Trabajo Fin de Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Análisis de vulnerabilidades y securización de protocolo Modbus/TCP

Ibai Marcos Cincunegui

Director: Ricardo J. Rodríguez Fernández

Ponente: José Merseguer Hernáiz

Escuela de Ingeniería y Arquitectura
Universidad de Zaragoza

Septiembre de 2018
Curso 2017/2018

Agradecimientos

Agradecer primeramente a toda la gente de CIRCE por hacer posible este proyecto y en especial a Giuseppe Porzio.
A toda mi familia y amigos por todo el apoyo. A mi madre por haber confiado en mí hasta el final.
Por último a Ricardo por brindarme esta oportunidad.

RESUMEN

Modbus/TCP es un protocolo de comunicaciones estandarizado de las redes de control industrial, utilizado principalmente en los controladores lógico programables y por sistemas o equipos de supervisión y adquisición de datos (SCADA). Inicialmente, estos sistemas se encontraban en redes aisladas, con lo que la seguridad de los datos no era un factor prioritario ni determinante. Sin embargo, la rápida evolución de la industria y de la tecnología, con la aparición de Internet, ha llevado a estos sistemas a hacer uso del protocolo Modbus en redes abiertas. Esta nueva exposición de la información en los sistemas SCADA ha generado una necesidad de securización de los datos, antes inexistente.

Con el objetivo de introducir una herramienta que permita el inicio de la securización del protocolo, en este trabajo se ha desarrollado un entorno de simulación del protocolo Modbus/TCP que permite analizar soluciones de seguridad en este tipo de redes sin necesidad de disponer de un entorno real. Para validar el simulador, se ha dispuesto de un entorno real con una red SCADA que hace uso del protocolo Modbus/TCP. De esta red, se han recogido muestras del tráfico de red, analizándolas en detalle. Tras la evaluación del trabajo realizado, se ha obtenido de forma satisfactoria un escenario simulado que reproduce un escenario equivalente al analizado. Además, se han podido reproducir diferentes ataques al protocolo Modbus/TCP en el simulador, lo que permite disponer de un entorno donde se pueden probar diferentes soluciones de seguridad.

La realización de este trabajo sirve como punto de partida para nuevos desarrollos orientados hacia la securización del ya estandarizado Modbus/TCP, acelerar la integración de nuevas tecnologías en las redes SCADA o facilitar el estudio, conocimiento y solución de los puntos vulnerables en sistemas similares.

Índice general

1. Introducción **1**

2. Conceptos Previos **5**
2.1. Redes industriales 5
2.2. Arquitectura del protocolo Modbus/TCP 6

3. Trabajo relacionado **11**

4. Análisis de tráfico Modbus/TCP en escenario real **13**
4.1. Entorno de experimentación: Centro de investigación CIRCE 13
4.2. Entorno de experimentación 14
4.3. Análisis de capturas 17
4.4. Resultados 19
4.4.1. Estadísticas de tipos de tráfico 19
4.4.2. Información de red 20
4.4.3. Estadísticas de características del tráfico Modbus 20
4.4.4. Resultados inyección de mensajes 21

5. Simulador del protocolo Modbus/TCP **25**
5.1. Requisitos funcionales y no funcionales 25
5.2. Simulador de dispositivo esclavo Modbus 26
5.2.1. Dependencias software 27
5.2.2. Estructura del simulador 27
5.2.3. Funcionamiento 28
5.3. Simulador de maestro Modbus 31
5.3.1. Dependencias software 32
5.3.2. Estructura del simulador 32
5.3.3. Funcionamiento 33

6. Reproducción del entorno real y validaciones **35**
6.1. Validación de los simuladores 35
6.1.1. Validación esclavo 35
6.1.2. Validación del maestro 37
6.2. Descripción del escenario de simulación 38

6.3. Reproducción de intrusiones 39
6.3.1. Descubrimiento de funciones 39
6.3.2. Vulneración de la confidencialidad 39
6.3.3. Denegación del servicio 40

7. Conclusiones y trabajo futuro **43**
7.1. Conclusiones 43
7.2. Trabajo futuro 44

Bibliografía **44**

A. Desglose de trabajo **47**

Índice de tablas

2.1. Tipos de datos Modbus. 8

4.1. Estadísticas por cada pareja de ficheros de captura. 19
4.2. Fragmento de fichero de registro del análisis de tráfico de red. 20

Índice de figuras

2.1. Estructura de red típica de una red SCADA. 6
2.2. Estructura general de mensaje Modbus (extraído de [12]). 7
2.3. Cabeceras y campos de datos de mensajes Modbus/RTU y Modbus/TCP (extraído de [11]). 7
2.4. Máquina de estados genérica del servidor Modbus (extraído de [12]). . . . 9
2.5. Modelo de bloques de datos en un cliente Modbus (extraído de [12]). . . . 10

4.1. Laboratorio de investigación de CIRCE. 14
4.2. Equipos controladores en el laboratorio de CIRCE. 14
4.3. Throwing Star LAN Tap (extraído de [2]). 15
4.4. Esquemas de la (a) red original y de la (b) red con el sistema de captura desplegado. 16
4.5. Sistema de monitorización de tráfico. 16
4.6. Tabla de datos Modbus del sistema real de estudio. 22

5.1. Diagrama de secuencia de mensaje *Read Coils*. 29
5.2. Diagrama de estados para el procesado de mensajes tipo *Read Coils* (extraído de [12]). 31

6.1. Traza de la comunicación sin ataques. 40
6.2. Contenido de los datos de la comunicación sin ataques. 41
6.3. Traza de la comunicación con ataques de denegación del servicio. 41
6.4. Contenido de los datos de la comunicación durante el ataque de denegación del servicio. 42

A.1. Diagrama de Gantt. 48

Capítulo 1

Introducción

En la década de los 60, donde los sistemas mecánicos eran la pieza clave para la industria y el desarrollo de los procesos industriales, los avances de la tecnología trajeron consigo una nueva forma de automatización: los sistemas eléctricos basados en relés electromagnéticos. El cambio se basaba principalmente en la entrada de los microprocesadores en la industria de embalaje y líneas de montaje en forma de autómatas programable mediante controladores de lógica programables o más conocidos por sus siglas en inglés PLC (*Programmable Logic Controller*). Se creaba entonces el concepto de producción automatizada. Del mismo modo, las fábricas pasaron de sistemas mecánicos aislados, a sistemas electrónicos automatizados con la posibilidad de comunicación. El conjunto de autómatas y equipos que agrupan estos sistemas es lo que hoy día se conoce como redes de control industrial.

A lo largo de los años, se han desarrollado diferentes protocolos de comunicación para mantener el control y la planificación de estos autómatas. Estos protocolos permiten a los diferentes equipos de la red realizar la transmisión de datos necesaria para el correcto funcionamiento del sistema. Con toda esta infraestructura, protocolos e información, nace el concepto de SCADA (del inglés, *Supervisory Control And Data Acquisition*), que define los sistemas de control y adquisición de datos.

A este respecto, el protocolo Modbus [12] es un protocolo de comunicación industrial ampliamente utilizado en las redes de autómatas programables. Modicon Industrial Automation Systems (actualmente la empresa Schneider Electric) [4], fabricante de PLCs, desarrolló el protocolo de forma cerrada para su gama de productos a finales de 1970, publicándose en 1979 y convirtiéndose en un estándar con mucha popularidad en el sector industrial. A partir de este punto, el protocolo Modbus se ha ido extendiendo debido a las diferentes necesidades e intereses de desarrolladores y usuarios. Hoy en día existen diferentes versiones del protocolo Modbus, según el uso y la capa física que emplea (por ejemplo, Modbus ASCII, Modbus Plus o Modbus/TCP entre otros).

En los inicios de las redes compuestas por PLCs, a consecuencia de las líneas de comunicaciones existentes entre los equipos y siendo el protocolo más popularizado Modbus/RTU (usa conexión serie), los sistemas se encontraban en forma de redes aisladas donde el acceso y la comunicación con los equipos se limitaba a la propia infraestructura

física. Con la aparición de Internet y la estandarización del uso de protocolos y equipos que permitían la interconexión de diferentes redes, comenzaron a crearse redes abiertas. Con ello nació Modbus/TCP, una variante de Modbus/RTU que adaptaba el protocolo a las características de las nuevas líneas físicas. Modbus/TCP ha sido la versión del protocolo Modbus más popular en la industria por las ventajas que implica en aspectos como la reducción de costes, la mejora de los procesos de fabricación y la flexibilidad en los procesos de producción.

Sin embargo, esta mejora tecnológica ha traído consigo nuevas problemáticas. Un estudio realizado por Kaspersky Lab sobre la seguridad en sistemas de automatización industrial muestra el aumento de los ataques en las redes industriales [9]. Este aumento de ataques informáticos ha sido una de las consecuencias más evidentes de la exposición de las redes a Internet por los protocolos como Modbus/TCP. Diversas investigaciones que han estudiado el protocolo han demostrado las carencias que posibilitan la vulneración de los principios básicos de la ciberseguridad [15] (la integridad, confidencialidad, disponibilidad y autenticación).

Realizar una evaluación de la seguridad del protocolo Modbus/TCP conlleva una dificultad añadida debido a la criticidad de los datos que se gestionan en los sistemas SCADA. La posible irrupción que implicaría este tipo de investigaciones puede provocar cambios en el funcionamiento habitual de los PLCs, pudiendo hacer entrar al sistema en estados anómalos que impliquen daños materiales de alto valor económico o incluso daños físicos.

Mediante este proyecto, se pretende crear un entorno de simulación donde sea posible analizar y diseñar las soluciones que las redes SCADA y el protocolo Modbus/TCP en particular requieren en cuanto a seguridad se refiere. Esta herramienta permitirá analizar y estudiar la seguridad de los equipos involucrados en comunicaciones Modbus/TCP, así como generar muestras de tráfico equivalentes a un sistema real. Además, un simulador de este tipo será util para auditorías de seguridad donde se realicen procesos de validación, permitiendo así verificar el correcto funcionamiento de dispositivos o programas que hagan uso de Modbus/TCP. Para validar la herramienta desarrollada se ha simulado un entorno de una red real SCADA conocida y contrastado que el comportamiento de la herramienta era igual al comportamiento de la red real. En concreto, en este proyecto se ha contado con la colaboración del centro CIRCE, centro de investigación de procesos ambientales de la Universidad de la Campania “Luigi Vanvitelli” (Italia). El centro ha ofrecido el acceso y uso excepcional de su red SCADA, que hace uso del protocolo Modbus/TCP, para ser objeto de estudio en este trabajo.

De manera adicional, este trabajo proporciona diversas herramientas útiles que permiten un el análisis de dispositivos en redes con Modbus/TCP, facilitando así los procesos de auditoría de seguridad en este tipo de redes.

Organización

En el Capítulo 2 se hace una introducción técnica sobre el protocolo Modbus y el uso más frecuente en las redes SCADA. Además, se detallan las diferencias entre las di-

ferentes implementaciones y se explica la funcionalidad de los mensajes del protocolo. El Capítulo 3 detalla la investigación y análisis realizado sobre herramientas existentes para una auditoría de sistemas que usen el protocolo Modbus. En el Capítulo 4 se estudia el comportamiento y la comunicación mediante el protocolo Modbus/TCP de dos equipos cliente-servidor de una red SCADA real. Finalmente, se hace un análisis del tráfico observado en la red. En el Capítulo 5 explica el desarrollo realizado del entorno de simulación. En el Capítulo 6 se valida y se justifica la utilidad del entorno de simulación de cara al desarrollo de nuevas funcionalidades de seguridad para las redes SCADA, generando ataques y analizando el comportamiento del entorno de simulación y comparándolo con la red real analizada en el Capítulo 4. Por último, el Capítulo 7 concluye el trabajo y establece unas líneas futuras.

De forma ilustrativa se ha introducido en forma de diagrama de Gantt en el Apéndice A el desglose de tareas y horas de trabajo empleadas en la realización de este trabajo.

Capítulo 2

Conceptos Previos

En este capítulo se introduce el protocolo Modbus de manera más técnica, introduciendo en primer lugar los conceptos más relevantes para la comprensión del trabajo realizado. Como ya se ha comentado en el anteriormente, a pesar de las diferentes variaciones que existen del protocolo Modbus, este proyecto se centra únicamente en el estudio y análisis del protocolo en su versión sobre TCP/IP: Modbus/TCP.

2.1. Redes industriales

Para entrar en contexto se debe tener en cuenta la estructura habitual de las redes de control industrial (véase la Figura 2.1). A pesar de las diferencias que puedan existir en las redes, suelen estructurarse de forma similar en cuando a la distribución de funcionalidades e interconexión de los equipos.

