contenido de estos ficheros (teniendo en cuenta a cual pertenecen) con las etiquetas de cada observación tendríamos la relación: ID_Observación - ID_Neurona - Etiqueta. Haciendo uso de los datos anteriores, podemos determinar cual es la etiqueta mayoritaria de una neurona, creando así la etiqueta predicha para todos los que formen parte de la misma. Todo este proceso se automatiza mediante el uso de scripts elaborados en Python para facilitar tanto la visualización como la recolección de métricas que expondremos a lo largo del presente capítulo.

Posteriormente, nos apoyaremos en la Matriz de Confusión para la comparación de resultados y obtener las conclusiones finales, ya que con ella se consigue expresar de manera muy resumida y visual la importancia tanto de los aciertos en la clasificación del tráfico como los errores.

5.2. Matriz de confusión

La matriz de confusión de un problema de **clase n** es una **matriz nxn** en la que las filas se nombran según las clases reales y las columnas, según las clases previstas por el modelo. Sirve para mostrar de forma explícita cuándo una clase es confundida con otra. Por eso, permite trabajar de forma separada con distintos tipos de error [10].

En el caso que nos atañe, un modelo binario que busque predecir si los paquetes pertenecientes al tráfico de red son considerados **legítimos** o, por otro lado, **anómalos**. Basándonos en determinadas características de este consideraremos las clases reales **P**(ositivo = tráfico anómalo) y **N**(egativo = tráfico real), y las clases pronosticadas por el modelo, **S**(í, es anómalo), o **N**(o, es real). De esta forma, la matriz de confusión para este modelo tiene etiquetadas sus filas con las clases reales, y sus columnas, con las predichas por el modelo. En la Tabla 5.1 podemos observar lo anteriormente explicado:

	N(modelo)	S(modelo)
n(real)	Negativos Reales	Falsos Positivos
p(real)	Falsos Negativos	Positivos Reales

Tabla 5.1: Matriz de confusión para clasificador binario.

De esta forma, la diagonal principal contiene la suma de todas las predicciones correctas (el modelo dice "S" y acierta, son paquetes considerados anómalos , o dice "N" y acierta también, son paquetes legítimos). La otra diagonal refleja los errores del clasificador: los falsos positivos o "true positives" (dice que es anómalo "S", pero en realidad no lo es "n"), o los falsos negativos o "false negatives" (dice que es legítimo "N", pero en realidad es anómalo "p").

Sin embargo, cuando las distintas "clases" están muy desequilibradas, esta forma de clasificar la "bondad" del funcionamiento de un clasificador resulta poco útil. El problema radica en que al medir la precisión del algoritmo de esa forma no distinguimos entre los errores de tipo falso positivo y falso negativo, **como si ambos tuvieran la misma importancia**. Y esto no es así. Si nos ceñimos a nuestro caso podemos verlo con el siguiente ejemplo:

- **Falso positivo o "Error tipo I":** El paquete es legítimo pero el algoritmo ha determinado que es anómalo. Se realizaran pruebas adicionales que acabarán descartando que el paquete es malicioso. Se producirá un coste en tiempo de investigación, pero no se traducirá en un riego o amenaza.
- **Falso negativo o "Error tipo II":** El paquete es malicioso pero el algoritmo predice que no. Este error del modelo se traduce en una falta de detección del tráfico sospechoso. El comportamiento relacionado con este paquete no se tratará y esto, indudablemente, puede suponer un riesgo.

Viendo la importancia que tiene discriminar en cada caso concreto los distintos tipos de error que pueden resultar de la aplicación de algoritmo, entendemos mejor la necesidad de trabajar con diferentes métricas. Así, vamos a expresar la matriz de confusión de esta otra forma, como podemos ver en la Tabla 5.2:

Matriz de Confisión		Predicho			
		Negativo	Positivo		
Real	Negativo	a	b	Verdadero Negativo (True negative rate)	a/(a+b)
	Positivo	c	d	Exactitud	d/(b+d)
		Sensibilidad d/(d+c)	Especifidad a/(a/b)	Precision = (a+d)/(a+b+c+d)	

Tabla 5.2: Matriz de confusión con otras métricas de evaluación

El significado de cada uno de los términos es el siguiente:

- ***a*** es el número de predicciones correctas de clase negativa (negativos reales)
- ***b*** es el número de predicciones incorrectas de clase positiva (falsos positivos)
- ***c*** es el número de predicciones incorrectas de clase negativa (falsos negativos)
- ***d*** es el número de predicciones correctas de clase positiva (positivos reales)

Para entender estos valores vamos a retomar nuestro caso. Así, "a" sería el número de veces que el algoritmo ha acertado al decir el paquete es legitimo; "b" sería el número de veces que el algoritmo se confunde y dice que es malicioso un paquete legitimo; "c" serían los casos en que el algoritmo predice que la paquete es legitimo, y resulta que no lo es y, por último, "d" serían los aciertos del algoritmo al predecir que el paquete es anómalo.