El elemento principal de estas redes es el PLC o también conocido como autómata. El equipo realiza las operaciones industriales sobre los dispositivos mecánicos que dispone. Tras el autómata, existe un equipo con las funciones de controlador cuya finalidad es mantener una comunicación con el PLC indicando el tipo de operación sobre los dispositivos. El protocolo Modbus/TCP ofrece la posibilidad de sustituir el equipo controlador por una red de dispositivos que monitorizan y operan con los datos provistos por los autómatas. A continuación, se enumeran diferentes equipos que pueden encontrarse en una red de control industrial:

HMI (*Human-Machine Interface*): Es un equipo que contiene una interfaz de usuario que se utiliza como sistema de configuración y monitorización de los autómatas por administradores de la red.

Unidades de control: Son equipos o servicios que permiten el almacenamiento o procesamiento de la información que se provee y obtiene de los sistemas de adquisición de los datos como PLCs.

Unidades de adquisición de datos: Son todos los dispositivos que aporten información a la red, ya sean PLC o cualquier tipo de sensor o actuador.

Modbus Gateway: En algunos sistemas SCADA es frecuente encontrar dispositivos que hacen uso exclusivo de líneas de comunicación serie, sin soporte del protocolo Modbus en su versión sobre TCP. En estas situaciones es muy común el uso de dispositivos de interconexión que permiten la coexistencia de este tipo de dispositivos y con las redes IP. Estos dispositivos son denominados *Modbus Gateways* o *Converters*.

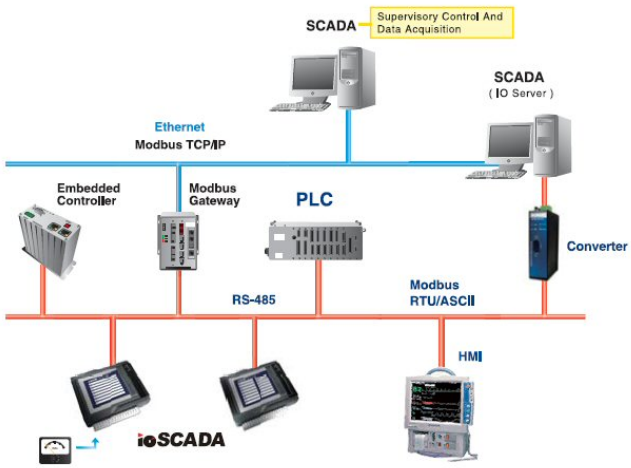


Figura 2.1: Estructura de red típica de una red SCADA.

2.2. Arquitectura del protocolo Modbus/TCP

El protocolo Modbus es un protocolo basado en paso de mensajes con una estructura cliente-servidor o maestro-esclavo. Para el inicio de la comunicación, el dispositivo maestro realiza una conexión TCP al autómata que puede mantener comunicaciones simultáneas con múltiples clientes. Tras el establecimiento de la conexión TCP, el dispositivo maestro transmite un mensaje Modbus indicando al esclavo la operación a realizar. El dispositivo esclavo lee y procesa el mensaje, actúa de forma consecuente (según sea la operación solicitada) y devuelve la respuesta correspondiente.

Modbus/TCP es un protocolo de la capa de aplicación del modelo OSI[1] basado en el paso de mensajes sobre el protocolo de transporte TCP. El conjunto de los datos que incluye una cabecera en la capa de aplicación se denomina ADU (del inglés, *Application Data Unit*), y el segmento de datos que define el mensaje, excluyendo la cabecera, se denomina PDU (*Protocol Data Unit*). Dichos mensajes siguen la estructura descrita en la Figura 2.2. Se dividen en dos partes donde puede diferenciarse los siguientes campos de datos de forma genérica para cualquier mensaje Modbus:

[1]Modelo OSI: modelo de referencia para los protocolos de la red de arquitectura en capas.

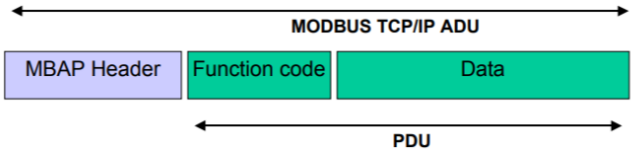


Figura 2.2: Estructura general de mensaje Modbus (extraído de [12]).

MBAP Header (7 bytes): Una cabecera específica de Modbus/TCP, utilizada para identificar los datos de aplicación en la capa TCP. Esta cabecera añade ciertas diferencias con Modbus/RTU (utilizado en comunicaciones serie RS-232 y RS-485), como se muestra en la Figura 2.3:

- **UID** (*Unit Identifier*): Se utiliza para la comunicación con enrutadores ó *Modbus Gateway* que soportan varios esclavos sobre una misma dirección IP.
- **Protocol ID**: Utilizado para identificar el protocolo Modbus en el segmento de datos TCP.
- **Transaction ID**: Número de secuencia utilizado para la identificación y asociación de mensajes pregunta-respuesta.
- **Length**: Identificador de la longitud del mensaje Modbus.

PDU (253 bytes): Denominada por sus siglas en inglés Unidad de Datos de Protocolo. Contiene un número en el campo *Function Code* que identifica el tipo de operación, mientras que en *Data* está la información adicional para la ejecución de dicha operación.

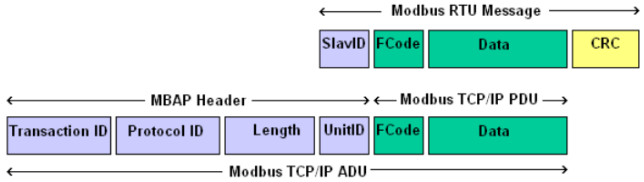


Figura 2.3: Cabeceras y campos de datos de mensajes Modbus/RTU y Modbus/TCP (extraído de [11]).

Independientemente de la arquitectura usada, la comunicación es de tipo maestro-esclavo, siendo el esclavo un autómata programable y el maestro un equipo con la función de **operador central** o **panel de operador**. Las operaciones están relacionadas con los valores de entrada y salida que dispone el PLC que actúa como esclavo.

Cada autómata dispone de unos registros de entrada/salida y una memoria que permite las operaciones de lectura y escritura para el control de los periféricos asociados

al PLC. Existen cuatro tipos de datos básicos en la memoria de un PLC. Estos tipos de datos tienen diferentes permisos según las operaciones que el maestro puede realizar sobre el esclavo. En la Tabla 2.1 pueden diferenciarse las características de los diferentes tipos de datos Modbus.

El protocolo permite limitar las operaciones que un maestro puede realizar sobre los dispositivos asociados al autómata por la lógica de las mismas a través de estos tipos de datos. Por ejemplo, un campo que identifica el valor de un sensor de temperatura no podría ser modificado. La diferencia en la composición de cada tipo de dato permite la gestión de información de diferentes contextos. Así pues, con datos de un bit de longitud se permite la lectura/modificación de los estados binarios de las mecánicas del PLC como el encendido/apagado. Los campos de mayor longitud permiten una gestión con más posibilidades, pudiendo transmitir números enteros, decimales o incluso secuencias de caracteres mediante el uso de varios registros de datos. La diferenciación en estas estructuras permite hacer el protocolo más eficiente en cuanto a la transmisión de datos útiles se refiere.

Tipo de dato	Tamaño	Permiso de acceso	Comentarios
Discret Input	1 bit	Sólo lectura	Provisto por sistemas de entrada/salida.
Coils	1 bit	Lectura-escritura	Este tipo de dato puede ser modificado por un programa de aplicación.
Input Registers	16 bits	Sólo lectura	Provisto por sistemas de entrada/salida
Holding Registers	16 bits	Lectura-escritura	Este tipo de dato puede ser modificado por un programa de aplicación.

Tabla 2.1: Tipos de datos Modbus.

A continuación, se mencionan algunas de las funciones más utilizadas del protocolo para el acceso y modificación de estos datos.

1. Función *Read Coils* (`0x01`): Permite la lectura de bits contiguos de la memoria. Estos bits son también modificables.

2. Función *Read Discrete Inputs* (`0x02`): Permite la lectura de bits de la memoria contiguos, siendo estos únicamente de lectura.

3. Función *Read Holding Registers* (`0x03`): Permite la lectura de múltiples registros (16 bits de datos) contiguos, siendo estos también únicamente de lectura.

4. Función *Read Input Registers* (`0x04`): Permite la lectura de múltiples registros contiguos. Estos registros son modificables.

5. Función *Write Single Coil* (`0x05`): Permite la modificación de un único bit de la memoria.

6. Función *Write Single Register* (`0x06`): Permite la modificación de un único registro de la memoria.

7. Función *Write Multiple Coil* (`0x0F`): Permite la modificación bits contiguos de la memoria.

Cada uno de los mensajes que envía el maestro (cliente) es procesado por el esclavo (servidor) y respondido de forma consecuente. La mecánica de procesado de cada una de las peticiones depende de la configuración del esclavo, la cantidad de funcionalidades que soporte y la petición de un tipo de dato correctamente configurado para su procesado en el PLC.

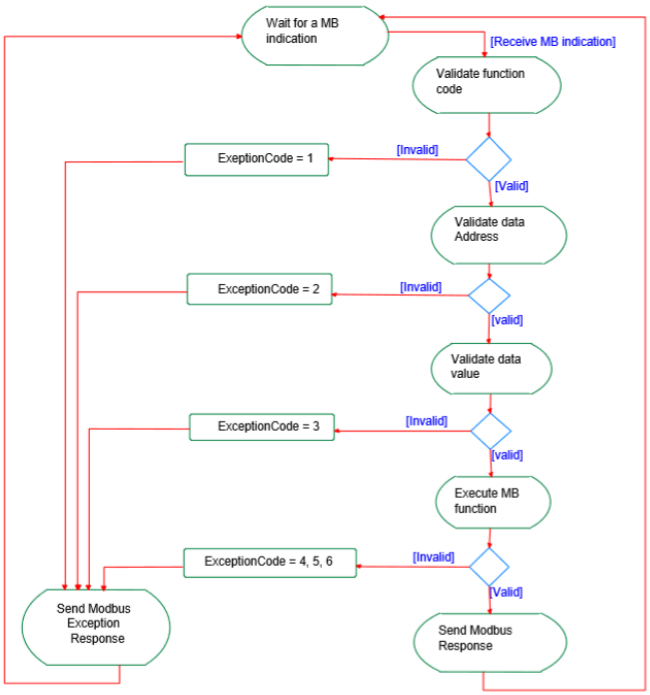


Figura 2.4: Máquina de estados genérica del servidor Modbus (extraído de [12]).

De forma genérica, puede seguirse el proceso de respuesta del servidor según el diagrama de estados de la Figura 2.4. Como se observa, el protocolo define una serie de respuestas a modo de excepción cuando el servidor entra en un estado de error, lo cual puede ocurrir por diversas razones (como un fallo en la ejecución de la función solicitada, errores en los campos requeridos por la función, o que el dispositivo no soporta la funcionalidad solicitada en la petición). En caso de entrar en un estado de excepción, el esclavo

devuelve al maestro el mensaje correspondiente indicando el estado de excepción y la causa del error. En caso de ser procesado correctamente, por cada una de las peticiones se obtiene una estructura de mensaje diferente con los datos sobre la operación solicitada en el mensaje de la consulta.

Para mantener un control del estado de los dispositivos mecánicos que controla el PLC (ya sean de entrada, de salida o de entrada/salida), el protocolo Modbus opera sobre unas tablas de datos donde se realizan las operaciones de lectura y escritura de los mensajes. Estas tablas tienen una relación con la memoria física del autómata de forma que existe una relación directa del estado de la mecánica del PLC con el contenido de las tablas.

Las tablas de datos Modbus pueden tener diferentes tamaños, con la única limitación de un máximo de 65.535 datos por tabla. Esta limitación se debe tamaño máximo del mensaje Modbus y los 2 bytes que se utilizan para el direccionamiento de las tablas. En la Figura 2.5 puede observarse la estructura más estandarizada de los PLCs, donde se asignan tablas de 10.000 direcciones a cada uno de los tipos básicos de Modbus.

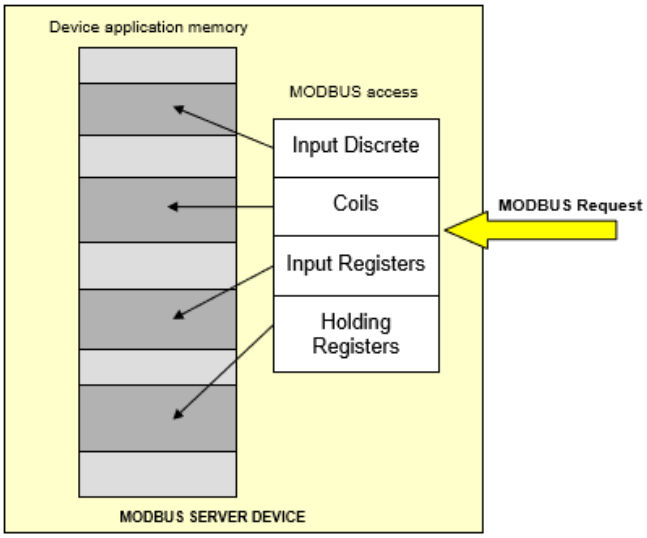


Figura 2.5: Modelo de bloques de datos en un cliente Modbus (extraído de [12]).

Capítulo 3

Trabajo relacionado

En este capítulo se describen aquellos trabajos que guardan alguna relación con este proyecto. En concreto, se describen las herramientas existentes con una funcionalidad similar a la desarrollada en este proyecto, destacando las diferencias relevantes.

En cuanto a las herramientas que realicen una simulación de un dispositivo esclavo, actualmente los fabricantes de PLCs disponen de programas de simulación de PLCs. Sin embargo, estos programas se encuentran bajo licencia comercial, y su objetivo principal es demostrar el funcionamiento de los productos de la propia marca.

Es interesante destacar ModbusPal [3], una herramienta escrita en Java y de código abierto pensada para responder a mensajes Modbus de modo que puede integrarse con otras herramientas que los generen. A pesar de esto, por el hecho de plantear mayormente fines educativos, la herramienta tiene como característica principal la configuración dinámica de las características: el soporte que un PLC dispone sobre los tipos de datos y funciones Modbus se generan dinámicamente con las peticiones que el extremo cliente realiza. Esta flexibilidad, aunque es interesante por su versatilidad, dista en gran medida con el comportamiento real de un PLC, limitando en gran medida las características de seguridad que se pretenden analizar en este proyecto.

De forma análoga a los simuladores de esclavos, las herramientas de simulación de un dispositivo maestro están creadas con fines educativos u orientados al testeo de controladores. A continuación, se destacan algunas de estas herramientas.

QModMaster [14] es una herramienta que permite realizar consultas a una dirección IP y puerto determinados, pudiendo definir de forma precisa cada uno de los campos del mensaje Modbus. La configuración del programa se limita a la definición de los campos de los mensajes dejando en un segundo plano su periodicidad, las ráfagas y el tiempo entre los mensajes. La herramienta simula la transmisión de los mensajes Modbus, pero no realiza una comunicación real de forma continuada. Del mismo modo, ModbusTester [5] tiene una usabilidad muy sencilla, pero con las mismas limitaciones que la herramienta QModMaster. Este tipo de herramientas son de gran utilidad para

validaciones y comprobaciones del correcto funcionamiento de un dispositivo esclavo, pero son inviables para la simulación de comunicaciones reales.

Existen otros trabajos similares a los mencionados que presentan todavía más limitaciones. El grado de acoplamiento, el soporte multiplataforma, la eficiencia temporal y la escalabilidad a nivel de desarrollo son requisitos que no se han mencionado y de carácter prioritario en este proyecto. Sin embargo, estos requisitos no se cumplen en ninguna herramientas analizadas.

Tras la realización del proyecto, la herramienta desarrollada ha incluido funcionalidades no disponibles en las herramientas analizadas, cubriendo mayormente la necesidad de reproducir una comunicación Modbus de forma idéntica a un dispositivo PLC real. De forma adicional y diferenciándose así de las herramientas citadas, se han creado unos simuladores de dispositivos maestro y esclavo de forma totalmente desacoplada, permitiendo así la integración tanto en entornos reales como de simulación o desarrollo.

Conviene también destacar el trabajo [8], donde se desarrolla una herramienta que implementa el protocolo Modbus/TCP usando la librería Scapy [7] (librería de Python para gestión de mensajes de red). Este trabajo se ha utilizado en el presente proyecto para facilitar la escabilidad de la herramienta desarrollada.

Por último, es interesante destacar otros trabajos orientados al análisis de la seguridad del protocolo Modbus, como [15], donde se demuestra de forma práctica las vulnerabilidades del protocolo y los fallos que un sistema SCADA debe considerar: autentificación e integridad de los datos, así como la confidencialidad de las transmisiones y la disponibilidad del servicio.

Capítulo 4

Análisis de tráfico Modbus/TCP en escenario real

En este capítulo se describe y detalla el escenario de trabajo real donde se ha realizado un proceso de recolección de datos de tráfico Modbus/TCP, la metodología llevada a cabo para obtener esta información, y el análisis del tráfico realizado posteriormente.

Una de las mayores necesidades a la hora de realizar una simulación de un sistema es la disponibilidad de información parametrizada del propio sistema en un entorno real para reproducir con la mayor exactitud posible dicho comportamiento. Las redes que implementan el protocolo Modbus suelen ser por lo general infraestructuras críticas, de alto coste económico y con una notoria necesidad de confidencialidad y disponibilidad. Por esta razón, disponer de muestras de tráfico de protocolo Modbus/TCP de un escenario real crea la necesidad de generar y almacenar este tipo de datos. Para solucionar este inconveniente, se ha contado con una red SCADA real donde se han realizado de forma excepcional y supervisada acciones que no son viables en otros centros de tipo similar.

De forma paralela, además, se ha realizado una auditoría de la red bajo estudio, con el objetivo de detectar posibles carencias de seguridad que pudieran afectar posteriormente al tráfico de red Modbus/TCP. Como resultado, se han propuesto diferentes soluciones a las carencias detectadas.

4.1. Entorno de experimentación: Centro de investigación CIRCE

El laboratorio científico CIRCE [1] (*del inglés, Center for Isotopic Research on Cultural and Enviromental heritage*) es un centro de investigación ubicado en Nápoles (Italia) y adscrito a la Universidad de la Campania "Luigi Vanvitelli" que estudia procesos ambientales y procesos físicos de aplicaciones industriales. Para ello, dispone de equipos mecánicos controlados mediante un sistema de control software que, a través del protocolo Modbus/TCP, mantiene una comunicación continua dentro de una red SCADA con un PLC que almacena y controla el estado de las máquinas conectadas a éste.

El extremo con funciones de maestro (cliente) en la red es el software MX-AOPC UA Server, ofrecido por el propio fabricante de los PLCs, que se encuentra instalado en un equipo con sistema operativo Windows 8. La línea de comunicación Ethernet es una única línea bidireccional que comunica el equipo cliente con el equipo servidor maestro-esclavo.

En la Figura 4.1 puede observarse la infraestructura que dispone el laboratorio, así como los diferentes equipos que realizan las funciones de control sobre los autómatas (véase la Figura 4.2).

Figura 4.1: Laboratorio de investigación de CIRCE.

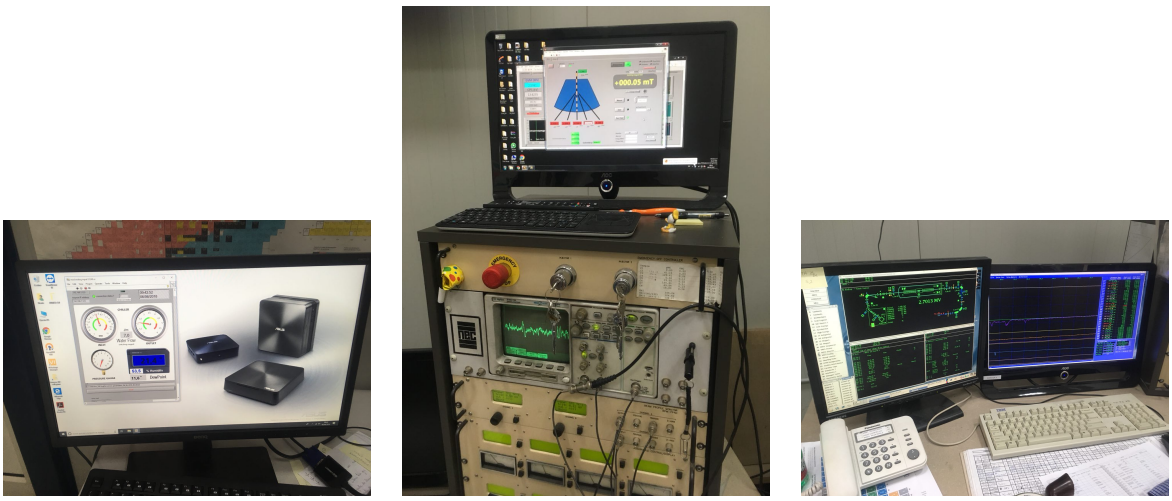


Figura 4.2: Equipos controladores en el laboratorio de CIRCE.

4.2. Entorno de experimentación

Como ya se ha mencionado anteriormente, para no interferir en el comportamiento del centro y no alterar el comportamiento de la red en cuanto al tráfico de datos y obteniendo información anómala, se ha realizado el despliegue de un sistema de monitorización que actúa de forma pasiva, es decir, únicamente lee y almacena la información que circula

por la red sin inyectar ningún tipo tráfico residual. Para ello, se ha hecho uso de las siguientes herramientas:

1. Throwing Star LAN Tap: También conocido como *network-tap*, es un dispositivo electrónico que dispone de 4 conexiones RJ-45. Este elemento se instala en el cableado físico de una red LAN, interceptando la comunicación y derivando el tráfico de entrada por una de las conexiones y el tráfico de salida por la otra. En la Figura 4.3 pueden observarse los componentes del dispositivo.

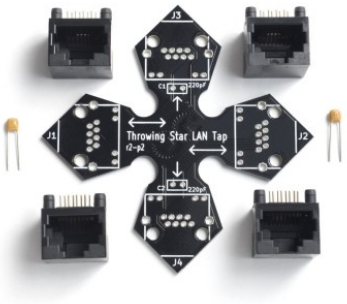
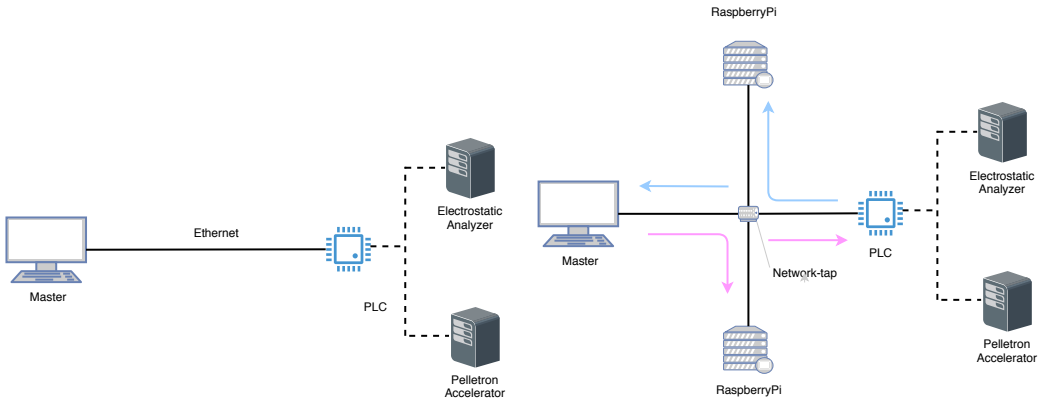


Figura 4.3: Throwing Star LAN Tap (extraído de [2]).

2. Dos dispositivos RaspberryPi 3B: Estos dispositivos disponen de un procesador Quad-Core Cortex A7 a 900MHZ, 1GB de RAM, sistema operativo Debian 9.3 y software de captura de tráfico de red tcpdump.

3. Cableado Ethernet categoría 5e

En la Figura 4.4(a) se representa la red del laboratorio donde un equipo master permanece directamente conectado a un PLC que dispone de varios dispositivos de entrada/salida. El montaje de monitorización se ha realizado introduciendo el network-tap en la conexión maestro-PLC. Al realizar esta división, el network-tap divide cada uno de los sentidos de la comunicación obteniendo en cada una de las RaspberryPi el tráfico de entrada y el de salida. En la Figura 4.4(b)se observa el esquema de la red con los elementos de monitorización, mientras que en la Figura 4.5 se muestra el montaje real en el laboratorio.



(a) Red original (b) Red con sistema de captura

Figura 4.4: Esquemas de la (a) red original y de la (b) red con el sistema de captura desplegado.