Basándonos en los valores de ésta nueva matriz de confusión más completa, vamos a definir una serie de métricas que nos serán muy útiles.

5.2.1. Precisión y exactitud.

La Precisión o "Accuracy" (AC) se refiere a la dispersión del conjunto de valores obtenidos a partir de mediciones repetidas de una magnitud. Cuanto menor es la dispersión mayor la precisión. Se representa por la proporción entre el número de

predicciones correctas (tanto positivas como negativas) y el total de predicciones, y se calcula mediante la ecuación:

$$AC = \frac{a+d}{a+b+c+d}$$

Figura 5.1: Precisión

Por otro lado, la Exactitud o, en inglés, "Precision" se refiere a lo cerca que está el resultado de una medición del valor verdadero. En términos estadísticos, la exactitud está relacionada con el sesgo de una estimación. También se conoce como Verdadero Positivo (o "True positive rate"). Se representa por la proporción entre los positivos reales predichos por el algoritmo y todos los casos positivos. Es decir, de todos los paquetes anómalos, cuántos ha predicho correctamente el algoritmo que lo son. Se calcula según la ecuación:

$$P = \frac{d}{b+d}$$

Figura 5.2: Exactitud

La Figura 5.3 ayuda a ver de forma práctica la diferencia entre precisión y exactitud. Así, por ejemplo, la figura (b) representa un resultado exacto y preciso, mientras que la (c) es preciso, pero no exacto y la (a) no es ni una cosa ni la otra.

Nota: En inglés se usa el término "precision" para la exactitud y el término español precisión se refiere a "accuracy". Puede llevar a confusión.

5.2.2. Sensibilidad y Especificidad.

La sensibilidad y la especificidad son dos valores que nos indican la capacidad de nuestro estimador para discriminar los casos positivos, de los negativos. La sensibilidad es la fracción de verdaderos positivos, mientras que la especifidad, es la fracción de verdaderos negativos.

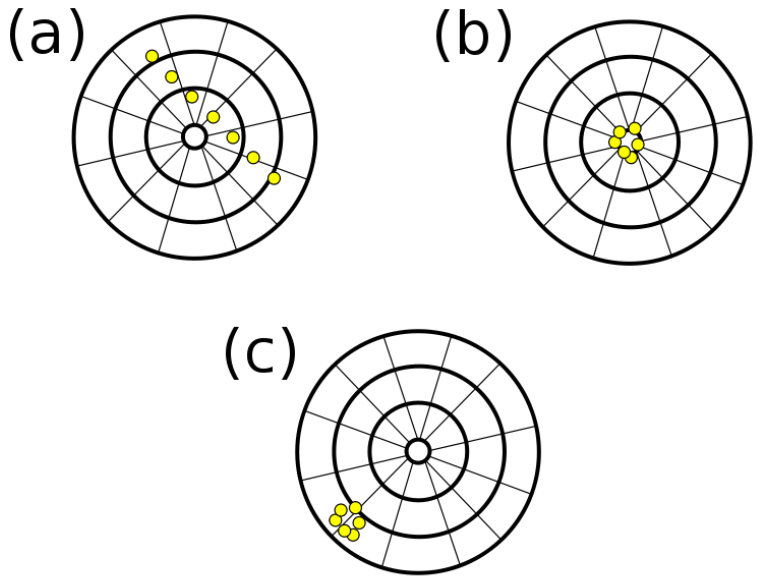


Figura 5.3: Precisión vs Exactitud (Public Domain)

La Sensibilidad ("Recall"), también se conoce como Tasa de Verdaderos Positivos (True Positive Rate) (TP). Es la proporción de casos positivos (paquetes anómalos) que fueron correctamente identificados por el algoritmo. Se calcula según la ecuación:

$$TP = \frac{d}{c+d}$$

Figura 5.4: Sensibilidad

La Especificidad, por otra parte, es la Tasa de Verdaderos Negativos, ("true negative rate" o TN). Se trata de los casos negativos (paquetes legítimos) que el algoritmo ha clasificado correctamente. Su ecuación es:

$$TN = \frac{a}{a+b}$$

Figura 5.5: Especificidad

La exactitud y la sensibilidad nos están indicando la relevancia de los resultados. Por ejemplo, un algoritmo muy exacto, (P alto) nos dará muchos más resultados relevantes que irrelevantes, mientras que un algoritmo muy específico, (TP alto), será el que detecte la mayoría de resultados de interés (los primeros).