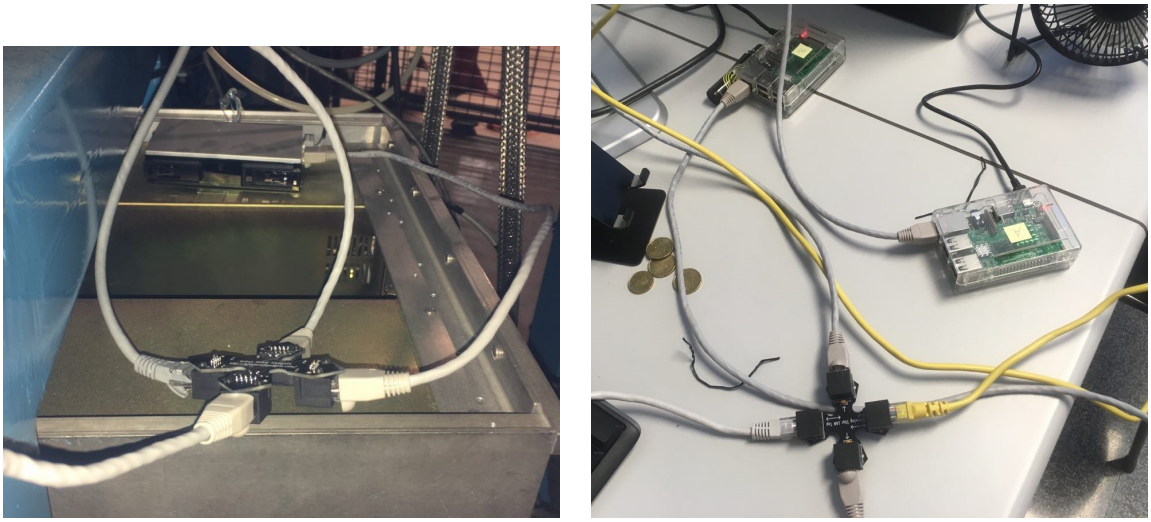


Figura 4.5: Sistema de monitorización de tráfico.

Cada una de las RaspberryPi captura el tráfico mediante la herramienta tcpdump, dando como resultado un fichero con formato PCAP con el tráfico de entrada y otro con el tráfico de salida. Para facilitar el posterior tratamiento de los datos, se ha procedido a la fusión de ambos ficheros.

Dado que el despliegue de este escenario de trabajo no afecta en el comportamiento de la red, la captura de datos se ha mantenido de forma ininterrumpida durante el transcurso de todo el proyecto. Como ya se ha mencionado anteriormente, el objetivo de este despliegue era obtener la mayor cantidad de datos del protocolo Modbus/TCP en un escenario real.

Tras un periodo de tiempo con el sistema de captura de tráfico en ejecución, se han obtenido diferentes parejas de ficheros de capturas. De forma excepcional, y con el permiso del laboratorio CIRCE, se ha procedido además a la inyección de paquetes de tráfico Modbus/TCP arbitrarios para capturar el comportamiento de la red ante todas las posibles situaciones en una comunicación. Para ello, se han desarrollado de diferentes scripts mediante Python que permiten la inyección de mensajes Modbus a una dirección IP y puerto determinados. Estos scripts se han liberado mediante licencia GNU/GPLv3, estando accesibles en `https://github.com/ibaimc24/modbus_traffic_generator` y explicándose en más detalle en la siguiente sección.

4.3. Análisis de capturas

Las capturas se han realizado durante el transcurso de seis meses (de Septiembre de 2017 a Mayo de 2018) de forma intermitente, obteniendo capturas de tráfico de 10 días de duración. Como resultado de la etapa de recolección de tráfico, se han obtenidos ficheros de entre 5 y 7 GB de datos. Esta cantidad de información plantea una dificultad añadida a la hora de manipular la información a analizar. Por ello, se han dividido las capturas en ficheros de menor tamaño para poder trabajar con la información de una manera adecuada. Una vez se ha dispuesto de suficiente cantidad de información sobre el tráfico de la red se ha procedido a su análisis. En concreto, se han cuantificado los siguientes extremos:

- Cantidad de tráfico del protocolo Modbus en la red.
- Equipos implicados en la comunicación del protocolo Modbus.
- Identificación de paquetes del propio protocolo Modbus.
  - Cantidad de paquetes.
  - Tiempos de respuesta.
  - Detección e identificación de ráfagas de mensajes.
  - Diferencias temporales de las transmisiones entre paquetes.
  - Captura de los datos utilizados en el protocolo.
- Defiinición e identificación de equipos no involucrados en la comunicación Modbus.
- Identificación del uso de otro tipo de protocolos por otros equipos de la red.

Dada la diversidad de información que se pretende analizar, se han desarrollado diferentes herramientas de análisis de las capturas mediante Python. A continuación, se describen las herramientas desarrolladas para el proceso de análisis, detallando el funcionamiento y finalidad.

Script 1: Inyección de tráfico. El primer script realizado permite la inyección de tráfico Modbus en la red. La herramienta permite diferentes configuraciones para la transmisión, siendo posible crear ráfagas de diferentes tipos de mensajes, modificar el retraso temporal en la transmisión de cada paquete de la ráfaga, determinar la cantidad de mensajes que se lanzan o realizar una transmisión de cada tipo de mensaje. La inyección de este tráfico tiene como objetivo conocer el comportamiento del dispositivo esclavo, y, por consiguiente, definir sus características, tomar estadísticas temporales y observar el impacto de la inyección de tráfico en el sistema.

Tal y como se ha explicado en el Capítulo 2, cada fabricante provee a los autómatas con una distribución de tablas de datos explicadas en las especificaciones del producto, pudiendo además ser definidas por los administradores en algunos casos. Con la ejecución de este script se pretende forzar a un dispositivo esclavo a responder con su configuración determinada.

Script 2: Segmentación de comunicaciones. El segundo script permite realizar una separación entre el tráfico Modbus/TCP de cualquier otro tráfico. Tomando como entrada los ficheros capturados, se identifican a través del protocolo TCP y mediante el puerto 502 (como origen o destino) los paquetes de tráfico Modbus. Como resultado se obtienen dos ficheros diferentes, uno con el tráfico de la comunicación Modbus y el otro con el tráfico residual. También se genera un archivo a modo de registro de ejecución donde se indica la proporción de tráfico (en términos porcentuales) que implica el contenido de cada uno de estos ficheros.

Script 3: Obtención de estadísticas. El tercer script permite el estudio de la comunicación Modbus/TCP. El script toma como entrada uno o varios ficheros de captura que contengan únicamente tráfico Modbus/TCP. La herramienta realiza una parametrización de este tráfico, obteniendo como resultado un registro de datos que indican la siguiente información:

- Identificación de dispositivos esclavo/maestro
- Identificación de mensajes Modbus y tamaños.
- Diferencias de tiempos entre dos mensajes Modbus iguales.

Script 4: Análisis de red. El último script permite realizar un análisis de la infraestructura de red donde se han tomado las muestras de tráfico. Mediante el tráfico residual (tráfico no Modbus/TCP) obtenido del script 2, esta herramienta realiza una búsqueda de información que permita la identificación de la infraestructura física de la red, así como los servicios disponibles. Esta información es de gran utilidad para un atacante dado que puede facilitarle la intrusión en la comunicación de control industrial bajo análisis al identificar los posibles objetivos de ataque. La información obtenida sobre cada equipo de la red es la siguiente: dirección IP, dirección MAC, puertos en un (<1024), protocolos más comunes usados (protocolo de resolución de nombres DNS, protocolo web HTTP o protocolo de mensajes de control de Internet ICMP, entre otros).

4.4. Resultados

Tras el desarrollo de estas herramientas y el procesado de las capturas se han obtenido como resultado, además de una estructura de ficheros de tráfico con un formato más legible y manejable, una serie de datos con lo que se ha considerado la información más relevante sobre cada bloque de información analizada.

4.4.1. Estadísticas de tipos de tráfico

En primer lugar, en las capturas se observa la existencia de diferentes tipos de tráfico. Este hecho indica la **ausencia de segmentación en las comunicaciones de la red**. Observando los resultados de cada una de las capturas por separado, puede verse en la Tabla 4.1 que la proporción de mensajes Modbus sobre el tráfico residual no es una variable constante. Esto implica que la problemática del impacto del tráfico residual sobre la comunicación Modbus sea de mayor complejidad: un aumento del tráfico residual en un corto periodo de tiempo podría afectar al sistema de comunicación industrial por una congestión en la red.

Fichero	Paquetes Modbus	Otros Paquetes
1.pcap	18152669	6378329
2.pcap	43132133	15884859
3.pcap	3267280	601016
4.pcap	6256732	2514383
5.pcap	38592123	21841579
6.pcap	0	1513816
7.pcap	0	46178221
8.pcap	284	16042342
9.pcap	14092837	16889363
10.pcap	3095374	5190806
11.pcap	26184430	20719495
TOTAL:	152773862 (49,84 %)	153754209 (50,16 %)

Tabla 4.1: Estadísticas por cada pareja de ficheros de captura.

Además, **la no segmentación de las comunicaciones no sólo facilita posibles ataques como la denegación del servicio sino, que también dificulta la detección de los ataques** por el aumento de tráfico y generación de información por el conjunto de equipos.

Como se muestra en la Tabla 4.1, alrededor de la mitad del tráfico no pertenece al protocolo Modbus. Teniendo en cuenta que la eficiencia temporal en estas redes es prioritaria, una congestión en la infraestructura de la red por otros tipos de tráfico podría degradar el funcionamiento del sistema. Por tanto, liberar a la red de esta carga de tráfico, evitaría posibles picos de tráfico (y la consecuente congestión en la infraestruc-

	Dirección MAC	Dirección IP	Protocolos	Puertos en uso
Equipo 1	00:08:9b:cb:b8:0e	192.168.1.111	ICMP,TLS,HTTP,DNS	138,137
Equipo 2	00:0a:f7:68:48:73	192.168.1.147	TLS,ICMP,DNS	68,137
Equipo 3	00:12:f0:99:63:71	192.168.0.106	LLC,ICMP,DNS,HTTP	138,137,123

Tabla 4.2: Fragmento de fichero de registro del análisis de tráfico de red.

tura) que puedan empeorar el funcionamiento del servicio. Además, el uso de diferentes protocolos en los mismos equipos involucrados en la comunicación Modbus aumenta las probabilidades de una intrusión en los equipos, ya que, a mayor cantidad de servicios en red activos, mayor probabilidad de contener vulnerabilidades que permitan la intrusión en el sistema.

4.4.2. Información de red

Como ya se ha descrito anteriormente, la no segmentación de servicios no involucrados en el proceso control industrial puede permitir a personas no autorizadas el acceso a información en caso de la vulneración de la red, además de facilitar la propagación de acciones maliciosas en los equipos de la red. A modo ilustrativo, en la Tabla 4.2 se muestra un extracto de la información recopilada sobre el resto de dispositivos que hacen uso de la red y por tanto pueden realizar una comunicación directa con los equipos involucrados en la comunicación Modbus o incluso con el propio dispositivo Modbus.

Como resultado, cabe mencionar que **se han encontrado alrededor de unos 70 equipos diferentes en la red**. Aunque se hayan contabilizado direcciones en broadcast, unicast, y direcciones multicast utilizadas en IPv6, cualquiera de estas identificaciones pone de nuevo de manifiesto la gran problemática ya mencionada anteriormente.

4.4.3. Estadísticas de características del tráfico Modbus

Debido a circunstancias del escenario de trabajo y la configuración realizada para la obtención de las capturas, se ha detectado un error en las referencias temporales de los paquetes, de modo que las estadísticas obtenidas a este respecto no tienen la fiabilidad esperada. Aún así, y conociendo este problema, se explica de forma detallada las conclusiones obtenidas:

- El tráfico se limita a peticiones con las funciones 1 (*ReadCoils*), 2 (*ReadDiscreteInputs*) y 15 (*WriteMultipleCoils*); siendo esta última la menos habitual.
- Las ráfagas constan de paquetes consecutivos con los códigos de función 1 y 2. En algunas de las ráfagas también se incluyen paquetes con el código 15.
- La mayoría de los paquetes son de la misma longitud (10, 12 ó 14 bytes). Por consiguiente, se puede aproximar la cantidad de datos que se piden en dichas transacciones.