La conveniencia de usar una métrica otra como medida del estimador dependerá de cada caso en particular y, en concreto, del "coste" asociado a cada error de clasificación del algoritmo. En nuestro trabajo, es más importante que la fracción de verdaderos positivos sea alta, es decir que el algoritmo sea muy sensible y haya un mayor número de paquetes anómalos detectados correctamente. El coste de un falso negativo, es decir, un paquete malicioso dado como normal, podría ser un factor de riesgo. En otros casos, es más interesante priorizar la especificidad sobre la sensibilidad. Por ejemplo, las pruebas que buscan detectar individuos sanos [21].

5.2.3. Otros términos derivados de la matriz de confusión.

Dejamos para el final otro par de métricas que también se usan habitualmente para la evaluación de algoritmos: la tasa de falsos positivos y la de falsos negativos.

La tasa de Falsos Positivos, (False Positive rate, FP) es la proporción de casos negativos que fueron erróneamente clasificados como positivos por el algoritmo. Es decir, de los paquetes legítimos, cuántos clasificó el algoritmo erróneamente como anómalos. Se calcula según la siguiente ecuación:

$$FP = \frac{b}{a+b}$$

Figura 5.6: Tasa de Falsos Positivos

La tasa de Falsos Negativos (False Negative rate, FN) es la proporción de casos positivos incorrectamente clasificados (paquetes anómalos no detectados). Su ecuación es:

$$FN = \frac{c}{c+d}$$

Figura 5.7: Tasa de Falsos Negativos

5.3. Primera prueba de detección de tráfico anómalo

En esta primera prueba hemos optado por usar un mapa de neuronas de 10x10 y las características del tráfico que hemos utilizado se detallan a continuación:

Los atributos **categóricos** seleccionados para este análisis son los siguientes:

- **ip.scr**: IP origen.
- **ip.dst**: IP destino.
- **ip.srcport**: Puerto TCP/UDP origen.
- **ip.dstport**: Puerto TCP/UDP destino.
- **ip.proto**: Protocolo utilizado en campa de transporte (TCP/UDP).
- **modbus.fuc_code**: Código de función perteneciente a la trama Modbus.

Por otro lado, las características del trafico clasificadas como **numéricas** se detallan a continuación:

- **ip.len**: Tamaño del paquete en la cabecera IP.
- **mbtcp.len**: Tamaño de trama del Protocolo Modbus.
- **modbus.reference_num**: Indica qué valor de datos o rango de valores se solicitan.

Como se explicaba en el Capítulo 4.3, durante el preprocesado debemos realizar la normalización de los datos y adaptarlos al formato exigido por la herramienta para poder procesarlos.

	Predicho: Legítimo	**Predicho: Anómalo**
Real: Legítimo	898.308	45
Real: Anómalo	537	868

Tabla 5.3: Matriz de confusión para la primera prueba

Tras la ejecución del proceso obtenemos los ficheros descritos al principio de este capítulo. En este caso, el número de neuronas con observaciones es de 47 por lo que tenemos un porcentaje de neuronas vacías del 53 % ya que el mapa utilizado es de 10x10. Finalmente, después de recolectar todos los datos en bruto devueltos por el proceso y procesarlos a través de los *scrips*, logramos construir la matriz de confusión, Figura 5.3, y todas las métricas descritas en el presente capitulo.

Gracias a la matriz de confusión podemos comprobar que el proceso de análisis de datos realizado ha obtenido una Precisión del 99,93 %.

De los 898845 paquetes que han sido clasificados como legítimos, el modelo ha clasificado correctamente 898308. De los 913 paquetes clasificados como anómalos, 868 eran, efectivamente, paquetes que presentaban anomalías, resultando en una Exactitud del 95,07 %.

La alta precisión y exactitud obtenida para el análisis podría hacernos creer erróneamente que los resultados obtenidos son muy buenos y que el modelo clasifica muy bien el tráfico, pero no es del todo correcto. Se trata de un resultado equivocado puesto que la muestra de datos es muy desigual en cantidad de paquetes legítimos y anómalos, siendo estés últimos un 0,158 % del total, como hemos explicado en la Sección 4.2.1.

Para ayudarnos a evaluar correctamente el modelo desarrollado, además de las métricas anteriores se deben observar aquellos paquetes que son de un tipo pero han sido clasificados como el otro. La sensibilidad (Recall) y la especificidad nos ayudan a medir este suceso en el que, por ejemplo, de los 1405 paquetes anómalos reales, 537 han sido clasificados como legítimos, dando como resultado una sensibilidad del

61,77 %, y por otro lado, siendo un total de 898353 paquetes legítimos, el algoritmo solo ha clasificado como anómalos 45, proporcionando una especificidad del 99,99 %.

Eso indica, por lo tanto, que tenemos un porcentaje alto de Falsos Negativos 38,22 % mientras que el el tanto por ciento de Falsos Positivos es despreciable.

Profundizando más en la clasificación realizada por el modelo respecto al tráfico real, en la Tabla 5.4 podemos observar el número de paquetes anómalos presentados por cada captura de tráfico y los que el modelo ha detectado. En lineas generales vemos que el sistema detecta todos los ataques excepto el presente en la captura de datos *Send a fake command modbus 6RTU with operate*, que se trata de una inyección en la red de una petición Modbus falsa. La razón por la que el sistema no ha sido capaz de detectar ninguno de los paquetes clasificados como anómalos, puede ser debido a la pequeña cantidad de paquetes maliciosos presentes, solo 10 para esta captura de tráfico en particular. Para el resto de ataques podemos observar que el número de paquetes es mayor y el sistema detecta, en mayor o menor medida, estos paquetes maliciosos.

Paquete de datos	**Tráfico anómalo real**	**Tráfico anómalo detectado por el modelo**
CnC uploading exe modbus 6RTU with operate	121	65
exploit ms08 netapi modbus 6RTU with operate	1.199	780
moving two files modbus 6RTU	75	23
send a fake command modbus 6RTU with operate	10	0
Total	1.405	868

Tabla 5.4: Comparación real predicho primera prueba

5.4. Segunda prueba de detección de tráfico anómalo

En esta segunda prueba hemos optado por usar un mapa de neuronas de 20x20 y hemos modificado las características utilizadas, añadiendo unas nuevas y borrando otras, resultando como sigue:

Los atributos **categóricos** seleccionados para este análisis son los siguientes:

- **ip.scr**: IP origen.
- **ip.dst**: IP destino.
- **ip.srcport**: Puerto TCP/UDP origen.
- **ip.dstport**: Puerto TCP/UDP destino.
- **ip.proto**: Protocolo utilizado en campa de transporte (TCP/UDP).
- **mbtcp.trans_id**: Identificador de transacción de cada trama Modbus.
- **modbus.fuc_code**: Código de función perteneciente a la trama Modbus.

Las características del trafico clasificadas como **numéricas** se detallan a continuación:

- **frame.len**: Tamaño del paquete.
- **ip.len**: Tamaño del paquete en la cabecera IP.
- **tcp.len**: Tamaño del paquete en la cabecera IP.
- **udp.length**: Tamaño del paquete en la cabecera IP.
- **mbtcp.len**: Tamaño del paquete en la cabecera IP.
- **mbtcp.len**: Tamaño de trama del Protocolo Modbus.
- **modbus.reference_num**: Indica qué valor de datos o rango de valores se solicitan.
- **modbus.bit_cnt**: Número de registros (1 bit) a leer/escribir.
- **modbus.word_cnt**: Número de registros (16 bits) a leer/escribir
- **modbus.num_reg**: Número de registros (1/16 bits) a leer/escribir

Tras la normalización y ejecución del proceso, obtenemos la matriz de confusión, Tabla 5.5, y todas las métricas asociadas que explicaremos a continuación.

	Predicho: Legítimo	**Predicho: Anómalo**
Real: Legítimo	898.353	0
Real: Anómalo	937	468

Tabla 5.5: Matriz de confusión para la segunda prueba

En este caso, el mapa de neuronas utilizado es de 20x20, observando que hay 147 neuronas con observaciones, obteniendo un 63,25 % de neuronas vacías.

En esta segunda prueba hemos obtenido una Precisión del 99,89 %, algo más baja si la comparamos con la primera prueba, pero una cantidad despreciable. Además, la Exactitud es del 100 % ya que ningún paquete malicioso han sido clasificados como legítimo. Esto nos permite afirmar que si el sistema ha detectado un paquete como anómalo, es muy probable que lo sea, pudiendo dar mucha confianza al sistema en este sentido.

Además, tenemos una Especificidad del 100 %, que se traduce en una tasa de Falsos Positivos del 0 %, es decir, el sistema ha clasificado correctamente todos los paquetes legítimos como tal. Por otra parte, hemos obtenido una Sensibilidad del 33.30 %, dicho de otro modo, una tasa de Falsos Negativos del 66,70 %. Esto quiere decir que el modelo desarrollado detecta 1 de cada 3 paquetes anómalos, o lo que es lo mismo, 2 de cada 3 paquetes maliciosos no son detectados por el sistema, ya que son clasificados como legítimos.