- Pese a la utilidad que provee el protocolo de la secuenciación de la comunicación Modbus, el cliente (dispositivo maestro) **no hace uso del campo que permite la identificación del mensaje**. Este campo es leído y copiado al paquete de respuesta por parte del esclavo, facilitando el emparejamiento de pregunta-respuesta de la comunicación. En este caso, se ha observado que el dispositivo maestro siempre coloca el valor 0 en este campo. Este hecho permite que tanto el equipo servidor (PLC) como el equipo cliente (dispositivo maestro) puedan ser objetivos de un ataque de *replay* (basado en la captura y retranmisión de uno o varios mensajes). Puesto que el protocolo Modbus/TCP está implementado sobre un protocolo no seguro como TCP, este tipo de ataques pueden realizarse tanto a nivel de transporte como a nivel de aplicación.

Los mensajes del Modbus tienen un tamaño variable, lo cual permite al extremo cliente de la comunicación realizar consultas de diferentes tamaños, y por consiguiente, la petición de modificación o acceso a una cantidad variable de datos. Algunos de estos mensajes son los identificados por las funciones 1, 2, 3 (*ReadHoldingRegisters*), 4 (*ReadInputRegisters*), 15 y 16 (*WriteMultipleRegisters*). Aunque cada uno realice operaciones sobre tipos de datos diferentes, recuérdese que pueden realizarse sobre una cantidad variable de los mismos (véase la Seción 2.2). Dependiendo de la funcionalidad y configuración de los PLC, puede ser necesario el acceso a una cantidad datos mayor para acceder al dato completo, como pueden ser números decimales, o de longitud mayor a 16 bits.

Analizando el tamaño de los paquetes de tráfico Modbus y teniendo en cuenta el tamaño máximo que tiene el segmento de datos por las limitaciones del protocolo, es posible detectar los tipos de datos que se están almacenando, qué cantidad de datos diferentes se están manejando, o qué estructura tienen las tablas de datos de Modbus. Este tipo de análisis pone en riesgo la confidencialidad de los datos mantenidos por el PLC. Por tanto, un atacante puede usar técnicas de inyección de mensajes con el objetivo de conocer la configuración del autómata y su consiguiente mecanismo de funcionamiento. A continuación, se describe el experimento realizado a este respecto como prueba de concepto.

4.4.4. Resultados inyección de mensajes

La ejecución del script de inyección de tráfico Modbus (script 1) ha dado como resultado diferentes respuestas que pueden resumirse con las siguientes conclusiones:

1. El esclavo permite conexiones múltiples de diferentes dispositivos maestros en paralelo, respondiendo a todos los clientes de forma simultánea.

2. La inyección de grandes cantidades de tráfico afecta significativamente al tiempo de respuesta del esclavo a cada uno de los mensajes recibidos.

3. Es posible realizar una inyección de tráfico obteniendo como resultado conocimiento de la estructura y el contenido de las tablas de datos Modbus del dispositivo esclavo, sin alterar el estado del sistema.

La Figura 4.6 muestra la configuración de dichas tablas en el sistema real analizado, obtenida mediante un acceso proporcionado por el propio dispositivo esclavo a través de una página web convencional. El Listado 4.1 muestra la estructura de datos del esclavo obtenida tras el proceso de inyección de tráfico. Como se puede observar, esta información coincide con la proporcionada por administradores del laboratorio mostrada en la Figura 4.6.

No.	Description	User-defined Start Address (DEC)	Function Code	Read/Write	Reference Address (DEC)	Total Channels	Data Type
1	DO (Relay) Value	0000	01:COIL STATUS	RW	00001	6	1 bit
2	DO (Relay) Pulse Status	0016	01:COIL STATUS	RW	00017	6	1 bit
3	DO (Relay) Value All Channel (Ch0-Ch5)	0032	03:HOLDING REGISTER	RW	40033	1	1 WORD
4	DI Value	0000	02:INPUT STATUS	R	10001	6	1 bit
5	DI Counter Value (Double Word)	0016	04:INPUT REGISTER	R	30017	6	2 WORD
6	DI Value All Channel (Ch0-Ch5)	0048	04:INPUT REGISTER	R	30049	1	1 WORD
7	DI Counter Start/Stop	0256	01:COIL STATUS	RW	00257	6	1 bit
8	DI Counter Clear	0272	01:COIL STATUS	RW	00273	6	1 bit
9	P2P Connect Status	4096	01:COIL STATUS	RW	04097	6	1 bit
10	P2P Output Safe Status	4112	01:COIL STATUS	RW	04113	6	1 bit
11	Clear P2P Output Safe Status	4128	01:COIL STATUS	RW	04129	6	1 bit

Figura 4.6: Tabla de datos Modbus del sistema real de estudio.

```
SUPPORTED FUNCTION CODES
        [1, 2, 3, 4, 5, 6, 8, 15, 16]

        COILS
                From 0 to 5
                From 16 to 21
                From 256 to 261
                From 272 to 277
                From 4096 to 4101
                From 4112 to 4117
                From 4128 to 4133

        DISCRETE INPUTS
                From 0 to 5

        HOLDING REGISTERS
                From 32 to 32
```

```
INPUT REGISTERS
        From 16 to 27
        From 48 to 48
```

Listado 4.1: Tabla de datos Modbus obtenida del PLC tras la inyección de mensajes.

Capítulo 5

Simulador del protocolo Modbus/TCP

El protocolo Modbus/TCP presenta una considerable cantidad de vulnerabilidades de seguridad [13] [15] como la vulneración de la confidencialidad de los datos, la integridad, la ausencia de autenticación y la falta de garantia de la disponibilidad del servicio. Además, como se ha demostrado de forma práctica en un escenario real en el Capítulo 4, una inyección simple de tráfico permite conocer la configuración del PLC. Con el objetivo de evaluar las posibles soluciones a las vulnerabilidades presentadas sin que afecte al funcionamiento del sistema real se ha desarrollado un sistema software a modo de simulador que actúa de forma equivalente a un escenario real. Este simulador sirve como plataforma para pruebas y desarrollo de posibles soluciones de seguridad (u otras mejoras) del protocolo Modbus/TCP. En primer lugar, se describen los requisitos funcionales y no funcionales del simulador. Después, se hace una descripción del diseño, estructura y funcionamiento de cada elemento del sistema de simulación.

5.1. Requisitos funcionales y no funcionales

Tras el estudio de herramientas similares en el Capítulo 3 y de la identificación de los parámetros de tráfico de mayor importancia en el Capítulo 4, se han definido una serie de requisitos funcionales y no funcionales que los simuladores deben cumplir.

Requisitos funcionales

1. El dispositivo cliente del sistema debe permitir al usuario la configuración del soporte de diferentes funciones Modbus.

2. El dispositivo cliente debe de soportar el servicio de forma concurrente. La estructura y contenido de las tablas de datos Modbus deberá ser única, independientemente de la cantidad de dispositivos maestros en la red.

3. El dispositivo maestro deberá permitir por parte del usuario la configuración de ráfagas de mensajes Modbus, donde se podrán definir los siguientes campos:
   - Cantidad y orden de tipos de mensajes Modbus.
   - Diferencia de tiempo entre la transmisión de mensajes sucesivos.
   - Una varianza que permita aleatorizar el tiempo entre las transmisiones.
4. El dispositivo maestro deberá permitir la configuración de diferentes tipos de mensajes Modbus.
5. Cada uno de los mensajes Modbus que transmita el dispositivo maestro podrá ser configurado de forma completa permitiendo la definición por parte del usuario de todos los campos del mensaje.

Requisitos no funcionales

1. Para que el presente proyecto pueda continuarse en trabajos posteriores, el desarrollo se realizará con herramientas de licencia pública GNU/GPLv3.
2. Se introducirán capas de abstracción para facilitar la comprensión y las futuras mejoras que se vayan a realizar.
3. Se realizará una herramienta escalable y legible mediante comentarios y documentación sobre el código realizado. Dada la envergadura del proyecto y las necesidades a cubrir, es necesario facilitar los trabajos posteriores, además de facilitar la integración de soluciones en el sistema desarrollado.
4. Puesto que se ha planteado la simulación de una comunicación cliente-servidor, se ha considerado interesante la opción de crear ambos extremos de forma totalmente desacoplada por las ventajas que esto supone, así como la integración de uno de los extremos en entornos reales, la posibilidad de validar las herramientas con herramientas externas o realizar un desarrollo paralelo de ambos extremos sin la necesidad de una integración mutua.

5.2. Simulador de dispositivo esclavo Modbus

La complejidad de un dispositivo esclavo en una comunicación Modbus se centra en la mecánica y operatividad de ésta, controlada tanto por los dispositivos remotos como por los dispositivos maestros. Dado que el proyecto está centrado en la comunicación de la red, se ha creado un sistema de simulación de esclavo Modbus donde la información involucrada en el protocolo es únicamente almacenada sin desencadenar ningún tipo de acción.

5.2.1. Dependencias software

Como lenguaje de programación, se ha escogido Python por ser un lenguaje interpretado, que facilita su uso multiplataforma. Para la gestión de las dependencias, facilitar la instalación, exportación e importación de las librerías de terceros y reducir el tiempo de despliegue del servicio, se ha hecho uso del gestor de paquetes y entornos Conda.

Una de las librerías a mencionar es Scapy [7], que facilita la manipulación y uso de paquetes y protocolos de red para su desarrollo o estudio de protocolos. Además, por ser una herramienta muy utilizada en el ámbito de la telemática y de licencia pública, tiene una gran comunidad de desarrolladores en activo y documentación que ayuda a comprender su uso y funcionamiento.

5.2.2. Estructura del simulador

El programa de simulación se ha estructurado en tres partes diferenciadas: una parte que gestiona el contenido y el acceso a los datos simulados, un servicio de conexión y comunicación que implementa el protocolo Modbus, y un proceso que simula la mecánica conectada al PLC. Dado que los datos tienen una relación directa tanto con la mecánica del autómata como de la comunicación con el cliente, existe un proceso concurrente entre todas las partes.

Aspectos del direccionamiento lógico

La memoria interna del controlador es utilizada para el control de los dispositivos que tenga conectados (la relación entre el direccionamiento físico y lógico depende del fabricante), pero el protocolo fija una limitación de 65535 direcciones. Esto se debe a la limitación de 253 bytes de datos que permite transmitir el protocolo en la implementación para líneas de comunicación serie [11], según el siguiente cálculo:

Modbus **PDU** for serial line communication: 256 – Server Address (1 byte) – CRC (2 bytes) = **253 bytes**
TCP Modbus **ADU**: 253 bytes (**PDU**) + MBAP Header (7 bytes) = 260 bytes

En la memoria del dispositivo pueden almacenarse, como ya se ha mencionado anteriormente (véase la Sección 2.2), varios tipos de datos donde su longitud es de 1 ó 16 bits. A pesar de existir diferencias entre diferentes marcas y modelos de controladores, por lo general estos datos son almacenados en tablas lógicas, haciendo un reparto equitativo en su direccionamiento como. De este modo, cada tabla esta direccionada de 0 a 10.000 (0000 - 270E). No obstante, el protocolo permite el direccionamiento y por tanto la asignación en el controlador de tablas es de hasta 65.535 direcciones (0000 - FFFF).

El simulador consta de una clase por cada uno de los tipos de datos que gestiona el acceso del sistema a los datos almacenados en las tablas. Cada clase consta de una matriz de números en binario del tamaño de la tabla, que actúa a modo de máscara. Cuando se recibe una solicitud de acceso a una sección de la tabla, se devuelve un valor booleano indicando si el acceso está permitido o no, dependiendo si todos los bits de las

direcciones que se solicitan están habilitados (es decir, con valor 1). Una vez el sistema tiene el permiso de la correspondiente máscara de la operación, se procede a la acción solicitada en la tabla de datos. La tabla de datos es una matriz de números binarios que representan el contenido de los datos del controlador. Puesto que en la simulación no se considera importante el contenido de los datos, se inicializan a cero. El acceso a las direcciones de la tabla de datos se creada en el proceso de configuración del simulador.

En un escenario real, los dispositivos conectados al PLC y el direccionamiento lógico de las tablas Modbus a las que acceder para la consulta del estado de estos dispositivos suelen ser parámetros definidos por los fabricantes y por tanto son configuraciones estáticas que un usuario no suele poder modificar. El contenido de las tablas depende parcialmente de la mecánica del autómata. Puesto que el simulador se comporta de forma genérica con objetivos centrados en el protocolo de comunicación Modbus, independientemente del contenido de las tablas del PLC, la importancia de estos parámetros queda por tanto en un segundo plano. No obstante, puesto que el direccionamiento de las tablas es una parte imprescindible del proceso de la comunicación, se ha creado un proceso de configuración que permite al usuario definir bloques de las tablas donde el protocolo tendrá acceso y con qué operaciones podrá hacerlo.