Haciendo hincapié en la tasa de Falsos Negativos y en la Sensibilidad, que son los estadísticos más interesantes en este trabajo, permitiendo detectar los paquetes anómalos clasificados como legítimos y la capacidad del sistema para detectar el mayor número de paquetes anómalos presentes, respectivamente. Comparando los resultados con los realizados en la prueba anterior, observamos que la tasa de Falsos Negativos se ha incrementado prácticamente un 30 % y la Sensibilidad se ha reducido en la misma cantidad. Esto nos permite afirmar que, en lineas generales, el sistema desarrollado en la segunda prueba es peor que el primero, puesto que el sistema

detecta menos paquetes anómalos aunque se ha reducido la tasa de Falsos Positivos a cero.

Paquete de datos	Tráfico anómalo real	Tráfico anómalo detectado por el modelo
CnC uploading exe modbus 6RTU with operate	121	33
exploit ms08 netapi modbus 6RTU with operate	1.199	419
moving two files modbus 6RTU	75	16
send a fake command modbus 6RTU with operate	10	0
Total	1.405	468

Tabla 5.6: Comparación real predicho segunda prueba

Observando la Tabla 5.6 y comparando los resultados con los obtenidos en la primera prueba, Tabla 5.4, vemos que la cantidad de paquetes anómalos detectados por el sistema son menores para los distintos *datasets*. Además, para la captura de datos *send a fake command modbus 6RTU with operate* tampoco hemos sido capaces de detectar ninguno de los maliciosos.

5.5. Tercera prueba de detección de tráfico anómalo

Para la tercera y última prueba, hemos utilizado las mismas características del tráfico empleadas en la segunda prueba 5.4 y hemos optado unicamente por cambiar el tamaño del mapa de 20x20 a 30x30 para estudiar el comportamiento de la red ante este factor.

Posterior al proceso de preparación de datos obtenemos los resultados mostrados en la matriz de confusión, Figura 5.7:

Para esta última prueba, como ya hemos comentado al principio del presente aparatado, hemos utilizado un mapa de neuronas de 30x30. De las 900 posibles neuronas que podemos obtener, hemos visto que en esta ejecución el mapa

	Predicho: Legítimo	**Predicho: Anómalo**
Real: Legítimo	898.351	2
Real: Anómalo	77	1.328

Tabla 5.7: Matriz de confusión para la tercera prueba

está formado por 363 neuronas con observaciones. Los resultados obtenidos permiten observar que tenemos un 40,33 % de neuronas con observaciones, dicho de otra manera, el porcentaje de neuronas vacías es prácticamente del 60 %.

En esta prueba, tanto la Precisión como la Exactitud obtenida en los análisis es del 99 %. Estos resultados nos dicen que la proporción de aciertos entre el total de muestras del sistema es muy alta, además de que la probabilidad de que la clasificación establecida por el sistema sea correcta, tanto para los paquetes legítimos como los anómalos.

La Sensibilidad obtenida en esta prueba es del 95 %, es decir, el sistema es muy sensible y detecta muchos de los paquetes clasificados como anómalos. Este suceso podría ocurrir si el modelo detectase la mayor parte de los paquetes cómo anómalos, indiscriminando entre paquetes legítimos y anómalos. Sin embargo, dado que hemos obtenido una Especificidad del 99,9 %, sabemos que esto no es así y que el sistema detecta bien entre unos y otros. Además, la tasa de Falsos Positivos es despreciable, solo 2 paquetes legítimos han sido clasificados como tráfico malicioso y, la tasa de Falsos Negativos es del 0,05 %, 77 de un total de 1405 paquetes maliciosos han sido detectados como tráfico legítimo.

Los estadísticos presentados anteriormente nos permiten afirmar que el sistema desarrollado en esta tercera prueba clasifica muy bien el tráfico, segmentando perfectamente entre tráfico anómalo y legítimo y mostrando unos resultados muy buenos.

Comparando los resultados obtenidos en esta prueba, Tabla 5.8, con las dos anteriores observamos que los resultados son mejores. En total se han detectado

Paquete de datos	**Tráfico anómalo real**	**Tráfico anómalo detectado por el modelo**
CnC uploading exe modbus 6RTU with operate	121	98
exploit ms08 netapi modbus 6RTU with operate	1.199	1.199
moving two files modbus 6RTU	75	31
send a fake command modbus 6RTU with operate	10	0
Total	1.405	1.328

Tabla 5.8: Comparación real predicho tercera prueba

1.328 paquetes anómalos, comparando con los 868 de la primera prueba o los 468 de la segunda. Hemos sido capaces de localizar todos los paquetes maliciosos relacionados con el *dataset exploit ms08 netapi modbus 6RTU with operate* pero, al igual que las otras dos pruebas, no hemos podido detectar ninguno de los paquetes pertenecientes a la captura *send a fake command modbus 6RTU with operate*.

Capítulo 6

Conclusiones y trabajos futuros

En este último capítulo procedemos a exponer las conclusiones obtenidas sobre el desarrollo del presente proyecto y se comentarán algunas ideas de trabajos futuros.

6.1. Conclusiones

En este proyecto se ha presentado un trabajo de investigación cuyo objetivo principal es el estudio del comportamiento del tráfico en redes IoT mediante redes de neuronas artificiales del tipo SOM y proporcionar un mecanismo de detección de anomalías adaptado a este tipo de entornos.

Las limitaciones del *dataset* respecto a la infraestructura simulada para la generación del tráfico puede influir en las pruebas realizadas, pues creemos que una infraestructura real es mucho más compleja. Aunque se han elaborado varios conjuntos de datos con diferentes configuraciones, en general, la infraestructura simulada es muy sencilla porque el número de dispositivos utilizados no es comparable con una infraestructura real. Además, el tráfico Modbus generado contiene un número reducido de funciones de todas las permitidas y la cantidad de operaciones manuales generadas es demasiado baja.

El tráfico clasificado como malicioso, como ya hemos comentado en la memoria, es solo un 0,158 % del total de paquetes presentes en el *dataset*. Igualmente, no se

abordan muchas de las tipologías de ataques que se pueden realizar en contra de este tipo de sistemas. Más concretamente, hay un tipo de ataque que tiene como objetivo este tipo de infraestructuras con el fin interrumpir el servicio que estas proporcionan. Estos son los ataques de denegación de servicio (DoS y DDos). Este tipo de ataques son muy populares en contra de estos sistemas, por lo que es una limitación importante no disponer de alguna captura con este tipo de tráfico para poder entrenar el sistema y medir los resultados.

La estrategia de etiquetado seguida por los autores hace que sea muy difícil detectar todos los paquetes clasificados como anómalos, tal y como se ha explicado en el Apartado 4.2. Sin embargo, hemos obtenido unos resultados generales muy favorables, más concretamente en la tercera prueba, ya que hemos conseguido detectar la mayor parte de los paquetes maliciosos presentes en cada *dataset*, lo que nos permite detectar los ataques en la red. Esto no es así para la captura de datos *send a fake command modbus 6RTU with operate*, puesto que no hemos podido detectar ninguno de los paquetes anómalos en ninguna de las tres pruebas. Esto puede ser debido a varios factores, como el reducido número de paquetes presentes o a que los datos no son demasiado representativos, ya que como se indica en la descripción del *dataset*, Tabla 4.2, los comandos se envían desde un RTU comprometido. Si la comunicación entre la RTU comprometida y el otro dispositivo es frecuente, aunque no sea legítima, tendríamos que fijarnos solo en las características relacionadas con Modbus, lo que dificulta la detección puesto que las características no tienen que presentar necesariamente valores anómalos .

El determinismo presente en este tipo de redes nos ha permitido validar nuestra principal hipótesis, es posible detectar anomalías en redes IoT a través de redes de neuronas artificiales de tipo SOM, lo que implica que puede ser un buen mecanismo de seguridad en este tipo de entornos. La baja tasa de Falsos Negativos corrobora esta afirmación, permitiendo detectar la mayor parte del trafico malicioso y, los pocos Falsos Positivos, producirían un número reducido de alertas erróneas. Además, estamos haciendo una clasificación por paquete. Como hemos detectado paquetes maliciosos de prácticamente todos los ataques, quiere decir que el sistema sería capaz de desencadenar una alerta para todo el comportamiento malicioso relacionado con ese paquete. Aún así, no se debe confiar en un único mecanismo de seguridad e,

indudablemente, se mejoraría la detección de amenazas mediante la correlación de eventos desde diferentes fuentes y tecnologías, además de la utilización de un sistema de gestión de eventos (SIEM) para centralizar todas ellas.

Por lo tanto, concluimos que es posible emplear un sistema de detección de anomalías basado en redes de neuronas del tipo SOM en el ámbito de las redes IoT de forma eficaz.

6.2. Trabajos futuros

A continuación se detallan los siguientes puntos como posibles desarrollos futuros:

Sería interesante realizar pruebas con este mismo conjunto de datos pero a nivel de flujo de datos en lugar de paquetes. Esta tarea puede realizarse mediante la herramienta ***pkt2flow*** [13], publicada en Github y que nos permite agrupar capturas de tráfico de red por flujos. Quedaría presente el desafío de etiquetar los flujos de manera correcta.