5.2.3. Funcionamiento

El simulador del dispositivo esclavo requiere primero de la generación de la configuración por parte del usuario, de los parámetros de red, de secciones de datos y de parámetros específicos del protocolo. Una vez creada la configuración, con opción a guardar o cargar dicha configuración en fichero, el simulador se inicia en un estado de escucha.

De forma similar a un autómata real, el simulador acepta una cantidad definida de conexiones TCP, que serán procesadas como comunicaciones maestro-esclavo Modbus. En el momento en el que sea establece la conexión TCP, el servicio permanece a la escucha de paquetes Modbus para procesarlos y responderlos de forma individual. El comportamiento ante la recepción de diferentes funciones Modbus depende tanto de la configuración previamente creada por el usuario como del tipo de paquete recibido. Así pues, un usuario puede definir las funciones que el esclavo va a procesar. Para las funciones no definidas por el usuario, el simulador devolverá un paquete de excepción Modbus indicando la no disponibilidad de la función solicitada. Para cada una de las funciones aceptadas, el software sigue una secuencia de operaciones diferente.

Como ejemplo, se ilustra en la Figura 5.1 el proceso que sigue el simulador en el caso de recibir una función Modbus 0x01 *ReadCoil*. Como puede observarse en el diagrama de secuencia, el programa está provisto de dos elementos que cobran importancia a la hora desempeñar las siguientes funciones:

Tabla de datos: el elemento identificado en el diagrama como *DataBlock* es una clase que contiene una serie de números enteros que representan el contenido de las tablas Modbus. Las diferentes tablas que almacenan cada uno de los tipos de datos Modbus, es identificado por un objeto de esta clase. El tamaño de estas tablas es configurable por el usuario.

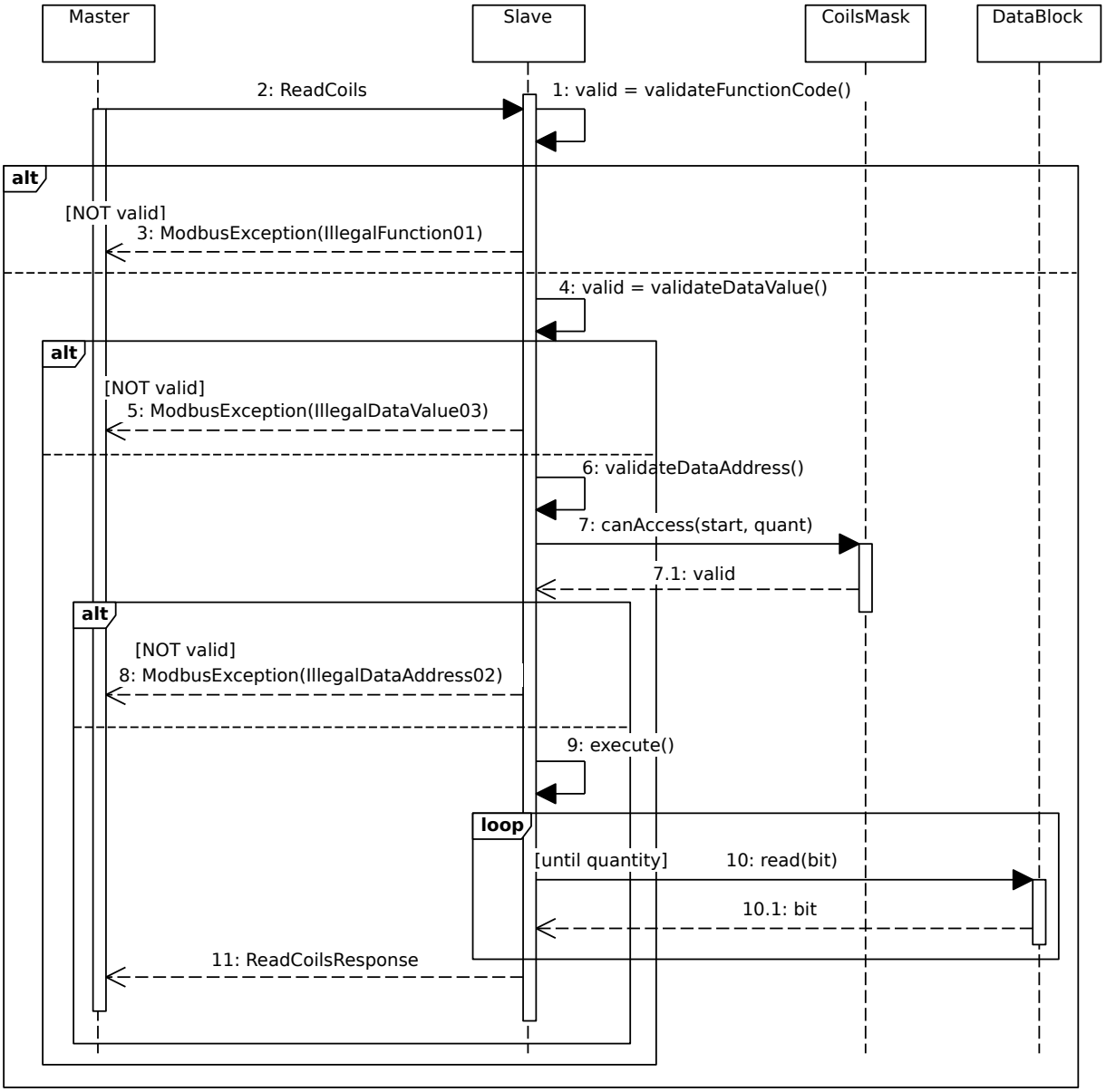


Figura 5.1: Diagrama de secuencia de mensaje *Read Coils*.

Máscara de acceso: identificado en el diagrama como *CoilsMask*, representa una clase que administra los permisos de acceso a las tablas de datos del protocolo Modbus. Dentro de la estructura de las tablas de datos, se definen las direcciones donde los datos serán almacenados y gestionados por el autómata. De este modo, todos los mensajes que referencien direcciones de la tabla fuera de este rango predefinido serán respondidos con un mensaje de error. La clase de máscara define las direcciones de la tabla que son accesibles. Durante la configuración del programa, el usuario deberá identificar en que direcciones de la tabla se almacenarán los datos.

Para comprobar la viabilidad de una operación y generar la respuesta correcta en ese caso concreto, se sigue el diagrama de estados descrito en la Figura 5.2, conforme a las especificaciones del protocolo [12]. Primeramente, se comprueba que la funcionalidad requerida en el mensaje es soportada. En el siguiente estado se comprueba la que la cantidad de datos solicitada sigue las especificaciones del protocolo. Por último, se realiza la comprobación del acceso a la tabla de datos. Este estado comprueba que la solicitud no se realiza sobre direcciones fuera de los rangos posibles. Un resultado negativo en cualquiera de estos procesos dará como resultado una respuesta de excepción por parte del simulador, en caso contrario, se procede a la ejecución de la función solicitada y la consecuente respuesta.

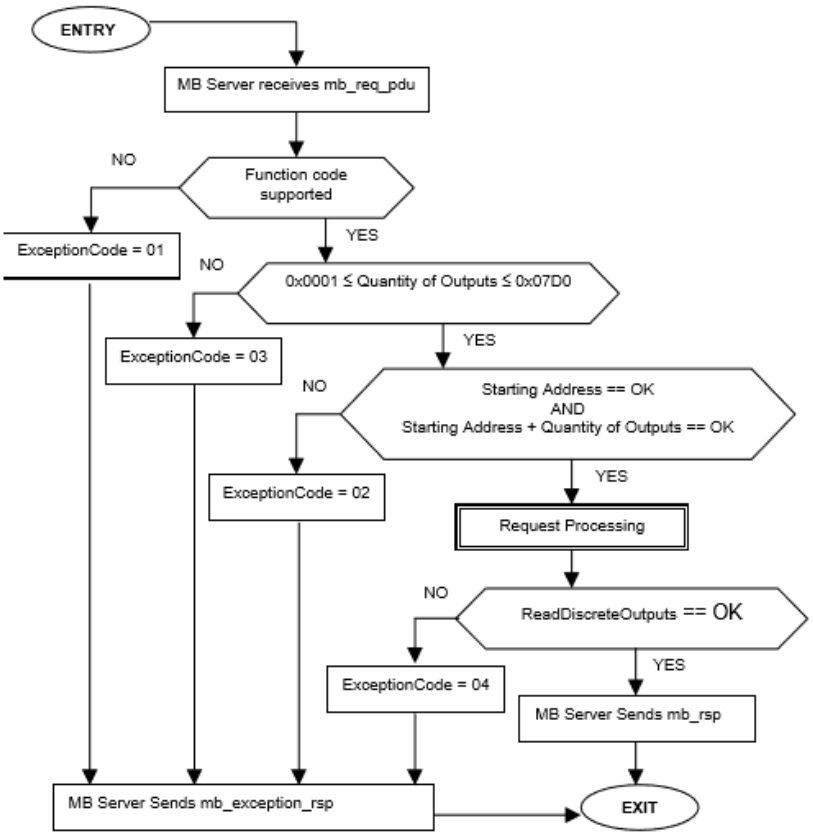


Figura 5.2: Diagrama de estados para el procesado de mensajes tipo *Read Coils* (extraído de [12]).

De forma análoga, el simulador sigue una secuencia de comprobación de parámetros los cuales generan una excepción y su correspondiente respuesta de error Modbus si no cumple estas validaciones. Finalmente, se procede a la lectura o escritura de datos en la clase correspondiente.

5.3. Simulador de maestro Modbus

Generalmente, en una red Modbus real el controlador que actúa a modo de maestro suele ser un software propietario del fabricante de PLC. Estos programas disponen de diversas funcionalidades que varían con el tipo de autómata adquirido. Obviando las funcionalidades de alto nivel que se engloban en el funcionamiento de los autómatas, se ha pretendido crear exclusivamente la simulación de la comunicación de red con el autómata. Por ello, se han ignorado las acciones tras la recepción de los paquetes enviados por el esclavo, prestando atención únicamente a las transmisiones.

A continuación, se detallan las características más relevantes que engloban el simulador de dispositivo esclavo Modbus.

5.3.1. Dependencias software

Al igual que el simulador de esclavo, se ha usado Python versión 3 con el gestor de paquetes Conda y la librería Scapy. Se ha tratado de realizar un servicio de fácil despliegue y con funcionalidades básicas en línea de comandos. Dado que será un programa pensado para utilizarse en entornos de desarrollo como en entornos de producción, durante el transcurso del proyecto se ha planteado como requisito no funcional crear una interfaz en línea de comandos. La dependencia de interfaces gráficas, aunque pudiera facilitar el uso y la comprensión del software, limitaría su despliegue en entornos y plataformas de acceso en remoto.

5.3.2. Estructura del simulador

El programa puede dividirse en dos partes claramente diferenciadas, según su funcionalidad:

Configuración: El maestro realizará la comunicación en función de una serie de parámetros definidos por el usuario. Algunos de los parámetros pueden tomarse por defecto o ignorarse, permitiendo una configuración más sencilla. Para un funcionamiento lo más real posible, se han considerado valores como el tiempo entre transmisiones de paquetes incluido un parámetro de aleatoriedad, y ráfagas de diferentes tipos de paquetes. Así, el usuario puede indicar qué tipos de paquetes se generarán, sus campos, y con qué frecuencia. Una vez definida toda la configuración, durante el transcurso de la comunicación el simulador hará uso estos parámetros para realizar las transmisiones acordes con lo indicado.

Comunicación: Como ya se indicado previamente, dado que no se va a prestar importancia al contenido de los datos de respuesta que transmite el esclavo, la funcionalidad del maestro que maneja la conexión y transmisiones realiza únicamente la construcción de los paquetes Modbus según la estructura que se indica en la configuración y los transmite a través de la comunicación TCP creada. Como módulo adicional, se ha creado un sistema de creación de registros que ofrece al usuario un control sobre el estado de la comunicación o una ayuda visual de los eventos de ejecuciones anteriores.

El sistema de registros permite además una forma sencilla de realizar comprobaciones y validaciones del código para desarrollos futuros, ya que es posible redireccionar los textos mostrados por pantalla durante la ejecución a los ficheros de registro de forma sencilla.