Conociendo las limitaciones presentes en este *dataset* sugerimos buscar algún otro conjunto de datos sobre el que realizar pruebas y comparar los resultados obtenidos con los del presente trabajo. El nuevo conjunto de datos puede estar enfocado a redes que no pertenezcan a ningún SCI y a otros protocolos IoT que no sean Modbus, como: MQTT (*MQ Telemetry Transport*), AMQP (*Advanced Message Queuing Protocol*), CoAP (*Constrained Application Protocol*), etc.

Con el objetivo de validar completamente el trabajo realizado y verificar que sí es posible la detección de anomalías en redes IoT reales, sería ideal poder probar el sistema en un entorno en real en producción o, en su defecto, con un *dataset* compuesto por datos reales, aunque, como ya se ha comentado en la memoria, es una tarea difícil debido a la sensibilidad que tienen los datos en este tipo de entornos.

Apéndice A

Glosario de acrónimos

1. ***APT*** (página 24): Advanced Persistent Threat (Amenaza Avanzada Persistente).
2. ***DDoS*** (página 76): Distributed Denial of Service (Denegación de Servicio Distribuido).
3. ***DoS*** (página 76): Denial of Service (Denegación de Servicio).
4. ***RFID*** (página 6): Radio Frequency Identification (Identificación por Radiofrecuencia)
5. ***SIEM*** (página 77): Security Information and Event Management (Gestión de Eventos e Información de Seguridad)

Apéndice B

Glosario de términos

1. ***Exploit*** (páginas 41, 42 y 45): Herramienta de software diseñada para aprovechar un fallo en un sistema informático, normalmente con fines maliciosos, como la instalación de malware.

2. ***Hacktivistas*** (página 23): Persona que obtiene acceso no autorizado a archivos o redes informáticas con el fin de promover fines sociales o políticos.

3. ***Three-way Handshake*** (página 41): 3-Way Handshake es un método de tres pasos utilizado en una red TCP/IP para crear una conexión entre un host/cliente local y un servidor.

4. ***Metasploit*** (páginas 44, 45): El Proyecto Metasploit es un framework de seguridad informática que proporciona información sobre vulnerabilidades de seguridad y ayuda en pruebas de penetración y desarrollo de firmas IDS. Es propiedad de la empresa de seguridad Rapid7.

5. ***Payload*** (página 42): El Payload (carga útil) se refiere al componente de un malware que ejecuta una actividad maliciosa.

6. ***Port Mirroring*** (página 31): Un Port Mirroring, también conocida como SPAN (Switched Port Analyzer), es un método de monitorización del tráfico de red. Con la duplicación de puertos habilitada, un switch envía una copia de todos los paquetes de red que se ven en un puerto (o en una VLAN completa) a otro puerto, donde se puede analizar el paquete.

7. ***Proxychains*** (página 42): Proxychains es un servidor proxy que soporta los protocolos de internet HTTP(S), SOCKS4 y SOCKS5 , funciona sobre distribuiciones Linux/GNU, BSD y Mac OS X (plataformas Unix). Proxychains permite que cualquier conexión TCP hecha por un programa dado siga una serie de proxies (de los protocolos mencionados) hasta su destino.

8. ***Sandbox*** (página 40): Mecanismo para ejecutar programas con seguridad y de manera aislada del Sistema Operativo local.

9. ***Spear-Phishing*** (página 24): El Spear-Phishing es una estafa de correo electrónico o comunicaciones dirigida a personas, organizaciones o empresas **específicas**. Aunque su objetivo a menudo es robar datos para fines maliciosos, los cibercriminales también pueden tratar de instalar malware en la computadora de la víctima.

Bibliografía

[1] Countercraft: "es imposible proteger tu compañía de los ciberataques al 100 %". [Online]. Available: https://byzness.elperiodico.com/es/innovadores/20190611/countercraft-startup-ciberseguridad-7497453.

[2] Definition of Big Data. Gartner. [Online]. Available: https://www.gartner.com/it-glossary/big-data/.

[3] Industrial Control System. Trend Micro. [Online]. Available: https://www.trendmicro.com/vinfo/us/security/definition/industrial-control-system.

[4] Modbus Application Protocol Specification. Modicon. [Online]. Available: http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf.

[5] The Man Who Found Stuxnet - Sergey Ulasen in the Spotlight. [Online]. Available: https://eugene.kaspersky.com/2011/11/02/the-man-who-found-stuxnet-sergey-ulasen-in-the-spotlight/.