5.3.3. Funcionamiento

De forma similar al simulador del dispositivo esclavo, el dispositivo maestro se ha desarrollado pensado para realizar una configuración sencilla de los paramentros más relevantes del protocolo. Así, mediante una interfaz de línea de comandos el usuario tiene que introducir una serie de parámetros de configuración. Una vez terminada la configuración, se permite la opción de exportar o importar la configuración creada a un fichero. Finalmente, el simulador de maestro iniciará la comunicación una vez introducido el comando de inicio. Durante la comunicación, el simulador del maestro realiza las transmisiones de los mensajes Modbus según los parámetros definidos en la configuración.

Capítulo 6

Reproducción del entorno real y validaciones

Una vez desarrollados los simuladores de dispositivos esclavo/maestro del protocolo Modbus/TCP, se ha realizado proceso de validación y una reproducción del entorno real analizado en la fase captura de datos (véase el Capítulo 4). Esta reproducción ha permitido el estudio de la seguridad del sistema, el análisis de las vulnerabilidades del protocolo y la reproducción de posibles ataques, así como plantear las soluciones a estos ataques. Por último, cabe destacar que la reproducción de un entorno real permite desarrollar soluciones de detección de intrusiones pasivas en la red.

6.1. Validación de los simuladores

Como en todo desarrollo de software, una de las partes más importantes del proyecto es la validación del producto. En este caso, al haber realizado dos extremos de una comunicación y estar fuertemente desacopladas, se ha procedido a realizar una validación para cada uno de los simuladores por separado.

6.1.1. Validación esclavo

Gracias al tráfico capturado en el entorno real, ha sido posible la comparativa del comportamiento del simulador del esclavo ante la misma ráfaga de paquetes Modbus. En este caso, dado que se pretende simular el mismo escenario de trabajo analizado, se han introducido los parámetros de configuración obtenidos durante el proceso de análisis del escenario real.

Para la validación, se procederá a la ejecución del simulador en una máquina virtual Windows 10 ejecutándose sobre VirtualBox con el interfaz de red configurado en modo adaptador puente. El objetivo es introducir el escenario de simulación en una red LAN convencional de forma análoga a la que se encuentra el escenario real. Pese a que el tráfico externo no afecta de ninguna manera al tráfico Modbus, si que puede afectar al tiempo de transmisión y procesado del tráfico analizado.

Una vez introducida la configuración e iniciado el proceso, se realizan dos primeras validaciones:

- Lanzar el script de inyección de trafico Modbus que se creó para la captura de tráfico en el escenario real.
- Inyectar tráfico específico que permita fuerce la respuesta de los valores introducidos en la configuración.

En esta parte, dado que se dispone del software que provee el fabricante del PLC, se ha hecho uso de herramientas de terceros desarrolladas con un objetivo similar al de este proyecto con intención de validar el simulador con mayor precisión.

Lectura y escritura en memoria

Dado que la funcionalidad más básica de un esclavo es la opción de lectura y escritura en los datos de memoria mediante el uso de mensajes Modbus, se ha procedido realizar mediante diferentes herramientas la comprobación de este proceso en el simulador.

Primeramente, se ha hecho uso de qModMaster [14], una herramienta escrita en C++ con interfaz gráfica que permite realizar transmisiones básicas de mensajes Modbus. Existe un grado de acoplamiento de este software con el comportamiento del cliente, por lo que no es posible interpretar con el propio cliente, el contenido de las respuestas del simulador. No obstante, realizando una captura mediante Wireshark, puede observarse el proceso pregunta-respuesta de la comunicación. Para la validación, se ha configurado el esclavo con una tabla de 'Coils' con datos en el rango de direcciones 0-40, y se ha desactivado el realizar cambios en los datos. Se ha hecho una lectura de los datos de las direcciones 20-34 (15 bits). Tras comprobar el estado de todos los bits a '0', se ha realizado una escritura a '1' del bit 25. Para comprobar el correcto funcionamiento del programa, se vuelto a realizar la lectura de los bits 20-34 y observar el bit 25 con el valor '1'.

Este mismo proceso de validación se ha realizado con cada una de las funciones de lectura/escritura *(Read/Write Coils, Read/Write Single Register, Write Multiple Coils/-Register)*.

Simulación real 1

Para comprobar que simulador se comporta de forma satisfactoria ante una comunicación similar a la obtenida en el análisis del escenario real, se ha realizado un script que crea una ráfaga de peticiones Modbus con una latencia específica. En este caso, se ha introducido una latencia inferior a la media obtenida en el escenario real para observar el grado de eficiencia temporal que se obtiene con el simulador.

Los parámetros considerados para el script han sido:

- Ráfagas de funciones 0x01 y 0x02 de forma alterna.
- Campos de los paquetes (los mismos para ambos)

- StartAddr: 0
- Quantity: 6

Respecto a la configuración del simulador, se han creado las mismas tablas que contiene el PLC real pese a que no interfieran en la actual comunicación. Analizando los resultados se puede observar un comportamiento similar al real.

6.1.2. Validación del maestro

Una de las problemáticas del estudio de redes con tráfico Modbus y de las razones del desarrollo de este proyecto es la falta entornos reales donde realizar análisis y validaciones mediante el uso de PLCs. Por esta razón, para validar el comportamiento del simulador del maestro ha sido necesario el previo desarrollo del simulador del esclavo. Una vez validado de comportamiento del esclavo, se ha procedido al desarrollo y validación del maestro. Como se ha podido observar en el análisis del comportamiento del protocolo en el entorno real, el Maestro realiza ráfagas continuadas de peticiones de lectura y escritura con un breve delay entre cada transmisión. Las secuencias observadas son las siguientes:

El maestro está diseñado para un funcionamiento que pueda ser definido por el usuario, por lo tanto, se ha procedido primeramente a la validación de la transmisión de cada uno de los tipos de mensajes de forma individual. Continuando con el caso anterior, y habiendo comprobado el correcto funcionamiento del esclavo, se ha creado una configuración donde el maestro realice peticiones de forma continua con una diferencia de 0.01248 (valor estimado a partir del estudio realizado en el Capítulo 4) segundos entre cada transmisión, con los siguientes campos en los mensajes:

Function Code: 1 (Read Coils)

Starting Address: 0

Quantity: 6

Se ha repetido la ejecución con cada uno de los posibles mensajes, con la siguiente configuración:

Function Code: 2 (Read Discrete Inputs)

Starting Address: 0

Quantity 6

Function Code: 3 (Read Holding Registers)

Starting Address: 32

Quantity: 1

Function Code: 4 (Read Input Registers)

Starting Address: 6

Quantity: 2

Tras comprobar que la comunicación es la esperada, se asume el correcto funcionamiento del simulador para la transmisión de estos paquetes. Por lo tanto, pese a la necesidad de hacer uso del simulador del esclavo previamente desarrollado, dado que el comportamiento es equivalente al PLC analizado en el entorno real, se asume que el simulador del maestro podrá integrarse completamente en la red del escenario real.

Para la validación del resto de funciones Modbus ha hecho uso de la librería de test 'mock' [10], que permite la opción de la simulación de datos haciendo la validación totalmente independiente de un equipo servidor en la comunicación.

6.2. Descripción del escenario de simulación

Pese a no ser la configuración recomendada para redes de control industrial, se ha decidido realizar la configuración en una red abierta, con varios equipos, y sin segmentación de las comunicaciones de la propia red. Recuérdese que la configuración recomendada en este tipo de redes es un aislamiento de las comunicaciones, permitiendo únicamente a los equipos involucrados en el sistema de adquisición de datos mantener comunicaciones activas, limitándose estas a las que sean estrictamente necesarias. Para la simulación, se ha hecho uso de la herramienta de virtualización VirtualBox versión 5.2.12, creando 3 máquinas virtuales:

Máquina 1

En la fase de análisis del escenario se ha detectado el uso de protocolo HTTP en el equipo servidor de la comunicación Modbus/TCP. Por lo tanto, es posible que el equipo sea un *gateway* que permite la conversión de Modbus/TCP a Modbus/RTU. Dado que la actividad en este equipo puede limitarse al uso del protocolo Modbus y HTTP, se ha creado una máquina virtual con Ubuntu Server 16.04. Como propiedades de la máquina, se modificado la dirección MAC a la que se dispone en el entorno real: 00:90:e8:36:77:b2. Aunque la memoria interna de un PLC sea mucho menor, por comodidad, y por la necesidad integrar un sistema operativo, se han asignado 2GB de memoria RAM. La red se ha configurado en modo adaptador-puente con direccionamiento IP estático.

Máquina 2

A modo de controlador (dispositivo maestro), se ha creado una virtual con sistema operativo Windows 10 Education. En este caso, la actividad del equipo en el entorno real es más notoria en el sentido que se ha detectado el uso de más protocolos de comunicaciones como DNS, ICMP, o TLS. Además,existen servicios activos en los puertos 137 y 138 que pueden ofrecer servicios como el protocolo NetBIOS sobre TCP. Igual que en el caso de la máquina 1, la dirección MAC se ha modificado para tener la misma que el

equipo del escenario real: 00:22:15:48:2e:e7. Se han asignado 4GB de memoria RAM, y una configuración de red de adaptador puente con direccionamiento dinámico.

Máquina 3

En el escenario analizado se han detectado una seria de equipos con diferentes propiedades en cuanto a nivel físico, nivel de enlace, de red y transporte. En este caso, como se pretende realizar un estudio sobre la seguridad de la red, se ha creado una máquina virtual que pretende realizar funciones maliciosas. Aunque no reproduzca las propiedades de ninguna máquina del escenario real, permite analizar los posibles vectores de ataque de la comunicación que puedan aprovecharse desde cualquier dispositivo de la red que haya sido comprometido. La máquina se ha creado con un sistema operativo Kali Linux 17.2 con 4 GB de RAM, y adaptador puente con direccionamiento dinámico.

6.3. Reproducción de intrusiones

Según estudios previos [13] [15], es posibles realizar intrusiones en el protocolo Modbus pudiendo vulnerar cualquiera de los principios básicos de la seguridad (autenticación, integridad y confidencialidad). A continuación, se describe la metodología y resultados de la realización de diferentes ataques que permiten la vulneración de los mencionados principios de la seguridad de la información.

6.3.1. Descubrimiento de funciones

Para este ataque se han usado dos herramientas diferentes: primero, el script desarrollado para la inyección de tráfico, que permite detectar la configuración de un dispositivo esclavo Modbus; y segundo, la herramienta qModMaster [14], descrita en el Capítulo 3. Para ello, se han iniciado las maquinas 1 y 2, reproduciendo el escenario del laboratorio previamente analizado. A través de la máquina 3 se ha inyectado tráfico destinado a la máquina 1. Para simplificar el estudio, la captura del tráfico se ha realizado mediante tcpdump en la máquina 1 (dispositivo esclavo).

De forma idéntica a la ejecución en el escenario real, se ha obtenido la estructura completa de los datos Modbus que contiene el esclavo.

6.3.2. Vulneración de la confidencialidad

Dado que la comunicación se realiza sobre un protocolo no seguro, es posible la intercepción y la modificación de los mensajes Modbus que se transmiten entre cliente-servidor y servidor-cliente, tal y como se demuestra en el estudio [13]. Mediante la técnica de envenenamiento ARP, se ha realizado un ataque con el fin de demostrar de forma práctica la posibilidad de vulnerar el principio de confidencialidad. El ataque se ha realizado mediante la herramienta Ettercap, disponible en cualquier versión de la distribución KaliLinux. Una vez realizada la configuración del ataque, se ha realizado una captura en la máquina atacante mediante Wireshark, donde se ha realizado una visualización del

tráfico Modbus/TCP. La falta de cifrado y autenticación en el protocolo permite este tipo de ataques así como la modificación de los datos y ataques de replay.

6.3.3. Denegación del servicio

Como resultado de la posibilidad de realizar inyección de trafico Modbus en la red y la capacidad del esclavo de soportar varias conexiones en paralelo, es posible realizar un ataque que fuerce al PLC a entrar en un estado no deseado de forma continuada. Las tablas de datos Modbus con permisos de lectura y escritura como pueden ser 'Coils' y 'Holding Registers' pueden estar configuradas para cambiar el estado del PLC. Esto implica que la operación de escritura en alguno de estos datos puede desencadenar una operación mecánica del PLC. Gracias a esta funcionalidad del protocolo, un atacante puede inyectar una serie de mensajes Modbus forzando al autómata a entrar en un estado que puede desencadenar una operación indeseada.