[6] Understanding Modbus. [Online]. Available: https://technologyuk.net/computing/computer-networks/industrial-networks/modbus.shtml.

[7] Defensa activa e inteligencia: de la teoría a la práctica . INCIBE-CERT. [Online]. Available: https://www.incibe-cert.es/blog/defensa-activa-e-inteligencia-teoria-practica.

[8] Wissam Aoudi, Mikel Iturbe, and Magnus Almgren. Truth will out: Departure-based process-level detection of stealthy attacks on control systems. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, New York, NY, USA, 2018.

[9] Diferencias entre TI y TO. INCIBE-CERT. [Online]. Available: https://www.incibe-cert.es/blog/diferencias-ti-to.

[10] Evaluación de modelos de clasificación. RPubs. [Online]. Available: https://rpubs.com/chzelada/275494.

[11] Advanced Factories. Cómo la industria 4.0 y el internet de las cosas (iot) están conectados. [Online]. Available: https://www.advancedfactories.com/la-industria-4-0-internet-las-cosas-iot-estan-conectados/.

[12] Garitano, Iñaki and Iturbe, Mikel and Aranaza-Nuño, Ignacio and Uribeetxeberria, Roberto and Zurutuza, Urko. Sistema de Detección de Anomalías para protocolospropietarios de Control Industrial. In *RECSI*, 2014.

[13] Pkt2flow. Github. [Online]. Available: https://github.com/caesar0301/pkt2flow.

[14] INCIBE-CERT. La ciberseguridad en la industria 4.0. [Online]. Available: https://www.incibe-cert.es/blog/ciberseguridad-industria-4-0.

[15] Mikel Iturbe. Visualizing Networks Flows and Related Anomalies in Industrial Networks using Chord Diagram and Whitelisting. In *VISIGRAPP (2:IVAPP)*, 2016.

[16] Mikel Iturbe. Detección de ataques en entornos industriales, 2018. [Online]. Available: https://www.youtube.com/watch?v=A-cfFamw5Y0.

[17] Kevin Ashton. That "Internet of Things" Thing. In *RFID Journal, vol 22*. Gartner, July 2009.

[18] Markus Koch and Ralf C. Schlaepfer. Challenges and solutions for the digital transformation and use of exponential technologies, 2015. [Online]. Available: https://www2.deloitte.com/content/dam/Deloitte/ch/Documents/manufacturing/ch-en-manufacturing-industry-4-0-24102014.pdf.

[19] Las claves de los últimos ataques en sistemas de control industrial. INCIBE-CERT. [Online]. Available: https://www.incibe-cert.es/blog/las-claves-los-ultimos-ataques-sistemas-control-industrial.

[20] Antoine Lemay and Jose M. Fernandez. Providing SCADA Network Data Sets for Intrusion Detection Research. In *9th Workshop on Cyber Security Experimentation and Test (CSET 16)*, Austin, TX, 2016. USENIX Association. [Online]. Available: https://www.usenix.org/conference/cset16/workshop-program/presentation/lemay.

[21] Machine Learning a tu alcance: La matriz de confusión. ElevenPaths. [Online]. Available: https://empresas.blogthinkbig.com/ml-a-tu-alcance-matriz-confusion/.

[22] Mikel Iturbe, José Camacho, Iñaki Garitano, Urko Zurutuza and Roberto Uribeetxeberria. On the Feasibility of Distinguishing Between Process Disturbances and Intrusions in Process Control Systems Using Multivariate Statistical Process Control. 2017.

[23] Modbus Protocol Reference Guide. Modicon. [Online]. Available: http://modbus.org/docs/PI_MBUS_300.pdf.

[24] David Olano. Desarrollo de un sistema de detección de tráfico anómalo en el contexto de la Internet de las Cosas. Febrero 2016. [Online]. Available: https://repositorio.uam.es/bitstream/handle/10486/669990/Olano_Arias_David_tfg.pdf?sequence=1.

[25] Esteban Pérez-López. Los sistemas SCADA en la automatización industrial. In *Tecnología en Marcha. Vol. 28, No 4*, pages 3–14, 2015.

[26] Rob van der Meulen. Gartner Says 8.4 Billion Connected "Things" Will Be in Use in 2017, Up 31 Percent From 2016, 2017. [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2017-02-07-gartner-says-8-billion-connected-things-will-be-in-use-in-2017-up

[27] Introduction to IoT. [Online]. Available: https://www.netacad.com/es/courses/iot/introduction-iot.

[28] Zhu Bonnie and Shankart Sastry. SCADA-specific intrusion detection/prevention system: a survey and taxonomy. In *Proceedings of the 1st Workshop on Secure Control systems (SCS)*, Vol. 11, 2010.