La simulación no tiene la capacidad de desencadenar operaciones mecánicas, por lo que en el siguiente escenario se pretende demostrar el efecto del ataque desde el punto de vista del maestro, donde se espera que el dispositivo esclavo devuelva una serie de datos aleatorios. Para realizar el ataque se ha generado un script en Python haciendo uso de la librería Scapy ya mencionada anteriormente. El script va a mandar un tráfico continuado además de cambiar el valor del estado del PLC.

La Figura 6.1 y 6.2 muestran la traza y el contenido de las comunicaciones, respectivamente, en una situación normal (sin ataques). Como se observa, la secuencia de mensajes es la esperada y los datos devueltos por el esclavo presentan un contenido aleatorio.

12	0.196504	192.168.1.11	192.168.1.84	Modbus/TCP	66	Query:	Trans:	0;	Unit:	0, Func: 1: Read Coils
13	0.197460	192.168.1.84	192.168.1.11	Modbus/TCP	64	Response:	Trans:	0;	Unit:	0, Func: 1: Read Coils
19	0.258005	192.168.1.11	192.168.1.84	Modbus/TCP	66	Query:	Trans:	0;	Unit:	0, Func: 2: Read Discrete Inputs
20	0.258750	192.168.1.84	192.168.1.11	Modbus/TCP	63	Response:	Trans:	0;	Unit:	0, Func: 2: Read Discrete Inputs
22	0.318545	192.168.1.11	192.168.1.84	Modbus/TCP	68	Query:	Trans:	0;	Unit:	0, Func: 15: Write Multiple Coils
23	0.319466	192.168.1.84	192.168.1.11	Modbus/TCP	66	Response:	Trans:	0;	Unit:	0, Func: 15: Write Multiple Coils
25	0.379596	192.168.1.11	192.168.1.84	Modbus/TCP	66	Query:	Trans:	0;	Unit:	0, Func: 1: Read Coils
26	0.380691	192.168.1.84	192.168.1.11	Modbus/TCP	64	Response:	Trans:	0;	Unit:	0, Func: 1: Read Coils
28	0.440117	192.168.1.11	192.168.1.84	Modbus/TCP	66	Query:	Trans:	0;	Unit:	0, Func: 2: Read Discrete Inputs

Figura 6.1: Traza de la comunicación sin ataques.

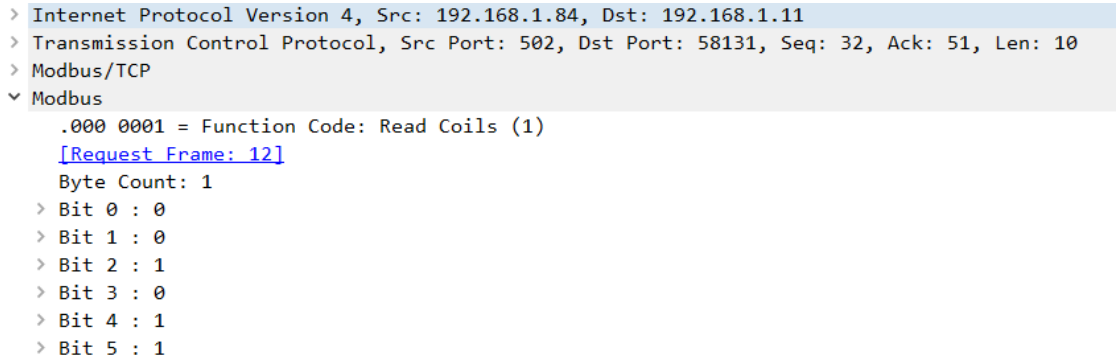


Figura 6.2: Contenido de los datos de la comunicación sin ataques.

En la Figura 6.3 y 6.4 se muestran la traza de comunicación y el contenido de las comunicaciones, respectivamente, una vez iniciado el ataque. Además de observarse un incremento en el número de mensajes, también se observa que la misma petición de lectura de datos que antes devolvía un contenido aleatorio, ahora devuelve todos los bits activos (con el valor 1). Esto es debido a que la máquina atacante ha inyectado un tráfico de forma continuada para modificar estos datos y establecer su valor a uno.

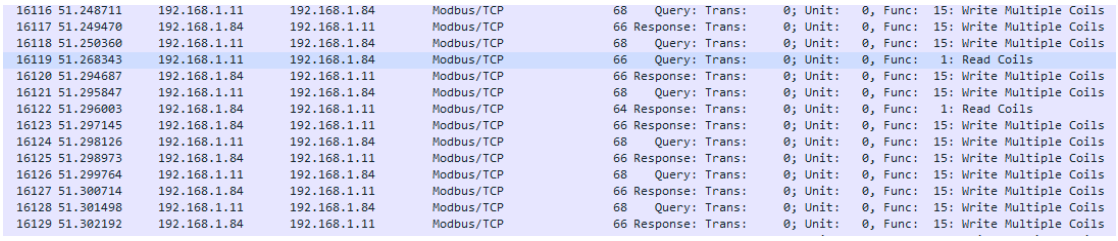


Figura 6.3: Traza de la comunicación con ataques de denegación del servicio.

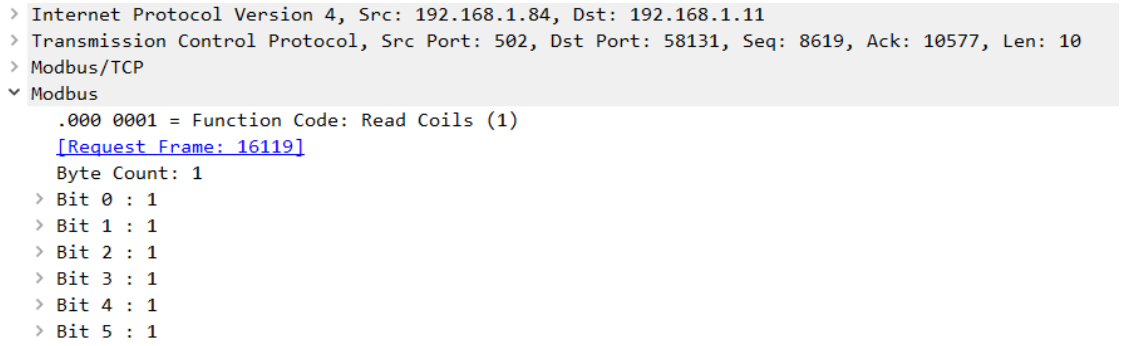


Figura 6.4: Contenido de los datos de la comunicación durante el ataque de denegación del servicio.

Capítulo 7

Conclusiones y trabajo futuro

En este capítulo se hace una comparación de los resultados obtenidos respecto a los objetivos planteados inicialmente y se detallan las posibilidades de continuación del proyecto.

7.1. Conclusiones

El protocolo de comunicación Modbus/TCP mantiene desde su publicación una problemática en la seguridad de la información. La exposición de los datos por la interconexión de los sistemas SCADA derivado de la implementación del protocolo ha provocado el aumento de intrusiones en redes industriales, dejando en evidencia las brechas de seguridad. Cabe destacar que la problemática de la securización y el desarrollo de soluciones radica en la falta de un entorno de simulación apropiado, así como de muestras de tráfico de red real que sirvan como base de experimentación.

En este trabajo, se ha realizado un análisis exhaustivo del comportamiento de una red SCADA y desarrollado una herramienta que permita un proceso de desarrollo de soluciones para las carencias que el protocolo Modbus/TCP presenta. La realización de esta herramienta bajo licencia pública GNU/GPLv3 permite la mejora y evolución del proyecto, así como el avance en la securización de redes de control industrial. Además, la herramienta se ha validado mediante la generación de muestras de tráfico Modbus/TCP equivalentes a un escenario real, lo cual presentaba grandes dificultades a priori.

Por otra parte, se han generado varias herramientas en forma de scripts que permiten la inyección de mensajes Modbus en sistemas SCADA reales, permitiendo mediante su uso la auditoría y análisis de dispositivos que implementen el protocolo Modbus/TCP. El funcionamiento de estos scripts ha sido verificado mediante su uso en entornos tanto reales como de simulación, obteniendo un resultado satisfactorio.

Del mismo modo, se espera que la herramienta desarrollada no sea más que el principio permitiendo la evolución del proyecto, así como la creación de nuevas ramas en el proyecto orientadas a diferentes objetivos dentro de la seguridad de los sistemas SCADA.

7.2. Trabajo futuro

En primer lugar, como trabajo futuro se puede realizar la implementación en entornos simulados de los trabajos teóricos de seguridad como [16], lo que permitiría validar su efectividad frente a ataques. Además, los simuladores desarrollados pueden integrarse con otras herramientas que realicen una detección pasiva de intrusiones en redes de comunicación Modbus, como se propone en [6], para su validación.

Bibliografía

[1] CIRCE Laboratory. [Online; http://www.matfis.unina2.it/OldSite/dipartimento-205/laboratori/circe]. Accedido 20-Agosto-2018.

[2] Great Scott Gadgets. https://greatscottgadgets.com/throwingstar/. [Online; accedido 20-Agosto-2018].

[3] Modbuspal. https://github.com/ControlThingsPlatform/modbuspal.

[4] Schneider electric webpage. https://www.schneider-electric.us/en/. [Online; accedido 20-Agosto-2018].

[5] Schneider Electric. Modbus tester. https://www.schneider-electric.com/en/faqs/FA180037/. [Online; accedido 20-Agosto-2018].

[6] Igor Nai Fovino, Andrea Carcano, Marcelo Masera, and Alberto Trombetta. Design and implementation of a secure modbus protocol. In Charles Palmer and Sujeet Shenoi, editors, *Critical Infrastructure Protection III*, pages 83–96, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[7] General Public License v2.0. Scapy. Packet crafting for Python2 and Python3. https://github.com/secdev/scapy. [Online; accedido 20-Agosto-2018].

[8] T. H. Kobayashi, A. B. Batista, A. M. Brito, and P. S. Motta Pires. Using a packet manipulation tool for security analysis of industrial network protocols. In *2007 IEEE Conference on Emerging Technologies and Factory Automation (EFTA 2007)*, pages 744–747, Sept 2007.

[9] Kaspersky Lab. Threat Landscape for Industrial Automation Systems in H2 2017. Technical report, Kaspersky Lab ICS CERT, 2018.

[10] BSD 2-Clause "Simplified"License. Mock. Python testing library. https://pypi.org/project/mock/.

[11] Modbus-IDA. *MODBUS MESSAGING ON TCP/IP IMPLEMENTATION GUIDE.* Modbus-IDA, 1.0a edition, 6 2004.

[12] Modbus-IDA. *MODBUS APPLICATION PROTOCOL SPECIFICATION.* Modbus-IDA, 1.1b edition, 12 2006.

[13] R. Nardone, R. J. Rodríguez, and S. Marrone. Formal security assessment of Modbus protocol. In *2016 11th International Conference for Internet Technology and Secured Transactions (ICITST)*, pages 142–147, Dec 2016.

[14] General Public License v2.0. Modbus Network Analysis Scripts. `https://github.com/zhanglongqi/qModMaster`.

[15] A. G. Voyiatzis, K. Katsigiannis, and S. Koubias. A modbus/tcp fuzzer for testing internetworked industrial systems. In *2015 IEEE 20th Conference on Emerging Technologies Factory Automation (ETFA)*, pages 1–6, Sept 2015.

[16] Dong Wei, Yan Lu, M. Jafari, P. Skare, and K. Rohde. An integrated security system of protecting smart grid against cyber attacks. In *2010 Innovative Smart Grid Technologies (ISGT)*, pages 1–7, Jan 2010.

Apéndice A

Desglose de trabajo

Tarea \ Semana	1 ene	2 ene	3 ene	4 ene	1 feb	2 feb	3 feb	4 feb	2 ma	3 ma	4 ma	1 abr	2 abr	3 abr	4 abr	1 ma	2 ma	3 ma	1 jun	2 jun	3 jun	4 jun	1 jul	2 jul	3 jul	4 jul	1 ago	2 ago	3 ago	TOTAL
Comprensión Protocolo	2	2	2	1	1																									8
Documentación de estudios	2	2	2																											6
Analisis de herramientas				2	2	2	1																3							10
Desarrollo scripts de inyeccón																1	1	4	12	22						5	5	5		55
Desarrollo scripts de procesado							2	4	11	9	0	9	1	2	2											1	1	1		43
Analisis de escenario													2	2	5	2	2													13
Diseño de simuladores																				8				1						9
Desarrollo de simuladores																					9	9	8	7	8					41
Validaciones																							1	1	1	1	1			5
Correcciones de simuladores																											4	4		8
Reproducción de ataques																										4	1			5
Elaboración de la memoria																											30	40	40	110
																														313

Figura A.1: Diagrama de Gantt.